An Adaptive Hierarchical Matrix on Point Iterative Poisson Solver

Nik Amir Syafiq\textsuperscript{1}, Mohamed Othman *\textsuperscript{1,2}, and Norazak Senu\textsuperscript{1,2}

\textsuperscript{1}Institute for Mathematical Research, Universiti Putra Malaysia, Malaysia
\textsuperscript{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 (\mathcal{H}\textsuperscript{-}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 \mathcal{H}\textsuperscript{-}matrix structures, \mathcal{H}\textsubscript{S}\textsuperscript{-} and \mathcal{H}\textsubscript{W}\textsuperscript{-} respectively. The adaption of the \mathcal{H}\textsuperscript{-}matrices to a linear system leads to the saving of memory utilization. An experiment was conducted which compares the proposed \mathcal{H}\textsubscript{W}\textsuperscript{-}matrix with the benchmarked \mathcal{H}\textsubscript{S}\textsuperscript{-}matrix. The results showed the superiority of the proposed method when comparing both \mathcal{H}\textsuperscript{-}matrix structures\textsuperscript{1}.

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

\textsuperscript{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 \( \mathcal{H} \)-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 carefully 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 \( \mathcal{H} \)-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 \( \mathcal{H} \)-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 \( \mathcal{H} \)-matrices.

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

Most researches in \( \mathcal{H} \)-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 \( \tau \) and \( \sigma \) be two index sets. A block \( b = \tau \times \sigma \) is said to satisfy the standard admissibility condition (or \( \eta \)-admissible) if

\[
\text{Adm}_\eta(b) = \text{true} \iff \min(\text{diam}(\Omega_\tau), \text{diam}(\Omega_\sigma)) \leq \eta \text{ dist}(\Omega_\tau, \Omega_\sigma),
\]

where \( \Omega_\tau \) and \( \Omega_\sigma \) are a union of the supports of the respective basis function.
An Adaptive Hierarchical Matrix on Point Iterative Poisson Solver

\[ \varphi_i, \text{ namely} \]
\[ \Omega^\tau := \bigcup_{i \in \tau} \text{supp}(\varphi_i), \quad \Omega^\sigma := \bigcup_{i \in \sigma} \text{supp}(\varphi_i). \]

From this definition, the term “diam” and “dist” are denoted as the Euclidean diameter and distance of \( \Omega^\tau \) and \( \Omega^\sigma \) that are defined as follows:

\[ \text{diam}(\Omega^\tau) := \max_{x_i, x_j \in \Omega^\tau} ||x_i - x_j||, \]
\[ \text{dist}(\Omega^\tau, \Omega^\sigma) := \min_{x_i \in \Omega^\tau, x_j \in \Omega^\sigma} ||x_i - x_j||. \]

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

![Figure 1: 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.](image)

This could affect the accuracy of the \( H \)-matrix. Thus, in this paper, the proposed method utilized an \( H \)-matrix structure with less low-rank sub-matrices that could produce a more accurate \( H \)-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,
\[
\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y), \ (x, y) \in \Omega, \quad (2)
\]
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 \( \Omega \) with grid spacing \( h \) in both directions and \( x_i = x_0 + ih, \ y_j = y_0 + jh \), for all \( i, j = 0, 1, \ldots, n, n + 1 \) 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,
\[
v_{i+1,j} + v_{i-1,j} + v_{i,j+1} + v_{i,j-1} - 4v_{i,j} = h^2 f_{i,j}, \quad (3)
\]
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,
\[
A v = f, \quad (4)
\]
where \( A \) is a square nonsingular matrix while \( v \) and \( f \) are denoted by the following column vectors containing all unknowns as,
\[
v = (v_{1,1}, v_{2,1}, \ldots, v_{n,1}, v_{1,2}, v_{2,2}, \ldots, v_{n,n-1}, v_{1,n}, v_{2,n}, \ldots, v_{n,n})^T;
\]
\[
f = (f_{1,1}, f_{2,1}, \ldots, f_{n,1}, f_{1,2}, f_{2,2}, \ldots, f_{n,n-1}, f_{1,n}, f_{2,n}, \ldots, f_{n,n})^T.
\]
Let \( P_m \) be defined as the unknowns in the \( m \)th row
\[
P_m = (P_{1,m}, \ldots, P_{n,m})^T.
\]
This can help introduce
\[
v_m = (v_{1,m}, v_{2,m}, \ldots, v_{n,m})^T \quad \text{and} \quad f_m = (f_{1,m}, f_{2,m}, \ldots, f_{n,m})^T,
\]
which yields
\[
v = (v_1^T, v_2^T, \ldots, v_n^T)^T \quad \text{and} \quad f = (f_1^T, f_2^T, \ldots, f_n^T)^T.
\]
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. $\mathcal{H}$-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 $\mathcal{H}$-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

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

where $\tau$ and $\sigma$ 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:

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

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:

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

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 $\mathbb{R}^{4 \times 4}$. 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 $H$-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.
To construct the $\mathcal{H}$-matrix, the following recursive algorithm is proposed:
Algorithm 1 hmconstruct\((n, A)\)

Require: size \(n\), matrix \(A \in \mathbb{R}^{n \times n}\)

Ensure: \(\mathcal{H}\)-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 \(X = U \ast S \ast V'\).
7: \((U, S, V) = \text{svd}(A_{1, \ldots, \frac{n}{2}, (\frac{n}{2})+1, \ldots, n})\); \(\triangleright\) MATLAB built-in function
8: \(A_{1, \ldots, \frac{n}{2}, (\frac{n}{2})+1, \ldots, n} = U_{1, \ldots, \frac{n}{2}, 1, \ldots, r} \ast S_{1, \ldots, r, 1, \ldots, r} \ast V_{1, \frac{n}{2}, 1, \ldots, r}'\);
9: \((U, S, V) = \text{svd}(A_{(\frac{n}{2})+1, \ldots, n, 1, \ldots, \frac{n}{2}})\);
10: \(A_{(\frac{n}{2})+1, \ldots, n, 1, \ldots, \frac{n}{2}} = U_{1, \frac{n}{2}, 1, \ldots, r} \ast S_{1, \ldots, r, 1, \ldots, r} \ast V_{1, \frac{n}{2}, 1, \ldots, r}'\);

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 \(\mathcal{H}\)-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 \(\mathcal{H}\)-matrix construction. Here, the program will go through Algorithm 1 to get its \(\mathcal{H}\)-matrix approximation \((v_{\text{new}} = \mathcal{H}(v_{\text{old}}))\). 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

1: Initialize: $u, v, e, f \in \mathbb{R}^{(m+2) \times (m+2)}$; $\epsilon_{\text{target}} = 1.0e − 10$;
2: start timer;
3: Let localf = 0;
4: while localf $\neq 1$ do
5:  localf = 1
6:  $u = \text{std51h}(m, u, f)$; $\triangleright$ Standard five-point method
7:  $u_{2\ldots m+1,2\ldots m+1} = \text{hmconstruct}(m, u_{2\ldots m+1,2\ldots m+1})$ $\triangleright$ Algorithm 1
8:  for $i = 2, 3, \ldots, m + 1$ do
9:     for $j = 2, 3, \ldots, m + 1$ do
10:        if $\text{abs}(u_{i,j} - v_{i,j}) > \epsilon_{\text{target}}$ then
11:           localf = 0; $\triangleright$ Check for convergence
12:        end if
13:     end for
14:  end for
15:  $v = u$; $\triangleright$ Swap old values becomes new values
16: end while
17: end timer;
18: display average absolute error and time taken to execute program;

4. Convergence Analysis

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

$$Av_\mathcal{H} = f,$$

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

$$Mv_{\mathcal{H}}^{k+1} = Nv_{\mathcal{H}}^k + f,$$

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

$$v_{\mathcal{H}}^{k+1} = M^{-1}Nv_{\mathcal{H}}^k + M^{-1}f,$$
which can be simplified into

\[ v^{k+1}_{\mathcal{J}} = Cv^k_{\mathcal{J}} + \tilde{f}, \]  

(8)

where \( C = M^{-1}N \) is often referred to as the iteration matrix and \( \tilde{f} = M^{-1}f \). If the current approximation \( v^k_{\mathcal{J}} \) is, in fact, the exact solution \( v_{\mathcal{J}} \), then the iterative method should produce a next iteration \( v^{k+1}_{\mathcal{J}} \) that is also the exact solution. That is, it should be true that

\[ v_{\mathcal{J}} = Cv_{\mathcal{J}} + \tilde{f}. \]  

(9)

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

\[ v_{\mathcal{J}} - v^{k+1}_{\mathcal{J}} = C(v_{\mathcal{J}} - v^k_{\mathcal{J}}). \]  

(10)

That is, since the current error is \( e^{(k)} = v_{\mathcal{J}} - v^k_{\mathcal{J}} \),

\[ e^{(k)} = Ce^{(k-1)} = C(Ce^{(k-2)}) = C^2e^{(k-2)} = \ldots = C^k e^{(0)}. \]  

(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 corresponds. By applying matrix norms to Eq. (11) will produce \( ||e^{(k)}|| = ||C^k e^{(0)}|| \leq ||C||^k ||e^{(0)}|| \), 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),

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

(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 $H$-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 $H_W$-matrix, was compared with another $H$-matrix structure. This structure was proposed by [Börm et al. (2003b)] that used the standard admissibility condition, denoted as $H_S$-matrix, and was used as a benchmark result.

<table>
<thead>
<tr>
<th>n</th>
<th>Methods</th>
<th>Ite</th>
<th>Ave. Abs. Error</th>
<th>Time (s)</th>
<th>Mem. (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>$H_W$-Std51h</td>
<td>597</td>
<td>2.2444E-4</td>
<td>0.3588</td>
<td>9.2773</td>
</tr>
<tr>
<td></td>
<td>$H_S$-Std51h</td>
<td>597</td>
<td>3.0627E-4</td>
<td>0.4680</td>
<td>12.3867</td>
</tr>
<tr>
<td>32</td>
<td>$H_W$-Std51h</td>
<td>2105</td>
<td>3.1375E-5</td>
<td>2.9484</td>
<td>9.3242</td>
</tr>
<tr>
<td></td>
<td>$H_S$-Std51h</td>
<td>2105</td>
<td>6.1090E-5</td>
<td>4.9764</td>
<td>12.4102</td>
</tr>
<tr>
<td>64</td>
<td>$H_W$-Std51h</td>
<td>7587</td>
<td>4.2058E-6</td>
<td>52.5099</td>
<td>9.3750</td>
</tr>
<tr>
<td></td>
<td>$H_S$-Std51h</td>
<td>7587</td>
<td>1.1153E-5</td>
<td>52.9155</td>
<td>12.5391</td>
</tr>
<tr>
<td>128</td>
<td>$H_W$-Std51h</td>
<td>27564</td>
<td>5.5879E-7</td>
<td>501.4184</td>
<td>9.6680</td>
</tr>
<tr>
<td></td>
<td>$H_S$-Std51h</td>
<td>27564</td>
<td>2.0066E-6</td>
<td>1127.6688</td>
<td>12.6406</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>n</th>
<th>Average Rank</th>
<th>$H_W$-Std51h</th>
<th>$H_S$-Std51h</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1.71</td>
<td>1.25</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>2.13</td>
<td>1.55</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>2.58</td>
<td>1.88</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>3.05</td>
<td>2.26</td>
<td></td>
</tr>
</tbody>
</table>

From Table 2 as the matrix size increases, the $H_W$-Std51h is shown to be more accurate than $H_S$-Std51h. This is because the $H_W$-Std51h has a higher average low-rank values, as shown in Table 2 which produces a more accurate $H$-matrix. Moreover, the $H_W$-Std51h has less execution time and memory cost. This is because the $H_W$-Std51h has a more coarser structure compared to the $H_S$-Std51h. This shows that the adaptation of the proposed $H_W$-matrix produces a more superior results compared to the $H_S$-matrix structure. Note
that this paper focuses more on the memory utilization. The only difference is the $H$-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 $H$-matrix has been successfully applied in an iterative solver. From the results, it shows that the proposed $H_{\text{W-Std51h}}$ is 26 – 72% more accurate, converges 0 – 56% faster and utilizes 23 – 26% less memory space when compared to the $H_{\text{S-Std51h}}$. 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](image)

![Figure 4: Comparison of Execution Time (in seconds)](image)
An Adaptive Hierarchical Matrix on Point Iterative Poisson Solver

![Memory Cost (MB) vs Matrix size](image)

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


Syafiq, N. A., Othman, M. and Senu, N.


