Malaysian Journal of Mathematical Sciences 10(3): 369–382 (2016)

MALAYSIAN JOURNAL OF MATHEMATICAL SCIENCES

Journal homepage: http://einspem.upm.edu.my/journal

An Adaptive Hierarchical Matrix on Point Iterative Poisson Solver

Nik Amir Syafiq[1], Mohamed Othman *[1,2], and Norazak Senu[1,2]

[1] *Institute for Mathematical Research, Universiti Putra Malaysia, Malaysia*

[2] *Department of Communication Technology and Network , Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Malaysia*

E-mail: mothman@upm.edu.my

* *Corresponding author*

ABSTRACT

In this paper, an adaptive hierarchical matrix ($\mathscr{H}$-matrix) points iterative method based solution was proposed to solve two-dimensional Poisson problem with Dirichlet boundary condition. The finite difference approximation was used to discretize the problem, which led to a system of linear equation. Two types of admissibility conditions, standard and weak, produces two different $\mathscr{H}$-matrix structures, $\mathscr{H}_S$- and $\mathscr{H}_W$- respectively. The adaption of the $\mathscr{H}$-matrices to a linear system leads to the saving of memory utilization. An experiment was conducted which compares the proposed $\mathscr{H}_W$-matrix with the benchmarked $\mathscr{H}_S$-matrix. The results showed the superiority of the proposed method when comparing both $\mathscr{H}$-matrix structures [1].

Keywords: Adaptive Hierarchical Matrix, Point Iterative Solver, Poisson Equation, Finite Difference Approximation.

[1]Abstract of the paper was presented in the 8th International Congress on Industrial and Applied Mathematics, ICIAM 2015, Syafiq et al. (2015)

1. Introduction

Throughout the years, iterative methods have been used extensively in solving any real world problems related to real time natural phenomena. Many researchers adopted this method to obtain high accuracy solutions with fast convergence rate. The only downside of this method is that it requires a large-scale memory size and long CPU time in calculating huge number of grid size as mentioned by Sakurai et al. (2002).

In 1998, Hackbush et al. introduced an $\mathscr{H}$-matrix approach which acted as an efficient treatment to dense matrices as it was stored in a special data-sparse way. This was done by carefuly partitioning the matrix in accordance to an admissibility condition. There were two known admissibility conditions; the weak admissibility and the standard admissibility conditions. During both the partitioning processes, some sub-matrices would be converted into low-rank approximations. This treatment eventually reduced the memory utilization.

The construction of $\mathscr{H}$-matrix was presented by Hackbusch and his colleagues [Hackbusch (1999), Hackbusch and Khoromskij (2000) and Hackbusch et al. (2004)]. Consequently, many definitions and illustrations of $\mathscr{H}$-matrices were developed and explained by many researchers [Grasedyck et al. (2008) and Izadi (2012)]. Moreover, many researchers [Benner and Mach (2010), Börm et al. (2003a), Hackbusch (1999), Hackbusch and Khoromskij (2000), Hackbusch et al. (2004) and Wan et al. (2011)] had studied extensively the properties of $\mathscr{H}$-matrices.

In the application aspect, Engquist and Ying (2011) constructed a tridiagonal block matrix where $\mathscr{H}$-matrix was applied to each subblocks. In addition, an algorithm to construct an $\mathscr{H}$-matrix approximation was developed by Lin et al. (2011). As noted by many researchers, the usage of $\mathscr{H}$-matrix in solving any real world problems would reduce memory utilization as compared to the absence of $\mathscr{H}$-matrix.

Most researches in $\mathscr{H}$-matrix mainly partitioned their matrix using the standard admissibility condition which is defined as:

Definition 1.1. ***[Börm et al. (2003b)].*** *Let $\eta > 0$ be a fixed parameter while τ and σ be two index sets. A block $b = \tau \times \sigma$ is said to satisfy the standard admissibility condition (or $\eta-$admissible) if*

$$Adm_\eta(b) = true :\Longleftrightarrow \min(diam(\Omega_\tau), diam(\Omega_\sigma)) \leq \eta\ dist(\Omega_\tau, \Omega_\sigma), \quad (1)$$

where Ω_τ and Ω_σ are a union of the supports of the respective basis function

φ_i, *namely*

$$\Omega_\tau := \bigcup_{i\in\tau} supp(\varphi_i), \quad \Omega_\sigma := \bigcup_{i\in\sigma} supp(\varphi_i).$$

From this definition, the term “diam" and “dist" are denoted as the Euclidean diameter and distance of Ω_τ and Ω_σ that are defined as follows:

$$diam(\Omega_\tau) := \max_{x_i,x_j\in\Omega_\tau} ||x_i - x_j||,$$
$$dist(\Omega_\tau,\Omega_\sigma) := \min_{x_i\in\Omega_\tau,x_j\in\Omega_\sigma} ||x_i - x_j||.$$

However, this partitioning produced a large amount of low-rank sub-matrices as shown in Figure 1.

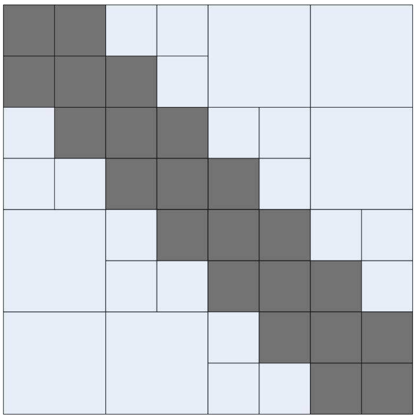

Figure 1: $\mathscr{H}$-matrix structure with standard admissibility condition. The darkened colored blocks consists of the full matrix while the other block becomes sub-matrices of low-rank approximation.

This could affect the accuracy of the $\mathscr{H}$-matrix. Thus, in this paper, the proposed method utilized an $\mathscr{H}$-matrix structure with less low-rank sub-matrices that could produce a more accurate $\mathscr{H}$-matrix.

This paper also discusses the problem derivation and discretization, proposed method, performance evaluation, conclusion and future works.

2. Problem Derivation and Discretization

Let's consider the two-dimensional Poisson problem that can be represented mathematically as,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y), \quad (x,y) \in \Omega, \tag{2}$$

with Dirichlet boundary conditions and satisfying the exact solution $u(x,y) = e(x,y)$, $(x,y) \in \partial\Omega$.

Let's take Eq. (2) as the model of the problem on the rectangular grid Ω with grid spacing h in both directions and $x_i = x_0 + ih$, $y_j = y_0 + jh$, for all $i,j = 0,1,\dots,n,n+1$ are used. Since Eq. (2) is solved by using the finite difference approximation, it will lead to the standard five-point method, which has been used by Othman and Abdullah (2000), as follows,

$$v_{i+1,j} + v_{i-1,j} + v_{i,j+1} + v_{i,j-1} - 4v_{i,j} = h^2 f_{i,j}, \tag{3}$$

where $v_{i,j}$ is an approximation to the exact solution $u(x_i,y_j)$ at the grid point (x_i,y_j) and $f_{i,j} = f(x_i,y_j)$. It can be seen that the application of Eq. (3) to each of the internal mesh points will result in a large and sparse linear system as follows,

$$Av = f, \tag{4}$$

where A is a square nonsingular matrix while v and f are denoted by the following column vectors containing all unknowns as,

$$v = (v_{1,1}, v_{2,1}, \dots, v_{n,1}, v_{1,2}, v_{2,2} \dots, v_{n,n-1}, v_{1,n}, v_{2,n}, \dots, v_{n,n})^T;$$
$$f = (f_{1,1}, f_{2,1}, \dots, f_{n,1}, f_{1,2}, f_{2,2} \dots, f_{n,n-1}, f_{1,n}, f_{2,n}, \dots, f_{n,n})^T.$$

Let P_m be defined as the unknowns in the m^{th} row

$$P_m = (P_{1,m}, \dots, P_{n,m})^T.$$

This can help introduce

$$v_m = (v_{1,m}, v_{2,m}, \dots, v_{n,m})^T \quad \text{and} \quad f_m = (f_{1,m}, f_{2,m}, \dots, f_{n,m})^T,$$

which yields

$$v = (v_1^T, v_2^T, \dots, v_n^T)^T \quad \text{and} \quad f = (f_1^T, f_2^T, \dots, f_n^T)^T.$$

Thus, Eq. (4) can be converted into $v = A^{-1}f = B$ where A^{-1} exists and $v = B \in \mathbb{R}^{n\times n}$.

3. $\mathscr{H}$-matrix Adaptation on Five Points Iterative Method

Let B be a matrix of dimension $n \times n$ with $n = 2^{L_M}$ for an integer L_M and let $\tilde{B}$ be the $\mathscr{H}$-matrix of B. Denote the set for all indices as $I_{0;1}$ where the former subscript indicates the level of partitioning and the latter is the index for blocks in each level. For simplicity, the weak admissibility condition will be used to partition the matrix which is defined as follows:

Definition 3.1. ***[Izadi (2012)].*** *The block $b = \tau \times \sigma$ is called W-admissible if it holds*

$$Adm_W(b) = true :\Longleftrightarrow b \text{ is leaf or } \tau \neq \sigma$$

where τ and σ are a set of indexes from i and j, respectively.

After the matrix has been partitioned, all of the admissible blocks will be converted to low-rank approximations by using singular value decomposition (SVD). The rank r is a set according to size of the partitioned block. Thus, each block will have a predefined rank.

On the first level of partitioning, matrix

$$\tilde{B} = \begin{pmatrix} B_{1;11} & B_{1;12} \\ B_{1;21} & B_{1;22} \end{pmatrix}$$

is obtained. According to the definition above, sub-matrices $B_{1;12}$ and $B_{1;21}$ will be the admissible blocks. By applying the SVD, it will produce their low-rank approximations.

On the second level of partitioning, the non-admissible sub-matrices from the previous level will be partitioned again thus producing the following matrix:

$$\tilde{B} = \begin{pmatrix} \begin{matrix} B_{2;11} & B_{2;12} \\ B_{2;21} & B_{2;22} \end{matrix} & B_{1;12} \\ B_{1;21} & \begin{matrix} B_{2;33} & B_{2;34} \\ B_{2;43} & B_{2;44} \end{matrix} \end{pmatrix}$$

Sub-matrices $B_{2;12}$, $B_{2;21}$, $B_{2;34}$ and $B_{2;43}$ are the admissible blocks as stated from the definition above. The low-ranks approximation of these blocks is done by using SVD.

For the third level of partitioning, sub-matrices that are non-admissible from the previous level will be partitioned again which will produce the following matrix:

$$\tilde{B} = \begin{pmatrix} B_{3;11} & B_{3;12} & \multicolumn{2}{c}{\multirow{2}{*}{$B_{2;12}$}} & & & & \\ B_{3;21} & B_{3;22} & & & \multicolumn{4}{c}{B_{1;12}} \\ \multicolumn{2}{c}{\multirow{2}{*}{$B_{2;21}$}} & B_{3;33} & B_{3;34} & & & & \\ & & B_{3;43} & B_{3;44} & & & & \\ & & & & B_{3;55} & B_{3;56} & \multicolumn{2}{c}{\multirow{2}{*}{$B_{2;34}$}} \\ \multicolumn{4}{c}{B_{1;21}} & B_{3;65} & B_{3;66} & & \\ & & & & \multicolumn{2}{c}{\multirow{2}{*}{$B_{2;43}$}} & B_{3;77} & B_{3;78} \\ & & & & & & B_{3;87} & B_{3;88} \end{pmatrix}$$

According to the definition above, submatrices $B_{3;12}$, $B_{3;21}$, $B_{3;34}$, $B_{3;43}$, $B_{3;56}$, $B_{3;65}$, $B_{3;78}$, $B_{3;87}$ will be the admissible blocks. Similar to the previous levels, by using SVD, low-rank approximations will be produced.

This partitioning will be continued until to the smallest sub-block size, which is $\mathbb{R}^{4\times4}$. This type of partitioning will reduce the memory of the matrix and to ensure that B is approximately the same with $\tilde{B}$ ($B \approx \tilde{B}$). Figure 2 shows the $\mathscr{H}$-matrix representation when (a) $n = 16$, (b) $n = 32$, (c) $n = 64$ and (d) $n = 128$. The diagonal darkened colored blocks consists of the full matrix while the other block becomes sub-matrices of low-rank approximation. The ranks for each of these blocks are shown accordingly.

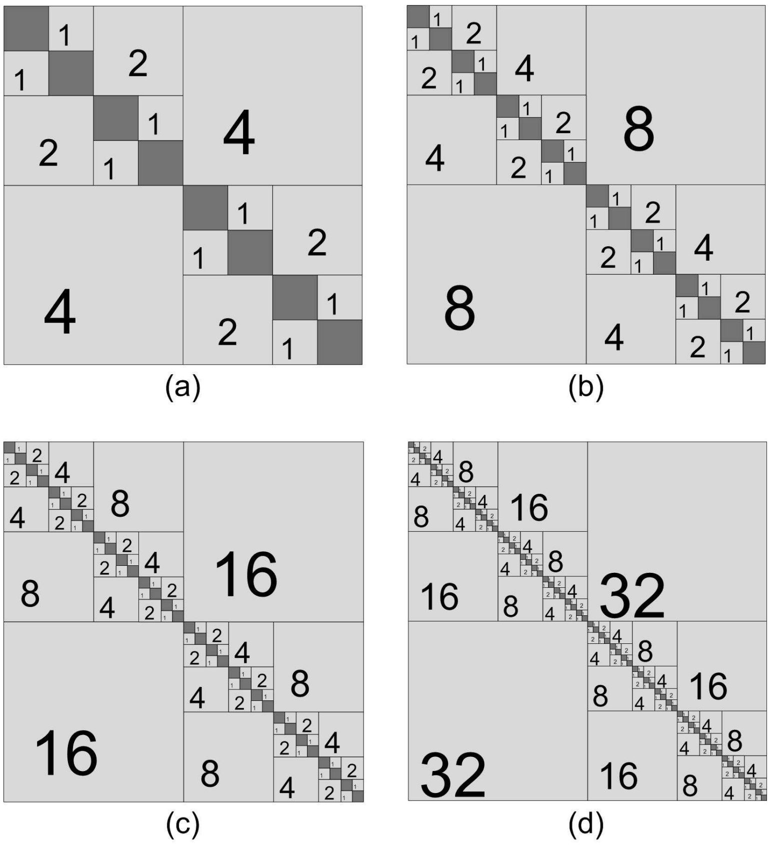

Figure 2: $\mathscr{H}$-matrix representation when (a) $n = 16$, (b) $n = 32$, (c) $n = 64$ and (d) $n = 128$. The numbers in each block shows their designated ranks..

To construct the $\mathscr{H}$-matrix, the following recursive algorithm is proposed:

Algorithm 1 hmconstruct(n, A)

```
Require: size $n$, matrix $A \in \mathbb{R}^{n\times n}$
Ensure: $\mathscr{H}$-matrix $A$
 1: Initialize: $r = (\frac{n}{4}) - 1$ ;
 2: if $n > 8$ then
 3:     $A_{1\ldots\frac{n}{2},1\ldots\frac{n}{2}} = \text{hmconstruct}((\frac{n}{2}), A_{1\ldots\frac{n}{2},1\ldots\frac{n}{2}})$ ;
 4:     $A_{(\frac{n}{2})+1\ldots n,(\frac{n}{2})+1\ldots n} = \text{hmconstruct}((\frac{n}{2}), A_{(\frac{n}{2})+1\ldots n,(\frac{n}{2})+1\ldots n})$ ;
 5: end if
 6: % $[U,S,V] = \text{svd}(X)$ produces a diagonal matrix $S$ of the same dimension
    as $X$, with nonnegative diagonal elements in decreasing order, and unitary
    matrices $U$ and $V$ so that $X = U * S * V'$.
 7: $(U,S,V) = \text{svd}(A_{1\ldots\frac{n}{2},(\frac{n}{2})+1\ldots n})$ ;          ▷ MATLAB built-in function
 8: $A_{1\ldots\frac{n}{2},(\frac{n}{2})+1\ldots n} = U_{1\ldots\frac{n}{2},1\ldots r} * S_{1\ldots r,1\ldots r} * V'_{1\ldots\frac{n}{2},1\ldots r}$ ;
 9: $(U,S,V) = \text{svd}(A_{(\frac{n}{2})+1\ldots n,1\ldots\frac{n}{2}})$ ;
10: $A_{(\frac{n}{2})+1\ldots n,1\ldots\frac{n}{2}} = U_{1\ldots\frac{n}{2},1\ldots r} * S_{1\ldots r,1\ldots r} * V'_{1\ldots\frac{n}{2},1\ldots r}$ ;
```

This recursive algorithm assists in partitioning the matrix and constructing the low-rank matrices. The 'SVD' call is a MATLAB built-in function which computes the singular matrix value decomposition [MATLAB® (2015)]. The dimensions of the resulting decomposed matrices will be determined by the size of the partition matrices and the preset ranks. This ensures that less memory is used.

In this research, the iterative part of Poisson solver is combined with the $\mathscr{H}$-matrix structure. In other words, the iterative Poisson program consists of three main parts. The first part is the iterative part. This is where the standard five-points method is applied. The second part is the $\mathscr{H}$-matrix construction. Here, the program will go through Algorithm 1 to get its $\mathscr{H}$-matrix approximation ($v_{new} = \mathscr{H}(v_{old})$). The final part is the convergence test. The program is checked for convergence test with a certain number of error tolerance. If it does not converge, the program will continue until it's converges and exits. In summary, the proposed iterative Poisson program can be written as in the following algorithm:

Algorithm 2 Proposed Algorithm

```
Require: size $m$
Ensure: average absolute error, time taken to execute program
Initialize: $u, v, e, f \in \mathbb{R}^{(m+2)\times(m+2)}$; $epsilon = 1.0e-10$ ;
start timer ;
Let localf = 0 ;
while localf $\neq 1$ do
    localf = 1
    $u = \text{std51h}(m, u, f)$ ;                        ▷ Standard five-point method
    $u_{2\ldots m+1,2\ldots m+1} = \text{hmconstruct}(m, u_{2\ldots m+1,2\ldots m+1})$   ▷ Algorithm 1
    for $i = 2, 3, \ldots, m+1$ do
        for $j = 2, 3, \ldots, m+1$ do
            if $\text{abs}(u_{i,j} - v_{i,j}) > epsilon$ then
                localf = 0 ;                             ▷ Check for convergence
            end if
        end for
    end for
    $v = u$ ;                                 ▷ Swap old values becomes new values
end while
end timer ;
display **average absolute error** and **time taken to execute program**;
```

4. Convergence Analysis

This section analyzes the convergence of an iterative method with $\mathscr{H}$-matrix adaptation when finding the solution to Eq. (4). Here, the v in the equation will be adapted with $\mathscr{H}$-matrix. From the previous section, it can be seen that by constructing the $\mathscr{H}$-matrix of v, denoted as $v_{\mathscr{H}}$, v will be approximately the same with $v_{\mathscr{H}}$ ($v \approx v_{\mathscr{H}}$). Thus Eq. (4) can be rewritten as

$$Av_{\mathscr{H}} = f, \tag{5}$$

From Eq. (5), by refering to Young (1971), Varga (2000), and Börm et al. (2003b), the iterative method will take the form of

$$Mv_{\mathscr{H}}^{k+1} = Nv_{\mathscr{H}}^{k} + f, \tag{6}$$

where $A = M - N$, M and N are square matrices, $v_{\mathscr{H}}^{k}$ is the current approximation and $v_{\mathscr{H}}^{k+1}$ is the approximation of the next iteration. Eq. (6) can be rewritten as

$$v_{\mathscr{H}}^{k+1} = M^{-1}Nv_{\mathscr{H}}^{k} + M^{-1}f, \tag{7}$$

which can be simplified into

$$v_{\mathscr{H}}^{k+1} = Cv_{\mathscr{H}}^{k} + \widetilde{f}, \tag{8}$$

where $C = M^{-1}N$ is often referred to as the iteration matrix and $\widetilde{f} = M^{-1}f$. If the current approximation $v_{\mathscr{H}}^{k}$ is, in fact, the exact solution $v_{\mathscr{H}}$, then the iterative method should produce a next iteration $v_{\mathscr{H}}^{k+1}$ that is also the exact solution. That is, it should be true that

$$v_{\mathscr{H}} = Cv_{\mathscr{H}} + \widetilde{f}. \tag{9}$$

To understand the convergence properties of the iterative method from Eq. (8), by subtracting it from Eq. (9), it will yield

$$v_{\mathscr{H}} - v_{\mathscr{H}}^{k+1} = C(v_{\mathscr{H}} - v_{\mathscr{H}}^{k}). \tag{10}$$

That is, since the current error is $e^{(k)} = v_{\mathscr{H}} - v_{\mathscr{H}}^{k}$,

$$e^{(k)} = Ce^{(k-1)} = C(Ce^{(k-2)}) = C^2e^{(k-2)} = \cdots = C^ke^{(0)}. \tag{11}$$

Note that, the superscript of C is the *power* of C, while the superscript of e (inside parentheses) is the *iteration number* to which this particular error coresponds. By applying matrix norms to Eq. (11) will produce $||e^{(k)}|| = ||C^ke^{(0)}|| \leq ||C||^k||e^{(0)}||$, then $||e^{(k)}|| \to 0$ if $||C|| \leq 1$.

5. Numerical Results and Discussion

The proposed method was applied to the following test problem which was used by Sakurai et al. (2002), Othman and Abdullah (2000) and Othman et al. (2004),

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = (x^2 + y^2)e^{xy}, \quad (x, y) \in \Omega = [0, 1] \times [0, 1], \tag{12}$$

subject to the Dirichlet boundary conditions and satisfying the exact solution $u(x, y) = e^{xy}$ for $(x, y) \in \partial\Omega$.

For the hardware, all computations are performed on Intel(R) Core(TM)2 Duo, 3.16 GHz CPU, 4.00 GB memory and 32-bit Operating System. As for the software, all computations are performed using MATLAB R2011a on the Windows 7 Operating System.

Here, the performance metrics are analyzed by examining the execution time, the memory cost and the accuracy. The execution time is the total time taken for the method to converge which is measured using the MATLAB software. The memory cost is mainly determined by the total memory cost of the program which is measured using the computer's task manager. The accuracy of the method is determined by the average absolute error which is calculated after the method has converged.

All results of the numerical experiments, which were gained from the implementations of Gauss-Seidel method with the presence of $\mathscr{H}$-matrix were recorded in Table 1. This implementation was carried out on several grid sizes, 16, 32, 64 and 128, and the convergence test used was the average absolute error with the error tolerance $\varepsilon = 10^{-10}$. The proposed method, denoted as $\mathscr{H}_W$-matrix, was compared with another $\mathscr{H}$-matrix structure. This structure was proposed by Börm et al. (2003b) that used the standard admissibility condition, denoted as $\mathscr{H}_S$-matrix, and was used as a benchmark result.

Table 1: Comparison of Iterations, the Average Absolute Errors, the Execution time (in seconds) and the Memory Cost (in MB) for the $\mathscr{H}_W$-Std51h and $\mathscr{H}_S$-Std51h methods.

n	Methods	Ite	Ave. Abs. Error	Time (s)	Mem. (MB)
16	$\mathscr{H}_W$-Std51h	597	2.2444E-4	0.3588	9.2773
	$\mathscr{H}_S$-Std51h	597	3.0627E-4	0.4680	12.3867
32	$\mathscr{H}_W$-Std51h	2105	3.1375E-5	2.9484	9.3242
	$\mathscr{H}_S$-Std51h	2105	6.1090E-5	4.9764	12.4102
64	$\mathscr{H}_W$-Std51h	7587	4.2058E-6	52.5099	9.3750
	$\mathscr{H}_S$-Std51h	7587	1.1153E-5	52.9155	12.5391
128	$\mathscr{H}_W$-Std51h	27564	5.5879E-7	501.4184	9.6680
	$\mathscr{H}_S$-Std51h	27564	2.0066E-6	1127.6688	12.6406

Table 2: Comparison of the average low-rank values at different sizes between $\mathscr{H}$-matrices.

n	Average Rank	
	$\mathscr{H}_W$-Std51h	$\mathscr{H}_S$-Std51h
16	1.71	1.25
32	2.13	1.55
64	2.58	1.88
128	3.05	2.26

From Table 1, as the matrix size increases, the $\mathscr{H}_W$-Std51h is shown to be more accurate than $\mathscr{H}_S$-Std51h. This is because the $\mathscr{H}_W$-Std51h has a higher average low-rank values, as shown in Table 2, which produces a more accurate $\mathscr{H}$-matrix. Moreover, the $\mathscr{H}_W$-Std51h has less execution time and memory cost. This is because the $\mathscr{H}_W$-Std51h has a more coarser structure compared to the $\mathscr{H}_S$-Std51h. This shows that the adaptation of the proposed $\mathscr{H}_W$-matrix produces a more superior results compared to the $\mathscr{H}_S$-matrix structure. Note that this paper focuses more on the memory utilization. The only difference is the $\mathscr{H}$-matrix structure, but both structures use the standard five-point method. The graphical representations of Table 1 are shown in Figures 3, 4, and 5.

6. Conclusion and Future Works

In this paper, the structure of $\mathscr{H}$-matrix has been successfully applied in an iterative solver. From the results, it shows that the proposed $\mathscr{H}_W$-Std51h is $26-72\%$ more accurate, converges $0-56\%$ faster and utilizes $23-26\%$ less memory space when compared to the $\mathscr{H}_S$-Std51h. The idea of this proposed method can be extended to different points and group iterative solver which will be reported separately in the future.

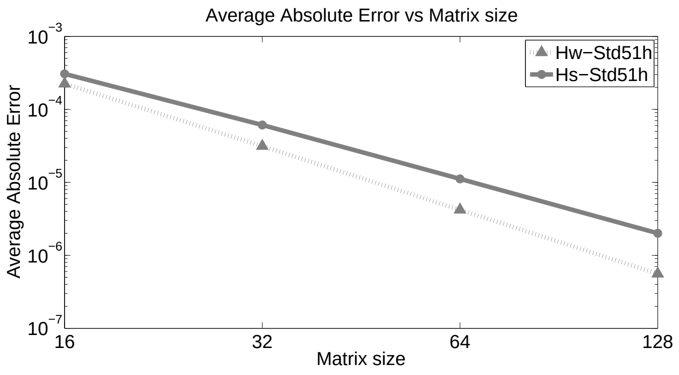

Figure 3: Comparison of Average Absolute Errors

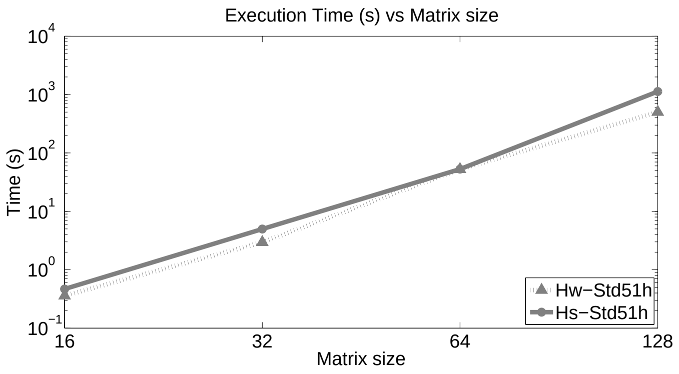

Figure 4: Comparison of Execution Time (in seconds)

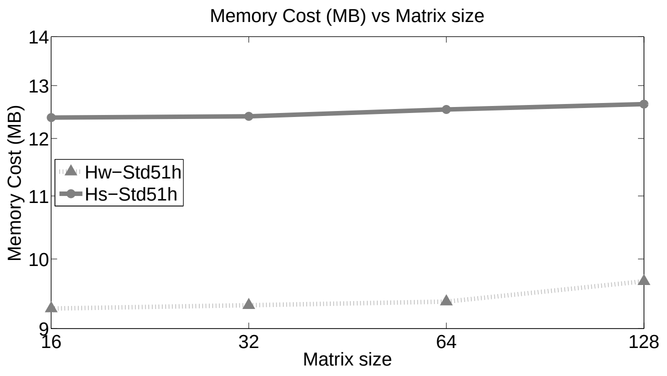

Figure 5: Comparison of Memory Cost (in MB)

Acknowledgement

This research was supported by the Malaysian Ministry of Education under the Fundamental Research Grant Scheme, FRGS/1/2014/ICT03/UPM/01/1.

References

Benner, P. and Mach, T. (2010). On the QR Decomposition of $\mathscr{H}$-matrices. *Computing*, 88:111–129.

Börm, S., Grasedyck, L., and Hackbusch, W. (2003a). Hierarchical Matrices. Lecture notes, URL: www.mis.mpg.de/scicomp/Fulltext/WS_HMatrices.pdf (Last access: 14 May 2015).

Börm, S., Grasedyck, L., and Hackbusch, W. (2003b). Introduction to Hierarchical Matrices with Applications. *Engineering Analysis with Boundary Elements*, 27:405–422.

Engquist, B. and Ying, L. (2011). Sweeping Preconditioner for the Helmholtz Equation: Hierarchical Matrix Representation. *Communications on Pure and Applied Mathematics*, 64:697–735.

Grasedyck, L., Hackbusch, W., and Kriemann, R. (2008). Performance of $\mathscr{H}$-LU Preconditioning for Sparse Matrices. Max-Planck-Institut für Mathematik in den Naturwissenschaften Leipzig.

Hackbusch, W. (1999). A Sparse Matrix Arithmetic Based on $\mathscr{H}$-matrices. part I: Introduction to $\mathscr{H}$-matrices. *Computing*, 62:89–108.

Hackbusch, W. and Khoromskij, B. N. (2000). A Sparse $\mathscr{H}$-Matrix Arithmetic. part II: Application to Multi-Dimensional Problems. *Computing*, 64:21–47.

Hackbusch, W., Khoromskij, B. N., and Kriemann, R. (2004). Hierarchical Matrices Based on a Weak Admissibility Criterion. *Computing*, 73:207–243.

Izadi, M. (2012). *Hierarchical Matrix Techniques on Massively Parallel Computers*. PhD thesis, Max Planck Institute for Mathematics in the Sciences, Leipzig.

Lin, L., Lu, J., and Ying, L. (2011). Fast Construction of Hierarchical Matrix Representation from Matrix-Vector Multiplication. *Journal of Computational Physics*, 230:4071–4087.

MATLAB® (2015). *The MathWorks, Inc.* Primer.

Othman, M. and Abdullah, A. R. (2000). An Efficient Four Points Modified Explicit Group Poisson Solver. *International Journal of Computer Mathematics*, 76:203–217.

Othman, M., Abdullah, A. R., and Evans, D. J. (2004). A Parallel Four Points Modified Explicit Group Algorithm on Shared Memory Multiprocessors,. *International Journal of Parallel, Emergent and Distributed Systems (formerly known as Parallel Algorithms and Applications)*, 19(1):1–9.

Sakurai, K., Aoki, T., Lee, W. H., and Kato, K. (2002). Poisson Equation Solver with Fourth-Order Accuracy by using Interpolated Differential Operator Scheme. *Computers and Mathematics with Applications*, 43:621–630.

Syafiq, N. A., Othman, M., and Senu, N. (2015). Hierarchical Matrix ($\mathscr{H}$-Matrix) Adaptation on Iterative Solvers. 8th International Congress on Industrial and Applied Mathematics, Beijing, China. (Extended abstract), `www.iciam2015.cn/Program%20&%20Abstracts.pdf`.

Varga, R. S. (2000). *Matrix Iterative Analysis*, volume 27. Springer-Verlag Berlin Heidelberg, second edition.

Wan, T., Hu, X. Q., and Chen, R. S. (2011). Hierarchical LU Decomposition-based Direct Method with Improved Solution for 3D Scattering Problems in fem. *Microwave and Optical Technology Letters.*

Young, D. M. (1971). *Iterative Solution of Large Linear Systems.* Harcourt Brace Jovanovich, Academic Press, INC. (London) LTD.