Hierarchical-matrix Preconditioners for Parabolic Optimal Control Problems

Suely Oliveira[1] and Fang Yang[1]

The University of Iowa, Iowa City IA 52242, USA

Abstract. Hierarchical ($\mathcal{H}$)-matrices approximate full or sparse matrices using a hierarchical data sparse format. The corresponding $\mathcal{H}$-matrix arithmetic reduces the time complexity of the approximate $\mathcal{H}$-matrix operators to almost optimal while maintains certain accuracy. In this paper, we represent a scheme to solve the saddle point system arising from the control of parabolic partial differential equations by using $\mathcal{H}$-matrix LU-factors as preconditioners in iterative methods. The experiment shows that the $\mathcal{H}$-matrix preconditioners are effective and speed up the convergence of iterative methods.

1 Introduction

Hierarchical-matrices ($\mathcal{H}$-matrices) [9], since their introduction [1, 2, 7, 9], have been applied to various problems, such as integral equations and partial differential equations. The idea of $\mathcal{H}$-matrices is to partition a matrix into a hierarchy of rectangular subblocks and approximate the subblocks by low rank matrices (Rk-matrices). The $\mathcal{H}$-matrix arithmetic [1, 7, 8] defines operators over the $\mathcal{H}$-matrix format. The fixed-rank $\mathcal{H}$-matrix arithmetic keeps the rank of a Rk-matrix block below a fixed value, whereas the adaptive-rank $\mathcal{H}$-matrix arithmetic adjusts the rank of a Rk-matrix block to maintain certain accuracy in approximation. The operators defined in the $\mathcal{H}$-matrix arithmetic include $\mathcal{H}$-matrix addition, $\mathcal{H}$-matrix multiplication, $\mathcal{H}$-matrix inversion, $\mathcal{H}$-matrix LU factorization, etc. The computation complexity of these operators are almost optimal $O(n \log^a n)$.

The $\mathcal{H}$-matrix construction for matrices from discretization of partial differential equations depends on the geometric information underlying the problem [7]. The admission conditions, used to determine whether a subblock is approximated by a Rk-matrix, are typically based on Euclidean distances between the supports of the basis functions. For sparse matrices the algebraic approaches [8, 11] can be used, which use matrix graphs to convert a sparse matrix to an $\mathcal{H}$-matrix by representing the off-diagonal zero blocks as Rk-matrices of rank 0.

Since the $\mathcal{H}$-matrix arithmetic provides cheap operators, it can be used with H-matrix construction approaches to construct preconditioners for iterative methods, such as Generalized Minimal Residual Method (GMRES), to solve systems of linear equations arising from finite element or meshfree discretizations of partial differential equations [3, 4, 5, 8, 11].

In this paper we consider the finite time linear-quadratic optimal control problems governed by parabolic partial differential equations. To solve these

problems, in [12] the parabolic partial differential equations are discretized by finite element methods in space and by θ-scheme in time; the cost function J to be minimized is discretized using midpoint rule for the state variable and using piecewise constant for the control variable in time; Lagrange multipliers are used to enforce the constraints, which result a system of saddle point type; then iterative methods with block preconditioners are used to solve the system.

We adapt the process in [12] and use $\mathcal{H}$-matrix preconditioners in iterative methods to solve the system. First we apply algebraic $\mathcal{H}$-matrix construction approaches to represent the system in the $\mathcal{H}$-matrix format; then $\mathcal{H}$-LU factorization in the $\mathcal{H}$-matrix arithmetic is adapted to the block structure of the saddle point system to compute the approximate $\mathcal{H}$-LU factors; at last, these factors are used as preconditioner in iterative methods to compute the approximate solutions. The numerical results show that the $\mathcal{H}$-matrix preconditioned approach is competitive and effective to solve the above optimal control problem.

This paper is organized as follows. In Sect. 2 we introduce the optimal control model problem and the discretization process; Section 3 is an introduction to $\mathcal{H}$-matrices; in Sect. 4, we review the algebraic approaches to $\mathcal{H}$-matrix construction; in Sect. 5 we present the scheme to build the $\mathcal{H}$-matrix preconditioners; finally in Sect. 6 we present the numerical results.

2 The Optimal Control Problem

The model problem [12] is to minimize the following quadratic cost function:

$$\begin{aligned} J(z(u),u) :=& \frac{q}{2}\|z(v)-z_*\|^2_{L^2(t_0,t_f;L^2(\Omega))} + \frac{r}{2}\|v\|^2_{L^2(t_0,t_f;\Omega)} \\ &+ \frac{s}{2}\|z(v)(t_f,x)-z_*(t_f,x)\|^2_{L^2(\Omega)} \end{aligned} \tag{1}$$

under the constraint of the state equation:

$$\begin{cases} \partial_t z + \mathcal{A}z = \mathcal{B}v, \quad t \in (t_0,t_f) \\ z(t,\partial\Omega) = 0 \\ z(t_0,\Omega) = 0 \end{cases} \tag{2}$$

, where the state variable $z \in Y = H^1_0(\Omega)$ and the control variable $v \in U = L^2(t_0,t_f;\Omega)$. $\mathcal{B}$ is an operator in $\mathcal{L}(L^2(t_0,t_f;\Omega), L^2(t_0,t_f;Y'))$ and $\mathcal{A}$ is an uniformly elliptic linear operator from $L^2(t_0,t_f;Y)$ to $L^2(t_0,t_f;Y')$. The state variable z is dependent on v. z_* is a given target function.

2.1 Discretization in Space

The system is first discretized in space by fixing t. Considering the discrete subspace $Y_h \in Y$ and $U_h \in U$, then the discretized weak form of (2) is given as:

$$(\dot{z}_h(t),\eta_h) + (\mathcal{A}z_h(t),\eta_h) = (\mathcal{B}u_h(t),\eta_h), \quad \forall \eta_h \in Y_h \text{ and } t \in (t_0,tf)\ . \tag{3}$$

Let$\{\phi_1, \phi_2, .., \phi_n\}$ be a basis of Y_h and $\{\psi_1, \psi_2, .., \psi_m\}$ be a basis of U_h, where $m \leq n$. Apply the finite element methods to (3), we obtain the following system of ordinary differential equations:

$$M\dot{y} + Ay = Bu, \quad t \in (t_0, t_f) \ . \tag{4}$$

Here $A_{i,j} = (\mathcal{A}\phi_j, \phi_i)$ is a stiffness matrix, $M_{i,j} = (\phi_j, \phi_i)$ and $R_{i,j} = (\psi_j, \psi_i)$ are mass matrices, and $B_{i,j} = (\mathcal{B}\psi_i, \phi_j)$. The semi-discrete solution is $z_h(t,x) = \sum_i y_i(t)\phi_i(x)$ with control function $u_h(t,x) = \sum_i u_i(t)\psi_i(x)$.

We can apply the analogous spatial discretization to the cost function (1), and obtain:

$$J(y,u) = \int_{t_f}^{t_0} e(t)^T Q(t) e(t) + u(t)^T R(t) u(t)\, dt + e(t_f)^T C(t) e(t_f) \tag{5}$$

, where $e(\cdot) = y(\cdot) - y_*(\cdot)$ is the difference between the state variable and the given target function.

2.2 Discretization in Time

After spatial discretization, the original optimal problem is transferred into minimizing (5) under the constraint of n ordinary differential equations (4). θ-scheme is used to discretize the above problem.

First the time scale is subdivided into l intervals of length $\tau = (t_f - t_0)/l$. Let $F_0 = M + \tau(1-\theta)A$ and $F_1 = M - \tau\theta A$. The discretization of equation (4) is given by:

$$E\mathbf{y} + N\mathbf{u} = \mathbf{f} \tag{6}$$

, where

$$E = \begin{bmatrix} -F_1 & & \\ \ddots & \ddots & \\ & F_0 & -F_1 \end{bmatrix}, \qquad N = \tau \begin{bmatrix} B & & \\ & \ddots & \\ & & B \end{bmatrix}, \quad \mathbf{y} \approx \begin{bmatrix} y(t_1) \\ \vdots \\ y(t_n) \end{bmatrix}, \quad \text{etc} \ . \tag{7}$$

Then discretize the cost function (5) by using piecewise linear functions to approximate the state variable and piecewise constant functions to approximate the control variable and obtain the following discrete form of (5):

$$J(\mathbf{y}, \mathbf{u}) = \mathbf{u}^T G \mathbf{u} + \mathbf{e}^T K \mathbf{e} \tag{8}$$

, where $\mathbf{e} = \mathbf{y} - \mathbf{y}_*$ and the target trajectory $z_*(t,x) \approx z_{*,h}(t,x) = \sum_i (y_*)_i(t)\phi_i(x)$. A Lagrange multiplier vector p is introduced to enforce the constraint of (6), and we have the Lagrangian

$$\mathcal{L}(\mathbf{y}, \mathbf{u}, \mathbf{p}) = \frac{1}{2}(\mathbf{u}^T G \mathbf{u} + \mathbf{e}^T K \mathbf{e}) + \mathbf{p}^T (E\mathbf{y} + N\mathbf{u} - \mathbf{f}) \ . \tag{9}$$

To find $\mathbf{y}$, $\mathbf{u}$ and $\mathbf{p}$ where $\nabla\mathcal{L}(\mathbf{y}, \mathbf{u}, \mathbf{p}) = 0$ in (9), we need to solve the following system:

$$\begin{bmatrix} K & 0 & E^T \\ 0 & G & N^T \\ E & N & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} M\mathbf{y}_* \\ 0 \\ \mathbf{f} \end{bmatrix} \ . \tag{10}$$

3 Hierarchical-matrices

The concept and properties of $\mathcal{H}$-matrices are induced by the index cluster tree T_I and the block cluster tree $T_{I\times I}$ [9] . In the rest of this paper, $\#A$ denotes the number of elements of set A and $S(i)$ denotes the children of node i.

3.1 Index Cluster Tree and Block Cluster Tree

An index cluster tree T_I defines a hierarchical partition tree over an index set $I = (0, \ldots, n-1)$. Note that $(0, 1, 2) \neq (0, 2, 1)$.T_I has the following properties: its root 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 pairwise disjoint.

A block cluster tree $T_{I\times I}$ is a hierarchical partition tree over the product index set $I\times I$. Given tree T_I and an admissibility condition (see below), $T_{I\times I}$ can be constructed as follows: its root is $I\times I$; if $s\times t$ in $T_{I\times I}$ satisfies the admissibility condition, it is an Rk-matrix leaf; else if $\#s < N_s$ or $\#t < N_s$, it is a full-matrix leaf; otherwise it is partitioned into subblocks on the next level and its children (subblocks) are defined as $S(s \times t) = \{\, i \times j \mid i, j \in T_I \text{ and } i \in S(s),\ j \in S(t) \,\}$. A constant $N_s \in [10, 100]$ is used to control the size of the smallest blocks.

An admissibility condition is used to determine whether a block to be approximated by an Rk-matrix. An example of an admissibility condition is:

$$s \times t \text{ is admissible if \& only if : } \min(\text{diam(s)}, \text{diam(t)}) \leq \mu \ \text{dist}(s, t) \qquad (11)$$

, where diam(s) denotes the Euclidean diameter of the support set s, and dist(s, t) denotes the Euclidean distance of the support set s and t. The papers [1, 2, 7] give further details on adapting the admissibility condition to the underlying problem or the cluster tree.

Now we can define an $\mathcal{H}$-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 if $\#s < N_s$ or $\#t < N_s$.

Fig. 1 shows an example of T_I, $T_{I\times I}$ and the corresponding $\mathcal{H}$-matrix.

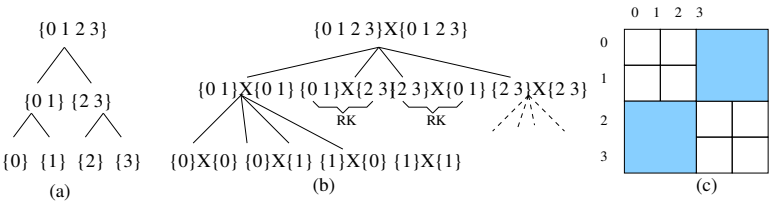

Fig. 1. (a) is T_I, (b) is $T_{I\times I}$ and (c) is the corresponding $\mathcal{H}$-matrix. The dark blocks in (c) are Rk-matrix blocks and the white blocks are full matrix blocks.

An $m \times n$ matrix M is called an Rk-matrix if $rank(M) \leq k$ and it is represented in the form of a matrix product $M = AB^T$, where M is $m \times n$, A is

$m \times k$ and B is $n \times k$. If M is not of rank k, then a rank k approximation can be computed in $O(k^2(n+m)+k^3)$ time by using a truncated Singular Value Decomposition (SVD) [1, 7].

3.2 $\mathcal{H}$-matrix Arithmetic

The following is a short summary of the $\mathcal{H}$-matrix arithmetic. A detailed introduction can be found in [1, 2].

$\mathcal{H}$-matrix operations perform recursively; therefore it is important to define the corresponding operations on the leaf subblocks, which are either full or Rk-matrices. These operations are approximate as certain operations do not create Rk-matrices (such as adding two Rk-matrices). In such case, a truncation is performed using an SVD to compute a suitable Rk-matrix. For example, the sum of two rank k matrices can be computed by means of a $2k \times 2k$ SVD.

The computational complexity of the $\mathcal{H}$-matrix arithmetic strongly depends on the structure of $T_{I\times I}$. Under fairly general assumptions on the block cluster tree $T_{I\times I}$ the complexity of $\mathcal{H}$-matrix operators is $O(n \log^{\alpha} n)$ [9, 7].

4 Algebraic Approaches for Hierarchical-matrix Construction

Algebraic $\mathcal{H}$-matrix construction approaches can be applied to sparse matrices. These approaches take advantage that most entries of a sparse matrix are zeros. They build $\mathcal{H}$-matrix cluster tree by partitioning a matrix graph either bottom-up or top-down. The multilevel clustering based algorithm [11] constructs the cluster tree "bottom-up", i.e., starts with the leaves and successively clusters them together until the root is reached. Domain decomposition in [8] and bisection are "top-down" algebraic approaches, which start with the root and successively subdivides a cluster into subsets.

4.1 Algebraic Approaches to Construct an Index Cluster Tree

In [11] we propose an $\mathcal{H}$-matrix construction approach based on multilevel clustering methods. To build clusters over the nodes in $G_i = (V(G_i), E(G_i))$, an algorithm based on Heavy Edge Matching (HEM) [10] is used. 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 a node $k \in V(G_{i+1})$; the edge weight w_{kt} of edge $e_{kt} \in E(G_{i+1})$ is 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 graph G_i. Recursively applying the above coarsening process gives a sequence of coarse graphs $G_1, G_2, \ldots, G_h$. The index cluster tree T_I is constructed by making $k \in V(G_i)$ the parent of every $s \in C_k^{(i)}$. The root of T_I is the set $V(G_h)$, which is the parent to all nodes in G_h.

In [8], domain decomposition based clustering is used to build a cluster tree T_I. Starting from I, a cluster is divided into three sons, i.e. $S(c) = \{c_1, c_2, c_3\}$

and $c = c_1 \cup c_2 \cup c_3$, so that the domain-clusters c_1 and c_2 are disconnected and the interface-cluster c_3 is connected to both c_1 and c_2. Then the domain-clusters are successively divided into three subsets, and the interface-clusters are successively divided into two interface-clusters until the size of a cluster is small enough.

To build a cluster tree T_I based on bisection is straight forward. Starting from a root set I and a set is successive partitioned into two subsets with equal size. This construction approach is suitable for the sparse matrices where the none zero entries are around the diagonal blocks.

4.2 Block Cluster Tree Construction for Algebraic Approaches

The admissibility condition used to build $T_{I\times I}$ for the algebraic approaches is defined as follows: a block $s \times t \in T_{I\times I}$ is admissible if and only if s and t are not joined by an edge in the matrix graph; an admissible block corresponds to a zero block and is represented as a Rk-matrix of rank zero; an inadmissible block is partitioned further or represented by a full matrix.

5 Hierarchical-matrix Preconditioners

The construction of $\mathcal{H}$-matrix preconditioners for a system of saddle point type is based on the block LU factorization.

To obtain a relative cheap yet good approximate LU factors, we replace the ordinary matrix operators by the corresponding $\mathcal{H}$-matrix operators[3, 8, 11].

First the matrix in (10) is converted to an $\mathcal{H}$-matrix. Since the nonzero entries of each subblock are centered around the diagonal blocks, we apply the bisection approach to the submatrix K, G, E and N respectively. Then we obtain the following $\mathcal{H}$-matrix, which is on the left side of the equation ($_\mathcal{H}$ indicates a block in the $\mathcal{H}$-matrix format):

$$\begin{bmatrix} K_\mathcal{H} & 0 & E_\mathcal{H}^T \\ 0 & G_\mathcal{H} & N_\mathcal{H}^T \\ E_\mathcal{H} & N_\mathcal{H} & 0 \end{bmatrix} = \begin{bmatrix} L1_\mathcal{H} & 0 & 0 \\ 0 & L2_\mathcal{H} & 0 \\ M1_\mathcal{H} & M2_\mathcal{H} & L3_\mathcal{H} \end{bmatrix} \begin{bmatrix} U1_\mathcal{H} & 0 & M1_\mathcal{H}^T \\ 0 & U3_\mathcal{H} & M2_\mathcal{H}^T \\ 0 & 0 & U3_\mathcal{H} \end{bmatrix} . \tag{12}$$

The block cluster tree $T_{I\times I}$ of $L1_\mathcal{H}$, $L2_\mathcal{H}$, $M1_\mathcal{H}$, and $M2_\mathcal{H}$ is same as the block cluster tree structure of $K_\mathcal{H}$, $G_\mathcal{H}$, $E_\mathcal{H}$, and $N_\mathcal{H}$ respectively. The block cluster tree structure of $L3_\mathcal{H}$ is based on the block tree structure of $E_\mathcal{H}$: the block tree of $L3_\mathcal{H}$ is symmetric; the tree structure of the lower-triangular of $L3_\mathcal{H}$ is same as the tree structure of the lower-triangular of $E_\mathcal{H}$; the tree structure of the upper-triangular of $L3_\mathcal{H}$ is the transpose of the tree structure of the lower triangular. $L1_\mathcal{H}$ and $L2_\mathcal{H}$ are obtained by apply $\mathcal{H}$-Cholesky factorization to $K_\mathcal{H}$ and $G_\mathcal{H}$: $K_\mathcal{H} = L1_\mathcal{H} *_\mathcal{H} U1_\mathcal{H}$ and $G_\mathcal{H} = L2_\mathcal{H} *_\mathcal{H} U2_\mathcal{H}$. Then use the $\mathcal{H}$-matrix upper triangular solve, we can get $M1_\mathcal{H}$ by solving $M1_\mathcal{H}U1_\mathcal{H} = E_\mathcal{H}$. $M1_\mathcal{H}$ have the same block tree as $E_\mathcal{H}$. In the same way we can compute $M2_\mathcal{H}$, which has the same block cluster tree structure as $N_\mathcal{H}$. At last we construct the block cluster tree for $L3_\mathcal{H}$ and then apply $\mathcal{H}$-LU factorization to get $L3_\mathcal{H}$: $L3_\mathcal{H}U3_\mathcal{H} = M1_\mathcal{H} *_\mathcal{H} M1_\mathcal{H}^T +_\mathcal{H} M2_\mathcal{H} *_\mathcal{H} M2_\mathcal{H}^T$.

6 Experimental Results

In this section, we present the numerical results of solving the optimal control problem (1) constrained by the following equation:

$$\begin{cases} \partial_t z - \partial_{xx} z = v, t \in (0,1), x \in (0,1) \\ z(t,0) = 0, z(t,1) = 0 \\ z(0,x) = 0, x \in [0,1] \end{cases} \tag{13}$$

with the target function $z_*(t,x) = x(1-x)e^{-x}$. The parameters in the control function J are $q = 1$, $r = 0.0001$ and $s = 0$.

GMRES iteration stops where the original residuals are reduced by the factor of 10^{-12}. The convergence rate a ia defined as the average decreasing speed of residuals in each iteration. The fixed-rank $\mathcal{H}$-matrix arithmetic is used and we set the rank of each Rk-matrix block to be ≤ 2. The tests are performed on a Dell workstation with AMD64 X2 Dual Core Processors (2GHz) and 3GB memory.

Table 1 shows the time to compute the different parts of the $\mathcal{H}$-LU factors and the time of GMRES iterations (in seconds). n is the size of the problem, $n1$ and $n2$ is the number of rows of K and G respectively. Based on Table 1, the time to compute $L3_{\mathcal{H}}$ contributes the biggest part of the total time to set up the preconditioner.

Table 1. Time for computing $\mathcal{H}$-LU factors and GMRES iterations

n(n1/n2)	$L1_{\mathcal{H}}$	$L2_{\mathcal{H}}$	$M1_{\mathcal{H}}$	$M2_{\mathcal{H}}$	$L3_{\mathcal{H}}$	GMRES iteration time	number of GMRES iterations
592(240/112)	0	0	0.01	0	0.01	0	1
2464(992/480)	0.01	0	0.01	0	0.04	0	1
10048(4032/1984)	0.03	0.01	0.13	0.02	0.39	0.04	1
40576(16256/8064)	0.21	0.06	0.84	0.26	4.13	0.32	3
163072(65280/32512)	1.09	0.42	4.23	1.74	25.12	2.66	6

Fig. 2-(a) shows the convergence rate of the $\mathcal{H}$-LU preconditioned GMRES and Fig. 2-(b), plotted on a log-log scale, shows the time for building the preconditioners and the time for the GMRES iterations.

Based the results, we can see that $\mathcal{H}$-LU speeds up the convergence of GMRES iteration significantly. The problem in our implementation is that the time to compute L_3 still consists a significant part of the LU-factorization time. In the future, more work needs to be done to reduce the complexity of computing L_3 further. More discussion about $\mathcal{H}$-matrix preconditioners and applications will come in [13].

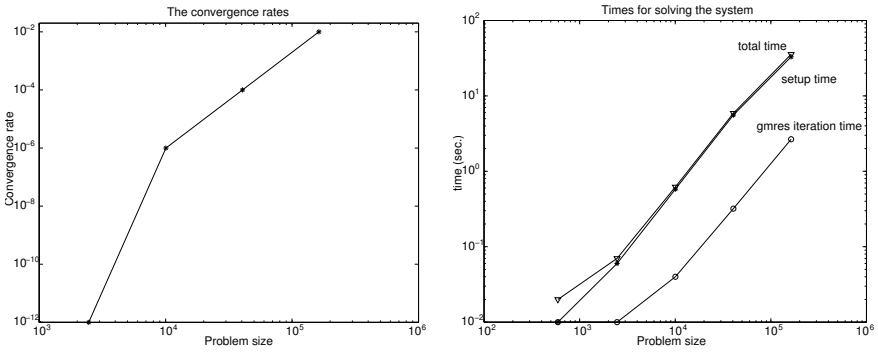

Fig. 2. (a) The convergence rates of GMRES. (b) Total times for solving the system

References

[1] S. Börm, L. Grasedyck, and W. Hackbusch. Introduction to hierarchical matrices with applications. *EABE*, 27:403–564, 2003.

[2] S. Börm, L. Grasedyck, and W. Hackbush. Hierarchical matrices. 2005.

[3] S. Le Borne. Hierarchical matrix preconditioners for the Oseen equations. 2006. to appear.

[4] S. Le Borne and L. Grasedyck. $\mathcal{H}$ preconditioners in convection-dominated problems. *SIAM J. Matrix Anal. Appl.*, 27(4):1172–1183, 2006.

[5] Sabine Le Borne, Lars Grasedyck, and Ronald Kriemann. Domain-decomposition based $\mathcal{H}$-LU preconditioners. In *Proceedings of the 16th International Conference on Domain Decomposition Methods (New York, 2005)*, LNCSE. Springer, 2006. To appear.

[6] G.A.Gravvanis. Explicit approximate inverse preconditioning techniques. *Archives of Computational Methods in Engeneering*, 9(4), 2002.

[7] L. Grasedyck and W. Hackbusch. Construction and arithmetics of $\mathcal{H}$-matrices. *Computing*, 70(4):295–334, 2003.

[8] L. Grasedyck, R. Kriemann, and S. Le Borne. Parallel black box domain decomposition based H-LU preconditioning. 2005.

[9] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62:89–108, 1999.

[10] 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.

[11] S. Oliveira and F. Yang. An algebraic approach for $\mathcal{H}$-matrix preconditioners. *computing, submmitted*, 2006.

[12] Christian E. Schaerer, Tarek Mathew, and Marcus Sarkis. Block iterative algorithms for the solution of parabolic optimal control problems. vecpar, 2006.

[13] Fang Yang. $\mathcal{H}$-matrix preconditioners and applications. *PhD thesis, the University of Iowa*.