International Journal of Applied Mathematics

Volume 26 No. 1 2013, 103-122
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
doi: http://dx.doi.org/10.12732/ijam.v26i1.10

A NUMERICAL STUDY OF HIERARCHICAL MATRIX ($\mathcal{H}$-MATRIX) FOR FINITE POINT SET METHOD (FPM) ON SOLVING A POISSON PROBLEM

Satyanarayana Gedela[1 §], S. Sundar[2]
[1,2]Department of Mathematics
Indian Institute of Technology (IIT) Madras
Chennai, 600 036, INDIA

Abstract: A class of matrices called $\mathcal{H}$-matrices is used to approximate large matrices of grid based methods. $\mathcal{H}$-matrix arithmetic has linear-logarithmic complexity for some special grids. In this paper, we present a study on $\mathcal{H}$-matrix for a large sparse un-patterned system arising from a grid free method called finite pointset method applied on a Poisson problem with mixed boundary conditions.

AMS Subject Classification: 46N40, 65F10, 35G15
Key Words: finite pointset method, $\mathcal{H}$-matrices, SAMG format, sparse matrices, BiCGStab

1. Introduction

Solving a partial differential equation in computational fluid dynamics using available numerical techniques involves solving a large sparse linear system, see [9]. The computation time for solving a large linear system is always high. There are many ways to reduce the computation time in the implementation level. Researchers are using now a days a new technique called GPU accelerated linear

Received: January 28, 2013

[§]Correspondence author

solver, GPU accelerated numerical technique etc. For example, Panchatcharam et al. [18] have proved the GPU acceleration for a $2D$-dam break problem with finite pointset method. One can reduce computation time by accessing the matrices in an efficient way.

Accessing a matrix directly consumes more time to solve a large sparse linear system. We need to reduce the computation cost by efficiently accessing matrices. In 1998, Hackbusch et al. [26] have developed a new theory called $\mathcal{H}$-matrices to access the matrix efficiently for grid based methods. These $\mathcal{H}$-matrices are a class of matrices, used as data-approximations of densely populated matrices. $\mathcal{H}$-matrices techniques are found to be successful in finding approximate inverses of sparse matrices in finite element methods and densely populated matrices arise from boundary element methods to solve partial differential equations, see [7].

With the use of data sparse matrix representations, the densely populated matrix can be stored in linear complexity. In addition to efficiently storing matrices, $\mathcal{H}$-matrix arithmetic operations such as addition, matrix vector multiplication, matrix-matrix multiplication, factorization or inversion can be approximated in $\mathcal{O}(nk^{\alpha}log^{\beta}(n)$ operations, where $\alpha, \beta \in 1, 2, 3$, see [28].

$\mathcal{H}$-matrices analysis has been done for grid based methods by many researchers. For instance, Hackbusch et al. [31] have applied $\mathcal{H}$-matrix techniques to boundary integral methods. A spacial class of $\mathcal{H}$-matrices on graded meshes is analyzed in [29] and $\mathcal{H}$-matrix approximation for the operator exponential with application discussed in [10]. In which, they proved that the optimal complexity and approximation results in the case of composite meshes and tensor-product meshes with polynomial, exponential grading in $\mathbb{R}^d$, $d = 1, 2, 3$ in both cardinality-balancing strategy or distance-balancing strategy.

The construction of $\mathcal{H}$-matrices involves three steps: the first step is to construct an index cluster tree T_I; the second step is to define an admissibility condition; the last step is to construct the block cluster tree $T_{I \times I}$, using the index cluster tree and the admissibility condition.

The leaves of the block cluster tree $T_{I \times I}$ describes hierarchical block partition of a matrix and the nodes in the block cluster tree named as admissible, non-admissible blocks. Using the Taylor series, interpolation [33], cross approximation [19] and Hybrid cross approximation [6], the admissible blocks (P_{far} blocks) can be approximated as rank k subblocks and non-admissible block (P_{near} blocks) retain as full matrices. The main advantage of this rank k matrix representation is that it can be reduce the storage, computational complexity of matrix operations.

Depending on the information used in the $\mathcal{H}$-matrix construction process, construction approaches can be divided into two categories: geometric construction approaches and algebraic construction approaches.

Using the geometric information of the underlying problems, the concept of $\mathcal{H}$-matrices was first introduced in [26, 28]. In which, Hackbusch and Khoromskij have studied the construction of $\mathcal{H}$-matrices and $\mathcal{H}$-matrix technique is explained for Finite Element Method (FEM) and Boundary Element Method (BEM) applications in two and three dimensions. Further, they have proved that $\mathcal{H}$-matrix are data sparse and allow an approximate matrix arithmetic of almost linear complexity. The basic idea of this $\mathcal{H}$-matrix arithmetics is formulated in [26] and a general approach is contained in [14]. The construction of $\mathcal{H}$-matrices on rectangular and triangular meshes has been proposed and analyzed by Hackbusch et al. [27]. Later, they [30] have studied $\mathcal{H}$-matrices based on a weak admissibility criterion. Koch et al. [11] have studied $\mathcal{H}$-matrix methods for linear and quasi-linear integral operators appearing in population balances and obtained linear complexity for matrix arithmetic. Lintner [17] have achieved $\mathcal{H}$-matrix arithmetic up to logarithmic factors for an eigenvalue problem for the $2D$-Laplacian and this author applied $\mathcal{H}$-matrix arithmetic to the heat and wave equation. The paper [16] shows a block decomposition approach for constructing a class of $\mathcal{H}$-matrices to represent the $2D$ stiffness and mass matrices arising from FEM applications, which uses the domain and grid information to construct the index cluster tree and block cluster tree. When the geometric information of underlying problem is not available then algebraic approaches are required to construct $\mathcal{H}$-matrices.

In [21] Oliviera and Yang discussed an algebraic approach to construct $\mathcal{H}$-matrices. In which, it uses the information of a matrix, multilevel clustering methods based on heavy edge matching to construct cluster tree without depending on the geometric information of the underlying problem. The main advantage of this algebraic approach is that the $\mathcal{H}$-matrices constructed in this way are very suitable for $\mathcal{H} - LU$ decomposition, $\mathcal{H}$-Cholesky factors. Bebendorf et al. [16] dealt with the existence of $\mathcal{H}$-matrices to inverses of FEM matrices in the case of uniformly elliptic operators with L^{∞}-coefficients. Recently Grasedyck et al. [13] have used domain decomposition based $\mathcal{H} - LU$ preconditioning technique in the iterative solution of the discrete (three-dimensional) convection-diffusion equation. These $\mathcal{H}$-inverse, $\mathcal{H} - LU$, $\mathcal{H}$-Cholesky can be used as preconditioners in iterative methods, such as GMRES, BicgStab [9, 8].

In finite element methods, the inverse of the stiffness matrix is densely populated and it can be approximated by an $\mathcal{H}$-matrix. The complexity of the $\mathcal{H}$-matrix inversion is $\mathcal{O}(nlog^2k^2n)$ which is almost linear, but there are many relatively large constants hidden. To overcome this draw backs, a weak admissibility condition is used in [30] to construct coarser block structures and it leads to smaller constants in the complexity. In [15, 13, 17] it is introduced an $\mathcal{H} - LU$ decomposition which is computed faster than $\mathcal{H}$-inverse and yields more accurate preconditioners. The parallelization of the $\mathcal{H}$-matrix arithmetic [12] also leads smaller constants in the complexity.

$\mathcal{H}$-matrix technique have been improved as $\mathcal{H}^2$-matrices by Hackbusch et al. [32]. They have introduced $\mathcal{H}^2$-matrices for storing discretizations of elliptic problems and integral operators from the BEM and proved that their arithmetic are quasi-linear complexity. Steffen [4] have used this technique on non-local operators. $\mathcal{H}^2$-matrices employed for nested local expansion systems in order to approximate matrices in optimal order of complexity. Also, $\mathcal{H}^2$-matrix approximation of integral operators by interpolation has been studied by Hackbusch et al. [33]. An approximate of integral operators by $\mathcal{H}^2$-matrices by using adaptive bases discussed in [2]. The complexity of $\mathcal{H}^2$-matrices briefly discussed in [5]. In [3], Steffen have investigated the problem of computing inverses of stiffness matrices resulting from the finite element discretization of elliptic partial differential equations.

Most of the $\mathcal{H}$-matrix analysis has been done for grid based methods. But applying this $\mathcal{H}$-matrix technique on grid free methods like Reproducing Kernel Particle Method(RKPM), Smoothed Particle Hydrodynamics (SPH), Moving Least Square methods (MLS), Finite Pointset Method (FPM) etc., is great challenging task. In [21], Oliveira and Yang applied RKPM Mesh free scheme for discretize a saddle point system and constructed $\mathcal{H}$-inverse, $\mathcal{H}$-Cholesky preconditioners by using Heavy Edge Matching (HEM) and Nested Dissection (ND) approaches. Finally, they proved that ND is the best among the preconditioners with respect to the total running time and convergence rates and HEM preconditioners is comparable.

The Finite Pointset Method (FPM) developed by Tiwari et al. [24, 23] is a fully Lagrangian meshfree particle method. In this method, the computational domain is filled by a finite number of particles which move with fluid velocities and they carry the fluid quantities such as pressure, temperature, density and so on. Tiwari et al. [24] have applied FPM technique for incompressible Navier-Stokes equation and later they have employed it for industrial problems [25, 22]. In incompressible flow simulation, at each time step, we need to solve pressure Poisson equation with mixed boundary conditions which is more time consuming process to solve. Discretization of pressure Poisson problem using FPM involves solving a large sparse linear system. Since solving the linear system arising from FPM consumes more time, it leads the entire simulation to be expensive.

In this paper, we present a study on $\mathcal{H}$-matrices analysis for grid free method, say finite pointset method on solving a Poisson problem. Application of $\mathcal{H}$-matrix technique on Poisson solver and its computational efficiency is presented.

This paper is organized as follows: In Section 2, we introduce the governing equations and numerical methods. Section 3 explains the details of $\mathcal{H}$-matrix and its arithmetic. Implementation of $\mathcal{H}$-matrix on FPM has been described in Section 4. Finally, the results and computational efficiency are presented in Section 5.

2. Governing Equations and Numerical Technique

The Pressure Poisson problem is considered to be important since in a typical transient incompressible flow problems at every time step, we need to solve pressure Poisson equation with mixed boundary conditions which is the most time consuming process. Hence, we consider the following Poisson equation

$$\Delta p = f \text{ in } \Omega \subset R^2 \tag{1}$$

with mixed boundary value condition

$$q_1 p + q_2 \frac{\partial p}{\partial \vec{n}} = g \text{ on } \partial\Omega, \tag{2}$$

where q_1 and q_2 are assumed to be constants.

2.1. Finite Pointset Method

Let us approximate the spatial derivatives of p at a particle position $\vec{x}$ around its neighboring points. To control the number of points in the neighborhood, we use the Gaussian weight function

$$w(\vec{x_i} - \vec{x}; h) = \begin{cases} \exp(-\alpha \frac{\left\|\vec{x_i} - \vec{x}\right\|^2}{h^2}) & \text{if } \frac{\left\|\vec{x_i} - \vec{x}\right\|}{h} \leq 1, \\ 0 & \text{else,} \end{cases} \tag{3}$$

where $\alpha = 6.25$ and h is the size of the compact support. Let $N_h(\vec{x}) = \{\vec{x_i} : i = 1, 2, ..., m\}$ be the set of neighboring points of $\vec{x}$ in a ball of radius h. For consistency reasons, in $2D$ there should be at least 6 particles and they should be neither be on the same line nor on a circle.

Consider Taylor's expansion of p_i around $\vec{x}$

$$\begin{aligned} p_i &= p(\vec{x}) + p_x(\vec{x})dx_i + p_y(\vec{x})dy_i \\ &+ p_{xx}(\vec{x})\frac{{dx_i}^2}{2} + p_{xy}(\vec{x})dx_i dy_i + p_{yy}(\vec{x})\frac{{dy_i}^2}{2} + e_i, \end{aligned} \tag{4}$$

where e_i is the error in the Taylor's series expansion at the point $\vec{x_i}$ and $dx_i = x_i - x, dy_i = y_i - y$. The unknowns $p, p_x, p_y, p_{xx}, p_{xy}, p_{yy}$ at $\vec{x}$ in (4) are computed by minimizing the error e_i for $i = 1, 2, ..., m$. Since the Poisson equation (1) must be satisfied at $\vec{x}$ and boundary condition must do for boundary particles, we add the following equations in the above system of m equations:

$$f = p_{xx} + p_{yy}, \tag{5}$$

$$g = q_1 p + q_2 p_x n_x + q_2 p_y n_y, \tag{6}$$

where n_x, n_y are the x, y components of the unit normal vector $\vec{n}$ on boundary at $\vec{x}$. Hence, we have a total of $m + 2$ equations for 6 unknowns for boundary particles. Let us compute $p, p_x, p_y, p_{xx}, p_{xy}, p_{yy}$ at $\vec{x}$ by minimizing the quadratic form

$$J = \sum_{i=1}^{m} w_i e_i^2 + (\Delta p - f) + (g_1 p + g_2 \frac{\partial p}{\partial \vec{n}} - g)^2. \tag{7}$$

The minimization of J formally yields

$$\vec{a} = (M^T W M)^{-1}(M^T W)\vec{b}, \tag{8}$$

where

$$M = \begin{pmatrix} 1 & dx_1 & dy_1 & \frac{dx_1^2}{2} & dx_1 dy_1 & \frac{dy_1^2}{2} \\ 1 & dx_2 & dy_2 & \frac{dx_2^2}{2} & dx_2 dy_2 & \frac{dy_2^2}{2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & dx_m & dy_m & \frac{dx_m^2}{2} & dx_m dy_m & \frac{dy_m^2}{2} \\ 0 & 0 & 0 & 1 & 0 & 1 \\ q_1 & q_2 n_x & q_2 n_y & 0 & 0 & 0 \end{pmatrix},$$

$$W = diag[w_1, w_2, \cdots, w_m, 1, 1]^T, \quad \vec{a} = [p, p_x, p_y, p_{xx}, p_{xy}, p_{yy}]^T,$$

$$\vec{b} = [p_1, p_2, ..., p_m, f, g]^T.$$

The matrices M and W and the vector $\vec{b}$ given above are for boundary particles. For interior particles the last row of the matrices M and W and the last component of the vector $\vec{b}$ are omitted.

Our goal is to find the value of p, not its derivatives. Equating the first component of the vector $\vec{a}$ and the corresponding value of right hand side in (8), gives

$$p = \sum_{i=1}^{m} \beta_i p_i + r, \tag{9}$$

where r has no p_i components. If we do the above process for all particles in the domain, we get a large, unstructured linear system of equations

$$AP = R, \tag{10}$$

where

$$A = \begin{pmatrix} * & * & \cdots & * & 0 & \cdots & * & 0 \\ * & \cdots & * & * & 0 & \cdots & 0 & * \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & * & \cdots & 0 & * & * \end{pmatrix},$$

$P = (p_1, p_2, ..., p_N)^T, R = (r_1, r_2, ..., r_N)^T$.

3. $\mathcal{H}$-Matrix

The basic requirements to construct $\mathcal{H}$-matrix for the matrix A are

1. Cluster tree T_I on index set $I = \{1, 2, \ldots, N\}$
2. Admissible condition
3. Block cluster tree on $I \times I$.

In this section, let us look at each of them in detail.

The complexity and storage of an $\mathcal{H}$-matrix depend on the construction an appropriate hierarchy of index set I partitioning. Let I be an index set with $N = |\mathrm{I}|$ elements generated in the domain for discretization. For each index $\alpha \in \mathrm{I}$, let $x_\alpha \in \mathbb{R}^2$ be an associated point in the domain. Any non-empty subset $t \in \mathrm{I}$, is called as cluster and each cluster corresponding to an axis oriented box bounded domain $B_t \in \mathbb{R}^2$, where B_t is the smallest rectangle box $\prod_{j=1}^{2}[a_j, b_j]$ contains all the points x_α for each α in t .

We define the cluster tree T_I as follows:

Definition 3.1 (Cluster tree [28]). Let T_{I} be denote the set of all cluster of a tree $\tilde{T}_{\mathrm{I}}$. $\tilde{T}_{\mathrm{I}}$ is called a cluster tree for the index set I with parameter N_s if the following conditions hold:

1. The index set I is the root of T_{I}
2. Each node of the tree T_{I} is a subsect of the index set
3. if $t \in T_{\mathrm{I}}$ is a leaf, then the number of indices in t ($|t|$) is less than N_s.
4. if $t \in \mathrm{T}_I$ is not a leaf, then there are two unique non-empty clusters t_1,$t_2 \in T_{\mathrm{I}}$ with $t = t_1 \cup t_2$ and $t_1 \cap t_2 = \emptyset$. The clusters t_1, t_2 called sons of t.

The parameter N_s is called leaf parameter for the cluster tree T_I and it controls the smallest blocks in hierarchical matrices, i.e. the smallest subblock in the $\mathcal{H}$-matrix of the size $N_s \times N_s$. If the block $N_s \times N_s$ is too small, the computation of constructing $\mathcal{H}$-matrix is more costly than the exact assembling.

The cluster tree T_I is obtained by splitting the bounding boxes. The following recursive algorithm1 [28] creates cluster tree over I.

Algorithm 1 Cluster Tree Creation Algorithm

```
function Cluster Tree Creation($t$,$T_{\mathrm{I}}$)
if $|t| \leq N_s$ then
    return
else
    Create axis oriented box $B_t$ of minimal size with support of $t$ i.e. $\cup t \subseteq B_t$
    Split $B_t$ along longest edge into $B_{t_1}$,$B_{t_2}$
    Define sons $t_1$;$t_2$ of $t$ by
        $t_1 = \{i \in t | x_i \in B_{t_1}\}$, $t_2 = t \setminus t_1$.
    Add $t_1$ to $T_{\mathrm{I}}$ and call CreateClusterTree ($t_1, T_{\mathrm{I}}$)
    Add $t_2$ to $T_{\mathrm{I}}$ and call CreateClusterTree ($t_2, T_{\mathrm{I}}$)
end if
```

Once cluster tree T_{I} is created over the index I, the next step is to do block partitioning by using the cluster tree over the index set $I \times I$. A block cluster tree $T_{I\times I}$ is cluster tree over the index set $I \times I$. The root of the block cluster $T_{I\times I}$ corresponds to the matrix A. For any clusters $t, s \in T_{\mathrm{I}}$, the pair (t, s) form a block in the matrix A. The hierarchical partition of $I \times I$ or the matrix A depend on "Admissible Condition".

Definition 3.2 (Admissibility [7]). For a fixed parameter $\eta > 0$, the block (t, s) is admissible if

$$\min\{\operatorname{diam}(B_t), \operatorname{diam}(B_s)\} \leq \eta.\operatorname{dist}(B_t, B_s) \tag{11}$$

Here the parameter η controls the number of admissible blocks in the matrix A. For a partitioning $\mathbb{P} \subseteq T_{\mathrm{I}\times\mathrm{I}}$ of $I \times I$, we define the **farfield**

$$\mathbb{P}_{far} := \{(t, s) \in \mathbb{P} \mid (t, s) \text{ is admissible}\}$$

and the **nearfield**

$$\mathbb{P}_{near} := \mathbb{P} \setminus \mathbb{P}_{far} = \{(t, s) \in \mathbb{P} \mid (t, s) \text{ is inadmissible } or\ |t| < N_s\ or\ |s| < N_s\}$$

The main idea of the hierarchical partitioning $\mathbb{P}$ is as follows:

- For admissible blocks $(t, s) \in \mathbb{P}_{far}$, use approximate low rank matrix for the sub block $A|_{t\times s}$.
- For inadmissible blocks $(t, s) \in \mathbb{P}_{near}$, keep the sub-block $A|_{t\times s}$ exactly.

The block partitioning is done by using Algorithm 2 (see [7]).

Algorithm 2 Block Partitioning

```
function Create$\mathcal{H}$Partitioning($\mathbb{P}_{far}$, $\mathbb{P}_{near}$,$t$,$s$)
**if** $(t, s)$ admissible **then**
    add $(t, s)$ to $\mathbb{P}_{far}$
**else if** $sons(t) \neq \emptyset$ ; and $sons(s) \neq \emptyset$ **then**
    **for** $(t_1, s_1) \in \text{sons}(t) \times sons(s)$ **do**
        call Create$\mathcal{H}$Partitioning ($\mathbb{P}_{far}$,$\mathbb{P}_{near}$,$t_1$,$s_1$)
    **end for**
**else**
    add $(t, s)$ to $\mathbb{P}_{near}$
**end if**
```

Definition 3.3 (Rk-matrix [26]). A $n \times m$ matrix A of the form

$$A = VW^T, \qquad V \in R^{n\times m}, W \in R^{k\times n}$$

is called Rk-matrix.

Note that if $\text{rank}(A) = k$, then A can be written in Rk-matrix format.

Definition 3.4 ($\mathcal{H}$-matrix [14])**.** Let $\mathrm{T}_{\mathrm{I}\times\mathrm{I}}$ be a hierarchical partition over the index set $I \times I$ and k be a positive integer. Define the set of $\mathcal{H}$-matrices as $\mathcal{H}(\mathrm{T}_{I\times I}, k) = \{A \in \mathbb{R}^{|I|\times|I|} : \text{rank}(A|_{t\times s}) \leq k, \text{ for all admissible leaves } t \times s\}$.

Algorithm 3 (see [14]) represents the construction of $\mathcal{H}$-matrix.

Algorithm 3 Construction of $\mathcal{H}$-matrix

```
INPUT: Index set $I$, admissibility parameter $\eta$, leaf parameter $N_s$
% Algorithms for computing entries for low-rank blocks and inadmissible blocks.%
Construct the cluster tree $T_I$ from the index set $I$.
Construct the block cluster tree $T_{I\times I}$ from the cluster tree $T_I$.
Compute the entries for admissible and inadmissible matrix blocks.
OUTPUT: $\mathcal{H}$-matrix in $\mathcal{H}(T_{I\times I}, k)$.
```

3.1. $\mathcal{H}$-Matrix Arithmetic

We described the class of matrices called $\mathcal{H}$-matrices. Since this class has certain features, instead of doing normal matrix arithmetic such as matrix addition, matrix-matrix multiplication, matrix-vector multiplication, matrix inverse, Hackbusch et al. [26, 28] have modified matrix arithmetic corresponding to $\mathcal{H}$-matrices called $\mathcal{H}$-matrix arithmetic.

Matrix-vector multiplication of $\mathcal{H}$-matrix is given in Algorithm 4, see [26].

Another important $\mathcal{H}$-arithmetic is $\mathcal{H}$-matrix inversion. In order to compute A^{-1}, we have to decompose A. If the set of sons of $t \times s$ is non-empty, then decompose A as follows: $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$. Algorithm 5 represents the $\mathcal{H}$-matrix inversion. We have employed Algorithm 5 to compute the $\mathcal{H}$-inverse in [10].

4. Implementation of $\mathcal{H}$-Matrix on FPM

$\mathcal{H}$-matrix originally developed to improve the computational efficiency and storage of finite element method and boundary element method and also applied to many grid based methods. In this section, we are going to describe the

Algorithm 4 $\mathcal{H}$-matrix vector multiplication

```
function $\mathcal{H}$-MatrixVectorMultiplication$(A, t, s, x, y)$
if $(t, s)$ is admissible then
    call $Rk$-matrix$(A, V, W)$
    $y|_t := y|_t + V_{t\times s_1} W^T_{s\times s_1} x|_s$
else if sons$(t, s) \neq \emptyset$ then
    for $(t', s') \in$ sons$(t, s)$ do
        call HMatrixVectorMultiplication$(A, t', s', x, y)$
    end for
else
    $y|_t := y|_t + A|_{t\times s} x|_s$
end if
```

Algorithm 5 $\mathcal{H}$-matrix inversion

```
function Invert$(A^{-1}, t, s, A)$
if s thenons$(t \times s) = \emptyset$
    calculate $A^{-1}$ exactly
else
    Decompose $A$
    $Z = A_{22} - A_{21}(A_{11}^{-1} A_{12})$
    $(A^{-1})_{22} = Z^{-1}$
    $(A^{-1})_{11} = A_{11}^{-1} + (A_{11}^{-1} A_{12} Z^{-1} A_{21} A_{11}^{-1})$
    $(A^{-1})_{12} = -A_{11}^{-1} A_{12} Z^{-1}$
    $(A^{-1})_{12} = -A_{22}^{-1} A_{21} (A^{-1})_{11}$
end if
```

implementation of $\mathcal{H}$-matrix technique for FPM. The matrix, say FPM matrix obtained from (10) is a large sparse matrix. Therefore, we use preconditioned BiCGStab algorithm to solve the system $AP = R$ using $\mathcal{H}$-matrix arithmetic as presented in Algorithm 6.

In this paper, we have used the technique of row compressed SAMG format [20] to store the FPM matrix. This format stores row pointers, column indices, and non-zero values explicitly in the array `ptr`, `ind` and `data`. The number of non-zeros of the ith row is given by $ptr[i+1]-ptr[i]$. The row compressed SAMG representation of an arbitrary matrix A discussed below. In which, the indices of matrix A are stored as follows: $A_{ii} = val[ind[ptr[i]]]$ $A_{ij} = val[ind[]ptr[i]]$.

Algorithm 6 $\mathcal{H}$-BICGSTAB Algorithm

```
function $\mathcal{H}$-BICGSTAB($A, P, R, tree, tolerance$)
$U = \mathcal{H}$-MatrixVectorMultiplication($A, I, I, P, Y$)
$U = R - U$
$\hat{U} = U, G = J = Q = 0, \alpha = \omega_0 = \rho_0 = 1$
while $(\sum U.U - \epsilon > 0)$ : do
    $\rho = \sum U.\hat{U}$
    $\beta = \rho/\rho_0 * \alpha/\omega_0$
    $G = U + \beta(U - \omega_0 J)$
    $J = \mathcal{H}$-MatrixVectorMultiplication($A, I, I, G, Y$)
    $\alpha = \rho / \sum \hat{U}.J$
    $U = U - \alpha J$
    $Q = \mathcal{H}$-MatrixVectorMultiplication($A, I, I, U, Y$)
    $\omega = \frac{\sum Q.U}{\sum Q.Q}$
    $P = P + \alpha G + \omega U$
    $U = U - \omega Q$
    $\rho_0 = \rho, \omega_0 = \omega$
end while
```

5. Numerical Results

We consider the following pressure Poisson problem

$$\Delta p = -2 \quad \text{in} \quad \Omega = (0,1) \times (0,1) \tag{12}$$

with mixed boundary conditions:

$$\begin{aligned} p = 0 \qquad & \text{on} \quad x = 1,\ y = 1 \\ \frac{\partial p}{\partial \vec{n}} = 0 \quad & \text{on} \quad x = 0,\ y = 0. \end{aligned}$$

Before applying the FPM method, we randomly generate global points $x_j,\ j = 1, 2, ..., n$ on boundary of unit square. Using the smoothing length h the neighborhood list $x_i,\ i = 1, 2, ..., N_j$ for each x_j is generated. In our test, we choose smoothing lengths $h = 0.1,\ 0.01,\ 0.001$. For the smoothing lengths $h = 0.1,\ 0.01,\ 0.001$, $N = 629, 6083, 52527$ points generated in the domain Ω respectively. The Poisson problem (12) is solved with the help of the points $N = 629, 6083, 52527$ in the domain respectively using Finite Point set method. A large sparse linear system $AP = R$ is obtained, where A is big

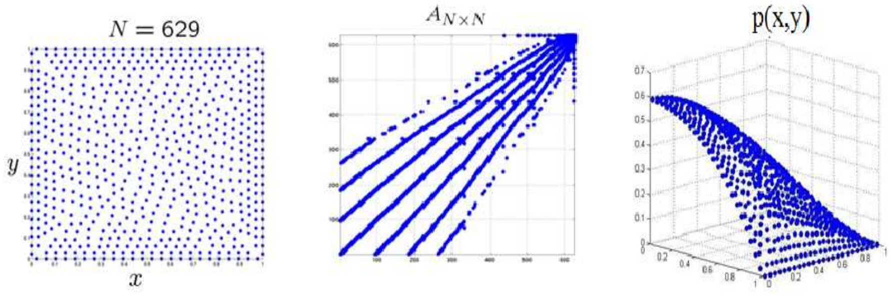
$$A = \begin{bmatrix} 2 & -2 & 0 \\ -1 & 2 & 3 \\ 0 & -1 & 2 \end{bmatrix}$$

$$\texttt{ptr} = [\ 1 \quad 3 \quad 6 \quad 8\ ]$$

$$\texttt{ind} = [\ 1 \quad 2 \quad 2 \quad 1 \quad 3 \quad 3 \quad 2\ ]$$

$$\texttt{data} = [\ 2 \quad -2 \quad 2 \quad -1 \quad 3 \quad 2 \quad -1\ ]$$

SAMG representation of A



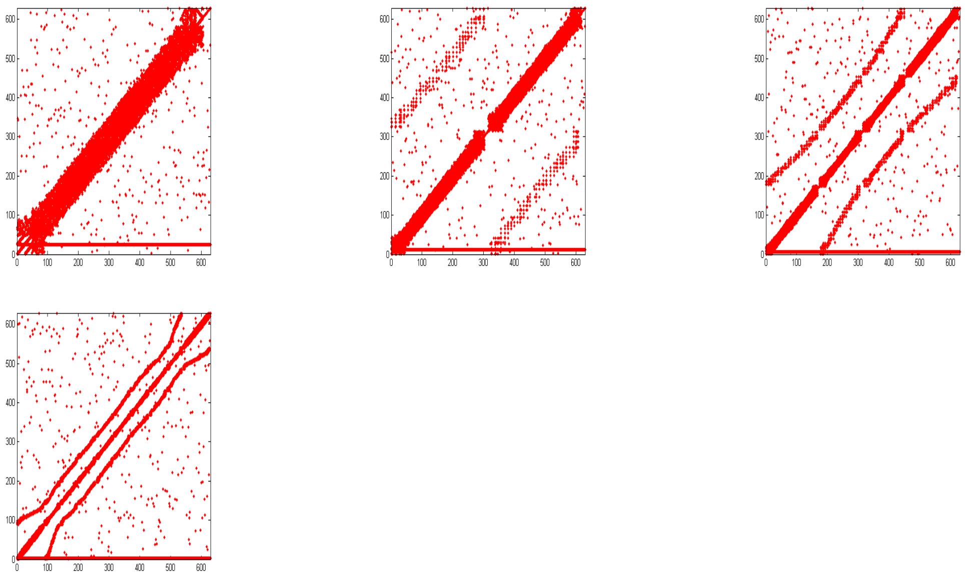
(a) Particle distribution on Ω (b) FPM Matrix (c) Solution of the problem

Figure 1: Solution of Poisson problem with mixed boundary conditions for the smoothing length $h = 0.1$

unstructured sparse matrix of order 629, 6083, 52527 for the smoothing lengths $h = 0.1,\ 0.01,\ 0.001$ respectively. Figure 1a shows the point generation on the domain Ω, Figure 1b denotes its corresponding matrix obtained after applying FPM method and Figure 1c shows the solution of the Poisson problem (12) with the given mixed boundary conditions for the smoothing length $h = 0.1$. In this sparse matrix A, neighboring particle index gives the entries of the matrix components and the distribution of the nonzero entries in A based on the mesh in the domain Ω.

The obtained sparse matrices are re-indexed in such a way that more non zero elements should be across and near the anti-diagonal by taking different grid sizes of the domain Ω are seen in Figure 2.

The obtained systems are stored in SAMG format and solved by $\mathcal{H}$-matrix technique with preconditioned BiCGStab as a linear solver with $\mathcal{H}$-matrix inverse as a preconditioner Algorithm 5 with the help of $\mathcal{H}$-matrix arithmetic. The experiments in this section were carried out on a Leo cluster machine.



(a) clustering for l=1 (b)clustering for l=0.5 (c) clustering for l=0.25 (d)clustering for l=0.10

Figure 2: Clustering for smoothing lengths l=1, 0.5, 0.25, 0.1

5.1. Computation Time

The time required for building $\mathcal{H}$-matrix representation for the Poisson problem for $N = 52527$ is given in Table 1. It is observed that the complexity is increased by a factor of more than 2 as the value of N_s decreases.

The time required for matrix vector multiplications of order $N = 52527$ is given in Table 2. For a fixed N_s and $\eta = 1$, 500, 1000, 2000, the time required for building $\mathcal{H}$-matrix is approximately same because the structure of $\mathcal{H}$-matrices are same. From Table 2 observes that the complexity grows slowly as N_s decreases and decreases slowly as η increases.

Figure 3 shows the $\mathcal{H}$-matrix structure of the matrix of order $N = 52527$ for different minimal admissible sizes $N_s = 1000, 2500, 5000$ and for a fixed admissible parameter $\eta = 1$. It is observed that the numbers of admissible blocks increases when N_s decreases. It implies the computation complexity of matrix vector multiplication increases if N_s decreases.

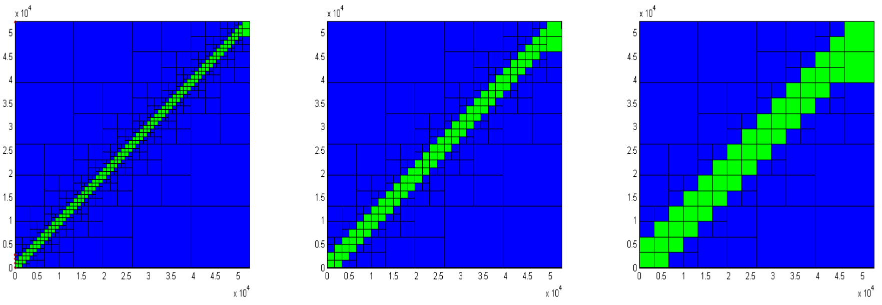
The time required for solving the sparse linear system by $\mathcal{H}$-Matrix technique with preconditioned BiCGStab solver for $N = 629, 6083, 52527$ are given

	N_s				
$N = 52527$	17000	5000	2500	1000	500
$\eta = 1$	0.282	1.92	4.48	12.73	24.28
$\eta = 500$	0.280	1.89	4.76	12.85	24.37
$\eta = 1000$	0.291	1.91	4.87	13.07	26.52
$\eta = 2000$	0.287	1.89	5.69	13.93	27.12

Table 1: Time [sec] required for building $\mathcal{H}$-matrix

$N = 52527$	$N_s = 5000$	$N_s = 2500$	$N_s = 1000$	$N_s = 500$	$N_s = 100$
$\eta = 1$	1.66	1.70	1.84	2.14	2.46
$\eta = 500$	1.64	1.65	1.84	2.08	2.37
$\eta = 1000$	1.61	1.64	1.82	2.03	2.35
$\eta = 2000$	1.57	1.61	1.79	2.00	2.31

Table 2: Time [sec] required for matrix vector multiplication



$N_s = 1000, 2500, 5000$ and $\eta = 1$

Figure 3: $\mathcal{H}$-Matrix structure for $N = 52527$,$N_s = 1000, 2500, 5000$, $\eta = 1$

in Tables 3, 4 and 5, respectively.

For $N = 629$, the minimum time required is 0.05 sec for $N_s = 60$ and $\eta = 0.3$, which is negligible. For $N = 6083$, the minimum time required is 2.2 sec for $N_s = 500$ and $\eta = 1$. For $N = 52527$, the minimum time required is 46.13 sec for $N_s = 17000$ and $\eta = 1$. The time required for $C_{leaf} = 17000$, $N_s = 5000$ and $C_{leaf} = 17000$, $N_s = 2500$ is almost linear since the structure of the corresponding $\mathcal{H}$-matrices are same respectively in each cases.

	N_s				
$N = 629$	100	80	60	40	20
$\eta = 0.1$	0.23	0.2	0.07	0.10	0.12
$\eta = 0.3$	0.20	0.15	0.05	0.08	0.15
$\eta = 0.5$	0.25	0.17	0.08	0.10	0.18
$\eta = 1.0$	0.30	0.20	0.10	0.12	0.20

Table 3: Time [sec] required for solving the Poisson problem

	N_s					
$N = 6083$	2500	1000	500	250	100	20
$\eta = 1$	2.30	2.30	2.20	2.42	4.28	5.82
$\eta = 20$	2.32	2.30	2.26	3.16	4.29	5.94
$\eta = 50$	2.36	2.31	2.28	3.20	4.31	6.02
$\eta = 100$	2.33	2.34	2.29	3.30	4.60	6.25

Table 4: Time [sec] required for solving the Poisson problem

	N_s						
$N = 52527$	17000	5000	2500	1000	500	100	20
$\eta = 1$	46.13	47.31	70.93	102.04	145.23	208.06	281.14
$\eta = 500$	46.13	47.57	71.19	102.57	146.08	209.43	282.27
$\eta = 1000$	46.16	47.47	71.21	103.24	147.25	210.31	283.42
$\eta = 2000$	46.15	47.82	69.37	104.47	148.48	211.37	284.16

Table 5: Time [sec] required for solving the Poisson problem

Direct solver	0.74 sec
$\mathcal{H}$-Matrix technique	0.05 sec

Table 6: Comparison for $N_s = 60$, $\eta = 0.3$ & N=629

Direct solver	3.2 sec
$\mathcal{H}$-Matrix technique	2.2 sec

Table 7: Comparison for $N_s = 500$, $\eta = 1$ & N=6083

Direct solver	92.0 sec
$\mathcal{H}$-Matrix technique	46.13 sec

Table 8: Comparison for $N_s = 17000$, $\eta = 1$ & N=52527

For $N = 629, 6083$ and 52527, it is observed that the complexity for solving the sparse linear system by $\mathcal{H}$-Matrix technique with BiCGStab solver is almost linear.

In Tables 6, 7 and Table 8, the CPU times are shown for solving the sparse linear systems for $N = 629$, $N = 6083$ and $N = 50527$. The computation time of $\mathcal{H}$-matrix approach is compared with the computation time of direct solver. We observed that CPU time of $\mathcal{H}$-Matrix technique is less than the CPU time of direct solver.

6. Conclusion

In this paper, we have solved pressure Poisson equation with mixed boundary condition using grid free method. A large sparse linear system has been obtained after the spatial discretization of the domain using FPM. FPM matrix are stored in row compressed SAMG format. By varying the values of N_l and η, we have obtained different structures of $\mathcal{H}$-matrix for FPM matrix. Using preconditioned BiCGStab on $\mathcal{H}$-matrix, we have solved the required linear system. Finally, we have showed that accessing a matrix using $\mathcal{H}$-matrix techniques, to solve a large sparse linear system, reduces the computation time.

References

[1] A. Chorin, Numerical solution of the Navier-Stokes equations, *J. Math. Comp.*, **22** (1968), 745-762.

[2] B. Steffen, Approximate of integral operators by $\mathcal{H}^2-$matrices with adaptive bases, *Computing*, **73**, No 3 (2005), 249-271.

[3] B. Steffen, Approximation of solution operators of elliptic partial differential equations by $\mathcal{H}$- and $\mathcal{H}^2$-matrices, *Numerische Mathematik*, **115** (2010), 165-193.

[4] B. Steffen, Data-sparse approximation of non-local operators by $\mathcal{H}^2-$matrices, *Linear Algebra and its Applications*, **422** (2007), 380-403.

[5] B. Steffen, $\mathcal{H}^2-$matrix arithimetics in linear complexity, *Computing*, **77**, No 1 (2006), 1-28.

[6] B. Steffen and L. Grasedyck, Hybrid cross approximate of integral operators, *Numer. Math.*, **101**, No 2 (2005), 221-249.

[7] B. Steffen, L. Grasedyck and W. Hackbusch, Introduction to hierarchical matrices with applications, *Engineering Analysis with Boundary Elements*, **27** (2003), 405-422.

[8] D.M. Young, *Iterative solution of Large Linear Systems*, Academic Press, New York, (1971).

[9] G. Anne, *Iterative Methods for Solving Linear Systems*, SIAM Publ., Philadelphia, PA, 1997.

[10] I.P. Gavrilyuk, W. Hackbusch and B. N. Khoromskij, $\mathcal{H}$-matrix approximation for the operator exponential with applications, *Numerische Mathematik*, **92** (2003), 83-111.

[11] J. Koch, W. Hackbusch and K. Sundmacher, $\mathcal{H}$-matrix methods for linear and quasi-linear integral operators appearing in population balances, *Computers and Chemical Engineering*, **31** (2007), 745-759.

[12] K. Kriemann, Parallel $\mathcal{H}$ arithmetics on shared memory systems, *Computing*, **74** (2005), 273-297.

[13] L. Grasedyck, R. Kriemann and S.L. Borne, Domain decomposition based $\mathcal{H}$-LU preconditioning, *Numerische Mathematik*, **112** (2009), 565-600.

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

[15] M. Bebendorf, Hierarchical *LU* decompostion based preconditions for BEM, *Computing*, **74** (2005), 225-247.

[16] M. Bebendorf and W. Hachbusch, Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients, *Numerische Mathematik*, **95** (2003), 1-28.

[17] M. Lintner, The eigenvalue problem for the 2D Laplacian in $\mathcal{H}$-matrix arithmetic and applications to the heat and wave equation, *Computing*, **72** (2004), 293-323.

[18] M. Panchatcharam and S. Sundar, Finite pointset method for 2D dam-break problem with GPU-acceleration, *International Journal of Applied Mathematics*, **25**, No 4 (2012), 547-577.

[19] S.A. Goreinov, E.E. Tyrtyshnikov, and N.L. Zamarashkin, A theory of pseudoskeleton appromimations, *Linear Algebra Appl.*, **261** (1997), 1-22.

[20] S. Klaus, *Structure and File Format Specification SAMG*, A short document, February (2005).

[21] S. Oliveira and F. Yang, An algebraic approach for $\mathcal{H}-$matrix preconditioners, *Computing*, **80** (2007), 169-188.

[22] S. Tiwari, C. Drumm, J. Kuhnert and H. J. Bart, Finite pointset method for simulation of the liquid-liquid flow field in an extractor, *Computers and Chemical Engineering*, **32** (2008), 2946-2957.

[23] S. Tiwari, and J. Kuhnert, A meshfree method for incompressible fluid flows with incorporated surface tension, *Revue aurope'enne des elements finis*, Volume **11** (2002).

[24] S. Tiwari, J. Kuhnert, Finite pointset method based on the projection method for simulations of the incompressible Navier-Stokes equations, Meshfree methods for partial differential equations, *Lect. Notes on Comput. Sci. Eng.* (Springer, Berlin) **26** (2003), 373-387.

[25] S. Tiwari and J. Kuhnert, Modeling of two-phase flows with surface tension by finite pointset method(FPM), *Journal of Computational and Applied Mathematics*, **203** (2007), 376-386.

[26] W. Hackbusch, A sparse matrix arithmetic based on $\mathcal{H}$-Matrix Arithmetic. Part I: Introduction to $\mathcal{H}$-matrices, *Computing*, **62** (1999), 89-108.

[27] W. Hackbusch and B.N. Khoromskij, A sparse $\mathcal{H}$-Matrix Arithmetic general complexity estimates, *Journal of Computational and Applied Mathematics*, **125** (2000), 479-501.

[28] W. Hackbusch and B.N. Khoromskij, A sparse $\mathcal{H}$-Matrix Arithmetic Part II: Application to Multi-Dimensional problems, *Computing*, **64** (2000), 21-47.

[29] W. Hackbush, and B.N. Khoromskij, $\mathcal{H}$-matrix approximation on graded meshes. In: John R. Whiteman (Ed.), *The Mathematics of Finite Elements and Applications,* **10**, Elsevier (2000), 307-316.

[30] W. Hackbusch, B.N. Khoromskij and R. Kriemann, Hierarchical matrices based on weak admissibility criterion, *Computing*, **73** (2004), 207-243.

[31] W. Hackbusch, L. Grasedyck and S. Börm, An introduction to hierarchical matrices, *Mathematica Bohemica*, **127**, No 2 (2002), 229-241.

[32] W. Hackbusch and S. Börm, Data-spares approximation by adaptive $\mathcal{H}^2$-matrices, *Computing*, **69** (2002), 1-35.

[33] W. Hackbusch and S. Börm, $\mathcal{H}^2$-matrix approximation of integral operators by interpolation, *Applied Numerical Mathematics*, **43** (2002), 129-143.