Newton-SOR Iteration for Solving Large-Scale Unconstrained Optimization Problems with an Arrowhead Hessian Matrices

K Ghazali¹, J Sulaiman², Y Dasril³, and D Gabda⁴

¹,²,⁴Mathematics with Economics Programme, Universiti Malaysia Sabah, 88400 Kota Kinabalu, Sabah, Malaysia.
³Faculty of Electronic and Computer Engineering, Universiti Teknikal Malaysia Melaka, 76100 Melaka, Malaysia.

E-mail: khadizah@ums.edu.my

Abstract. Solving unconstrained optimization problems using Newton method requires calculating Newton’s direction, which involves inverse Hessian matrices. Once the order of Hessian matrices is large, it may be impossible to store the inverse of the Hessian matrices using the direct method. To overcome this problem, we used a point iterative method as an inner iteration in finding Newton direction. Therefore in this paper, we proposed a combination between Newton method and successive overrelaxation (SOR) point iterative method for solving large scale unconstrained optimization problems in which the Hessian of the Newton direction is arrowhead matrices. To calculate and validate the performance of the proposed method, we used a combination of Newton method with Gauss-Seidel point iteration and Jacobi point iteration scheme as a reference method. The proposed method provides results that are more efficient compared to the reference methods in terms of execution time and a number of iteration.

1. Introduction
Unconstrained optimization problems appear in a wide range of applications, including electric power systems for the nonlinear large mesh-interconnected system [1], discrete-time optimal control [2], to simulate the DNA reproduction process [3] and the sign recurrent neural network [4]. Unconstrained optimization problems can be solved with either a direct search method or a gradient descent method. Direct search methods include the Nelder-Mead simplex method [5], Rosenbrock’s method [6], pattern search method [7] and Hooke and Jeeve’s method [8] only considers objective function and not its partial derivatives of the objective function. Despite that, these methods often cited as asymptotically slower in convergence rate than the gradient descent method and showed having limitations in large-scale problems, while gradient descent methods used function derivative and require the objective function to be continuous.

Various gradient descent methods have been developed for solving unconstrained optimization problems such as steepest descent method [9], Newton’s method [10], conjugate gradient method [11], Quasi-Newton method [12] and Levenberg-Marquardt’s algorithm [13]. Not limited to these methods only, many researchers are also modified this existing gradient descent method to have an efficient method as in [14-16]. Out of these methods, the Newton method is one of the most fundamental tools in computational mathematics due to its quadratic convergence [17]. Therefore, in this paper, we choose Newton’s method for solving unconstrained optimization problems. Finding unconstrained minimizer
using Newton’s method can be started via two different approaches; the necessary condition for minimization, which is \( \nabla f(x^*) = 0 \) or by using a truncated Taylor series as stated by [18]. Both approaches, however, need the starting point, \( x^{(0)} \) to be selected closer to the optimal solution so that its convergence is guaranteed. This method also need extra storage for calculating the second derivative of Hessian, especially when involving large scale problems.

Thus, many researchers modified the classical Newton’s method or combining it with another method to overcome this difficulty [18-22]. Grapsa in [19] proposed a modified Newton’s direction method using a proper gradient’s vector modification to have a descent property without a line search technique for solving problems of unconstrained optimization and called it as Componentwise Approximated Gradient (CAG) methods, while Shen et al. [20] presented a regularized Newton method for unconstrained optimization problems with local minimizer Hessian. They used the modified Cholesky factorization algorithm to replace the objective function’s Hessian by a positive definite matrix. In the approach proposed by Shi in [21], a new globalization strategy for solving unconstrained optimization problems based on combining Newton’s method with the steepest descent direction achieved both global convergence and high local convergence order. Taheri et al. [22] recently developed a new algorithm for finding the unconstrained minimizer by combining the anti-gradient direction with the Newton direction to improve the convergence rate of Newton’s method.

Thus, in this paper, motivated by how to improve the computational efficiency and to save the storage requirements by cut down the work in the Newton method. We proposed a more reliable method for finding large scale unconstrained optimizer by combining Newton method with SOR iterative method, namely as Newton-SOR method. This combination uses the SOR iterative method as inner iteration for finding the Newton direction then Newton’s method is used as the outer iteration to estimate the solution of the problem. The SOR iterative method has been first introduced by Young [23], where this method uses the optimal relaxation factor to produce the fastest convergence compare to the classical Jacobi and Gauss-Seidel method [24]. This SOR method is categorized as one of the numerical methods that have an advantage of the efficient point iteration for solving any linear systems [23-26] including large-scale system. Noted that, solving unconstrained optimization problems using Newton-SOR method as proposed in this paper is different from the existing combination of Newton’s method with other methods as stated in the previous paragraph. To analyze the performance of the Newton-SOR method, we consider a combination of Newton method with Jacobi iteration and Newton method with Gauss-Seidel iteration as reference method and they are called as Newton-Jacobi method and Newton-GS method respectively.

To investigate the capability of Newton-SOR method, we consider a large scale unconstrained optimization problem being definite as

\[
\min_{x \in \mathbb{R}^n} f(x)
\]  

(1)

where \( f: \mathbb{R}^n \rightarrow \mathbb{R} \) is the objective function and at least twice continuously differentiable. Since we used Newton’s method for solving problem (1) then we begin with initial guess points, \( x^{(0)} \) of the optimal solution, \( x^* \) to estimate the solution by generate the sequence of iteration \( \{x^{(k)}\} \) approaching the solution. To estimate the solution for problem (1) using classical Newton’s method, require calculating the Newton direction by using direct methods which can be obtained by solving the linear system resulting from solving problem (1). However, solving this linear system for higher dimension directly becomes computationally too costly for calculating and storing the inverse of the Hessian matrix. Thus, using SOR iterative method to discard the disadvantages of Newton’s method can reduced the cost of computing Newton Direction.
2. Derivation of Newton Scheme with an Arrowhead Hessian Matrix

In this section, the Newton iteration for solving large scale unconstrained optimization was formulated for minimizing the quadratic approximation to the objective function, \( f(\mathbf{x}) \) in problem (1) at the current point \( \mathbf{x}^{(k)} \) using the first three terms of Taylor series expansion as:

\[
f(\mathbf{x}) \approx f(\mathbf{x}^{(k)}) + \left[ \nabla f(\mathbf{x}^{(k)}) \right]^T (\mathbf{x} - \mathbf{x}^{(k)}) + \frac{1}{2} (\mathbf{x} - \mathbf{x}^{(k)})^T \nabla^2 f(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}),
\]

where \( \nabla f(\mathbf{x}^{(k)}) \) represent the gradient of the first partial derivatives of \( f(\mathbf{x}) \) and \( \nabla^2 f(\mathbf{x}^{(k)}) = \mathbf{H}(\mathbf{x}^{(k)}) \) represent as the Hessian matrix of second partial derivatives of \( f(\mathbf{x}) \). This quadratics approximation (2) will achieves its minimum value by differentiating it with respect to \( \mathbf{x} \) and equating the resulting expression to zero as:

\[
\nabla f(\mathbf{x}^{(k)}) + \mathbf{H}(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}) = 0.
\]

Thus, we can simplify equation (3) to obtain:

\[
\mathbf{x} = \mathbf{x}^{(k)} - \left[ \mathbf{H}(\mathbf{x}^{(k)}) \right]^{-1} \nabla f(\mathbf{x}^{(k)}).
\]

Setting \( \mathbf{x} = \mathbf{x}^{(k+1)} \) in equation (4), then we can formally define the Newton iteration by:

\[
\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left[ \mathbf{H}(\mathbf{x}^{(k)}) \right]^{-1} \nabla f(\mathbf{x}^{(k)}),
\]

where \( \left[ \mathbf{H}(\mathbf{x}^{(k)}) \right]^{-1} \) is the inverse of the Hessian matrix \( \mathbf{H}(\mathbf{x}^{(k)}) \). Noting that \( \mathbf{d}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \), we can conclude that

\[
\mathbf{d}^{(k)} = - \left[ \mathbf{H}(\mathbf{x}^{(k)}) \right]^{-1} \nabla f(\mathbf{x}^{(k)}),
\]

meaning that the Newton direction (6) can be obtained by solving the Newton equation [27];

\[
\mathbf{H}(\mathbf{x}^{(k)})\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)}).
\]

The Newton direction (6) is a descent direction as long as \( \mathbf{H}(\mathbf{x}^{(k)}) \) is positive definite. Since \( \mathbf{H}(\mathbf{x}^{(k)}) \) is positive definite, then its inverse must exists and is positive definite. As a result, this Newton direction (6) is a descent because it satisfies

\[
\left[ \nabla f(\mathbf{x}^{(k)}) \right]^T \mathbf{d}^{(k)} = - \left[ \nabla f(\mathbf{x}^{(k)}) \right]^T \left[ \mathbf{H}(\mathbf{x}^{(k)}) \right]^{-1} \nabla f(\mathbf{x}^{(k)}) < 0.
\]

In this study, we are interested in solving problem (1) for a large scale unconstrained optimization with the Hessian matrix, \( \mathbf{H}(\mathbf{x}^{(k)}) \) involved is a large and sparse matrix. As a particularly interesting case, in this paper we consider the Hessian of an arrowhead matrix of order \( n \) with the general form is given by [28];

\[
\mathbf{H}(\mathbf{x}^{(k)}) = \begin{bmatrix}
    b_1 & c_1 & c_2 & \cdots & c_{n-2} & c_{n-1} \\
    a_2 & b_2 & 0 & \cdots & 0 & 0 \\
    a_3 & 0 & b_3 & \cdots & 0 & 0 \\
    \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
    a_{n-1} & \vdots & \vdots & \vdots & 0 & b_{n-1} \\
    a_n & 0 & 0 & 0 & 0 & b_n
\end{bmatrix},
\]

where \( b_1 \ldots b_n, a_2 \ldots a_n \) and \( c_1 \ldots c_{n-1} \) are real number.

3. Formulation of the proposed Iterative Method

Based on matrix (9), the Hessian of an arrowhead matrix, \( \mathbf{H}(\mathbf{x}^{(k)}) \) in high dimensions is large scale and sparse. Therefore, solving equation (7) with a coefficient matrix (9) by using a direct method such as Gauss elimination method [29] or simultaneous method [30] needs more memory space and computing time if the coefficient matrix (9) is large. To overcome this problem, we propose an iterative method as
in [31-33] for solving the linear system of equation (7). Let the linear system (7) be rewritten in general form as

$$Hd = f$$ \tag{10}$$

where, $H$ is as in the form of matrix (9), $g^T = [d_1 \ d_2 \ d_3 \ ... \ d_n]$ and $f^T = [f_1 \ f_2 \ f_3 \ ... \ f_n]$. To derive the formulation of the proposed iterative method, we decompose the real coefficient matrix $H$ of the linear system (10) as;

$$H = D - L - U$$ \tag{11}$$
in which $D$ is the nonzero diagonal part, $L$ is strictly lower triangular part and $U$ is strictly upper triangular part, of $H$. By applying the decomposition in equation (11) into a linear system (7), the iterative formulation of the Jacobi method can be stated in vector form as [25,34];

$$d^{(k+1)} = D^{-1} (L + U)g^{(k)} + D^{-1}f$$ \tag{12}$$

Similarly, the vector form of the Gauss-Seidel method can be written as;

$$d^{(k+1)} = (D - L)^{-1}Ud^{(k)} + (D - L)^{-1}f$$ \tag{13}$$

Noted that from equation (12), the new approximation, $d^{(k+1)}$ is generated from multiplying by the inverse of diagonal matrix $D$. While in equation (13), matrix $(D - L)$, the lower triangular part of $H$ is applied. Through this change, the Gauss-Seidel method can reduce the storage of computing $d^{(k+1)}$ up to 50% and is more rapidly convergent than the Jacobi method.

In analogy to what was done for Gauss-Seidel iterations and by considering the implementation of the relaxation factor, $\omega$ into the iteration. Thus, we state the iterative formulation of the SOR method in vector form as [23,26,35];

$$d^{(k+1)} = (D - \omega L)^{-1}(\omega U + (1 - \omega)D)d^{(k)} + \omega(D - \omega L)^{-1}f$$ \tag{14}$$

Referring to the linear system (10), this equation can be rewritten in iterative form as follows;

$$b_id_i + \sum_{i=1}^{n-1} c_id_{i+1} = f_i, \ i = 1$$ \tag{15a}$$

$$a_id_i + b_id_i = f_i, \ i = 2,3, \ldots, n$$ \tag{15b}$$

Therefore, based on equation (15), the implementation of SOR point iterations, each component $d^{(k+1)}_i$, $i = 1,2, \ldots, n$ can be computed as;

$$d^{(k+1)}_i = (1 - \omega)d^{(k)}_i + \frac{\omega}{b_i} \left( f_i - \sum_{j=1}^{n-1} c_id^{(k)}_{i+1} \right), \ i = 1$$ \tag{16a}$$

$$d^{(k+1)}_i = (1 - \omega)d^{(k)}_i + \frac{\omega}{b_i} \left( f_i - a_id^{(k+1)}_i \right), \ i = 2,3, \ldots, n$$ \tag{16b}$$

where $\omega$ represents as a relaxation factor with the optimal value in the range of [1,2] and selected based on the smallest number of inner iterations. Equation (14) is turn out to be the Gauss-Seidel method, with $\omega = 1$ and in many cases, it can converge ten or up to a hundred times faster than Gauss-Seidel method [36]. Theoretically, the optimal value of $\omega$ is given by [23];

$$\omega_{OPT} = \frac{2}{1 + (1 - \rho^2)^{1/2}}$$
with $\rho$ denotes as the spectral radius of the iteration matrix. By using the formulation of SOR iterative method to calculate the Newton direction (6) in Newton equation (7), we propose the reliable algorithm of Newton-SOR scheme for solving problem (1), as stated in Algorithm 1.

**Algorithm 1: Newton-SOR Scheme**

i. Initialize

Set up the objective function: $f(x)$.

$f(x^*) \leftarrow \mathbb{R}$, $x^{(0)} \leftarrow \mathbb{R}^n$, $\varepsilon_1 \leftarrow 10^{-6}$, $\varepsilon_2 \leftarrow 10^{-8}$ and $n \leftarrow \{1000, 5000, 10000, 20000, 30000\}$

ii. Assign the optimal value of $\omega$

iii. For $j = 1, 2, ..., n$, implement

a. Set $d^{(0)} \leftarrow 0$

b. Calculate $f(x^{(k)})$

c. Calculate the approximate value of $d^{(k+1)}_i$ as follows;

\[ d^{(k+1)}_i = (1 - \omega)d^{(k)}_i + \omega \frac{\partial^2 f}{\partial x_i \partial x_i} \sum_{j=1}^{n-1} e_i d^{(k)}_{i+1} \]

\[ d^{(k+1)}_i = (1 - \omega)d^{(k)}_i + \frac{\omega}{\partial f_i} (f_i - a_{ij}d^{(k+1)}_i) \]

iv. Display approximate solutions

**4. Numerical Experiments**

In order to implement our proposed algorithm using C language for solving problem (1), three unconstrained optimization test functions presented in [37], were considered. Each function was run with five different order of Hessian matrix, $n$ ($1000, 5000, 10000, 20000$ and $30000$) and three different initial points, $x^{(0)}$ which is randomly selected but closer to the solution point, $x^*$. Therefore for three test functions, we have 45 test cases. The proposed algorithm was stopped when the convergence test, $\|\nabla f(x)\| < 10^{-8}$ for inner iteration and $\|\nabla f(x)\| < 10^{-6}$ for outer iteration. For the sake of comparison, we used the combination Newton method with other two similar subroutines (Gauss-Seidel and SOR point iterations). Thus, we compare the efficiency of our proposed method with Newton-GS iteration and Newton-SOR iteration subject to the number of inner iterations and the execution time.

Now we consider three test functions with its Hessian of the Newton direction (6) that leads an arrowhead matrix as follows;

**Example 1: LIARWHD Function [37]**

\[ f(x) = \sum_{i=1}^{n} \left( 4(x_i^2 - x_i)^2 + \sum_{i=1}^{n} (x_i - 1)^2 \right) \quad (17) \]

This function has a global minimum, $f^* = 0$ at $x^* = 1$ for $i = 1, 2, ..., n$ with Figure 1 shows the graph of this function when $n = 2$. The used starting points, $x^{(0)}$ were:

(a) $x^{(0)} = (4.0, 4.0, ..., 4.0, 4.0)$

(b) $x^{(0)} = (1.5, 1.5, ..., 1.5, 1.5)$

(c) $x^{(0)} = (3.3, 3.5, ..., 3.3, 3.5)$
Example 2: NONDIA Function [37]

\[
f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} 100(x_1 - x_{i-1})^2
\]  

(18)

This function has a global minimum, \( f^* = 0 \) at \( x_i^* = 1 \), for \( i = 1, 2, ..., n \) with Figure 2 shows the graph of this function when \( n = 2 \). The used starting points, \( \hat{x}^{(0)} \) were:

(a) \( \hat{x}^{(0)} = (-1.0, -1.0, ..., -1.0, -1.0) \)
(b) \( \hat{x}^{(0)} = (2.0, 2.0, ..., 2.0, 2.0) \)
(c) \( \hat{x}^{(0)} = (2.0, 1.5, ..., 2.0, 1.5) \)

Example 3: DIAG-AUP1 Function [37]

\[
f(x) = \sum_{i=1}^{n} 4(x_i^2 - x_1)^2 + \sum_{i=1}^{n} (x_i^2 - 1)^2
\]  

(19)

This function has a global minimum, \( f^* = 0 \) at \( x_i^* = 1 \), for \( i = 1, 2, ..., n \) with Figure 3 shows the graph of this function when \( n = 2 \). The used starting points, \( \hat{x}^{(0)} \) were:

(a) \( \hat{x}^{(0)} = (4.0, 4.0, ..., 4.0, 4.0) \)
(b) \( \hat{x}^{(0)} = (1.5, 1.5, ..., 1.5, 1.5) \)
(c) \( \hat{x}^{(0)} = (3.3, 3.5, ..., 3.3, 3.5) \)

Despite the fact that our initial points were selected randomly, but an initial point given in [37], was used as an option (a). The efficiency comparison results for the execution time (seconds) and the number of inner iterations and the number of outer iterations are tabulated in Table 1. Whereas, Table 2 gives the decrement percentage of the number of inner iterations for Newton-SOR and Newton-GS methods compared to Newton-Jacobi method. In order to better understand the numerical results, we have presented the comparison of speedup ratio for Newton-SOR method with Newton-Jacobi method and Newton-GS method, in Table 3.
Table 1. Comparison of the number of inner iteration, the number of outer iteration and execution time (seconds) for Newton-Jacobi, Newton-GS, and Newton-SOR methods.

| Example | Order of Hessian matrix, n | No. of Inner Iterations | No. of Outer Iterations | Execution Time (seconds) |
|---------|---------------------------|-------------------------|------------------------|--------------------------|
| 1(a)    | 1000                      | 3345                    | 1567                   | 376                      | 129 38 14 | 0.07 0.04 0.01 |
| 5000    | 4771                      | 2194                    | 408                    | 203 52 17               | 0.44 0.38 0.05 |
| 10000   | 5479                      | 2509                    | 443                    | 227 58 17               | 0.99 0.63 0.09 |
| 20000   | 6149                      | 2821                    | 507                    | 240 63 18               | 2.23 1.46 0.19 |
| 30000   | 6456                      | 2899                    | 549                    | 264 65 21               | 3.58 2.07 0.33 |
| 1(b)    | 1000                      | 2289                    | 1013                   | 207                     | 130 32 10 | 0.06 0.03 0.02 |
| 5000    | 2274                      | 1039                    | 232                    | 178 46 8                | 0.25 0.21 0.03 |
| 10000   | 2283                      | 1022                    | 232                    | 219 36 20               | 0.53 0.35 0.06 |
| 20000   | 2287                      | 1038                    | 232                    | 240 52 8                | 1.11 0.65 0.18 |
| 30000   | 2303                      | 1042                    | 236                    | 260 58 15               | 1.68 0.97 0.31 |
| 1(c)    | 1000                      | 3378                    | 1556                   | 394                     | 148 37 17 | 0.07 0.04 0.03 |
| 5000    | 4670                      | 2131                    | 415                    | 202 50 15               | 0.43 0.27 0.07 |
| 10000   | 5242                      | 2384                    | 460                    | 227 57 23               | 1.07 0.56 0.18 |
| 20000   | 5487                      | 2553                    | 538                    | 214 61 20               | 2.02 1.19 0.32 |
| 30000   | 5513                      | 2593                    | 580                    | 254 33 23               | 3.12 1.64 0.50 |
| 2(a)    | 1000                      | 56222                   | 22086                  | 1079                    | 14009 7528 124 | 2.09 1.34 0.03 |
| 5000    | 106378                    | 52084                   | 183                    | 91368 29374 225         | 53.87 18.95 0.26 |
| 10000   | 359514                    | 128743                  | 2633                   | 198451 96881 320        | 227.37 123.63 0.66 |
| 20000   | 593232                    | 243256                  | 3308                   | 416445 209648 319       | 880.42 518.18 1.58 |
| 30000   | 708901                    | 351402                  | 4129                   | 645745 320806 669       | 2017.75 1182.51 3.70 |
| 2(b)    | 1000                      | 101544                  | 38789                  | 2762                    | 14976 7536 214        | 2.71 1.55 0.08 |
| 5000    | 461271                    | 168711                  | 7024                   | 89585 45606 526        | 72.64 37.86 0.82 |
| 10000   | 763029                    | 302950                  | 10346                  | 172737 98059 747       | 256.17 152.13 2.44 |
| 20000   | 1369207                   | 557276                  | 14252                  | 357916 209806 1537     | 1004.41 621.94 7.45 |
| 30000   | 1889369                   | 799141                  | 17801                  | 517316 320742 2134     | 2181.71 1194.97 14.71 |
| 3(a)    | 1000                      | 1021288                 | 38695                  | 2810                    | 14985 7535 110       | 3.68 1.18 0.06 |
| 5000    | 495794                    | 173806                  | 6905                   | 89583 45608 652       | 89.09 33.15 0.84 |
| 10000   | 792833                    | 302014                  | 10989                  | 192949 98058 656      | 333.82 129.24 2.38 |
| 20000   | 1328778                   | 554369                  | 14569                  | 351904 209088 1769     | 1186.07 522.01 7.72 |
| 30000   | 1938600                   | 796395                  | 17818                  | 587757 326742 1446     | 2465.87 1193.60 13.17 |
| 3(b)    | 1000                      | 10287                   | 412                    | 185                    | 44 17 15       | 0.06 0.03 0.01 |
| 5000    | 927                       | 452                     | 184                    | 56 20 17               | 0.22 0.05 0.03 |
| 10000   | 959                       | 463                     | 187                    | 64 21 17               | 0.30 0.11 0.08 |
| 20000   | 971                       | 470                     | 188                    | 71 23 17               | 0.56 0.20 0.14 |
| 30000   | 977                       | 473                     | 188                    | 74 24 18               | 0.87 0.27 0.23 |
| 3(c)    | 1000                      | 595                     | 279                    | 105                    | 42 13 10       | 0.02 0.01 0.00 |
| 5000    | 615                       | 284                     | 106                    | 56 17 10               | 0.08 0.07 0.04 |
| 10000   | 625                       | 286                     | 108                    | 62 19 10               | 0.15 0.07 0.06 |
| 20000   | 629                       | 287                     | 107                    | 68 20 10               | 0.33 0.14 0.11 |
| 30000   | 633                       | 289                     | 107                    | 72 21 10               | 0.47 0.19 0.15 |
| 3(c)    | 1000                      | 820                     | 404                    | 176                    | 45 16 12       | 0.02 0.01 0.01 |
| 5000    | 982                       | 434                     | 170                    | 58 20 14               | 0.10 0.05 0.03 |
| 10000   | 913                       | 441                     | 172                    | 66 22 15               | 0.21 0.09 0.08 |
| 20000   | 922                       | 444                     | 178                    | 71 23 16               | 0.40 0.18 0.16 |
| 30000   | 928                       | 446                     | 176                    | 75 24 16               | 0.64 0.28 0.24 |
Table 2. Decrement percentage of the number of inner iterations compared to Newton-Jacobi method.

| Example | Percentage of the number of inner iterations (%) | $x^{(0)}$ |
|---------|-----------------------------------------------|-----------|
|         | Method                                         | 1         | 2         | 3         |
| (a)     | Newton-GS                                      | 53.13-55.10 | 50.42-73.48 | 50.18-51.60 |
|         | Newton-SOR                                     | 88.75-91.91 | 98.10-99.44 | 77.63-80.76 |
| (b)     | Newton-GS                                      | 54.31-55.74 | 57.93-63.42 | 53.11-54.37 |
|         | Newton-SOR                                     | 89.75-90.96 | 97.28-99.10 | 82.35-83.10 |
| (c)     | Newton-GS                                      | 52.97-54.52 | 58.28-64.94 | 50.73-51.94 |
|         | Newton-SOR                                     | 88.34-91.22 | 97.23-99.08 | 78.54-81.16 |

Table 3. Comparison of speedup ratio for Newton-SOR method with Newton-Jacobi method and Newton-GS method.

| Example | Total Execution Time (seconds) | Speedup Ratio |
|---------|--------------------------------|---------------|
|         | Newton-Jacobi (I) | Newton-GS (II) | Newton-SOR (III) | $I / II$ | $I / III$ | $II / III$ |
| 1 (a)   | 7.31               | 4.58           | 0.67             | 1.60     | 10.91    | 6.84      |
| 1 (b)   | 3.63               | 2.21           | 0.60             | 1.64     | 6.05     | 3.68      |
| 1 (c)   | 6.71               | 3.70           | 1.10             | 1.81     | 6.10     | 3.36      |
| 2 (a)   | 3175.50            | 1844.61        | 6.23             | 1.72     | **509.71** | **296.09** |
| 2 (b)   | 3517.64            | 2008.45        | 25.50            | 1.75     | 137.95   | 78.76     |
| 2 (c)   | 4078.53            | 1879.18        | 24.17            | 2.17     | 168.74   | 77.75     |
| 3 (a)   | 2.01               | 0.66           | 0.49             | 3.05     | 4.10     | 1.35      |
| 3 (b)   | 1.05               | 0.48           | 0.36             | 2.19     | 2.92     | 1.33      |
| 3 (c)   | 1.37               | 0.61           | 0.52             | 2.25     | **2.63** | **1.17**  |

5. Conclusion

As presented in this paper, our proposed combination of Newton method with SOR point iterative method is more efficient in the process for solving large scale unconstrained optimization problems with an arrowhead Hessian matrix. This improvement can be seen through the execution time and the number of iterations given in Table 1 as a result of our implementation proposed algorithm. From Table 2, it is apparent that the number of inner iteration for our proposed method decreased from the referenced methods with the least percentage decrease is 77.63% (in example 3(a)), and the largest percentage decrease is 99.44% (in example 2(a)). As expected, with the use of the relaxation factor, $\omega$, the speedup ratio for our proposed method was much faster than the reference methods. From the speedup ratio in Table 3, we see that our proposed method is 1.17-296.09 times more rapid compared to Newton-Jacobi and 2.63-509.71 times faster compare to Newton-GS. Thus, it can be concluded that our proposed iterative method (Newton-SOR) is able to show substantial improvement in the number of iterations and execution time compared to the Newton-Jacobi and Newton-GS point iterative methods. In the future, this work will investigate the efficiency of the combination of the Newton method with block iterative method as in [32,38].

6. References

[1] Lin S Y and Lin C H 1995 An efficient method for unconstrained optimization problems of nonlinear large mesh-interconnected systems, *IEEE Transaction on automatic control* 40(3) pp 490-495

[2] Ng C K, Liao L Z and Li D 2002 A Globally Convergent and Efficient Method for Unconstrained Discrete-Time Optimal Control, *Journal of Global Optimization* 23: pp 401-421

https://doi.org/10.1023/A:1016595100139
[3] Zhang W and Liu X 2017 A Genetic Algorithm Using Triplet Nucleotide Encoding and DNA Reproduction Operations for Unconstrained Optimization Problems. *Algorithms 2017* **10**(3); 76. https://doi.org/10.3390/a10030076

[4] Maratos N G and Moraitis M A 2018 Some results on the Sign recurrent neural network for unconstrained minimization, *Neurocomputing* **287** pp 1-21. https://doi.org/10.1016/j.neucom.2017.09.036

[5] Dasril Y B and Wen G K 2016 Modified Artificial Bees Colony algorithm with Nelder-Mead search algorithm in 12th International Conference on Mathematics, Statistics and Their Applications (ICMSA), Banda Aceh pp 25-30. doi: 10.1109/ICMSA.2016.7954301

[6] Rosenbrock H H 1960 An automatic method for finding the greatest or least value of a function *Comput. J.* **3** pp 175-84 DOI: 10.1093/comjnl/3.3.175

[7] Mariani V C and Leandro D S C 2011 A hybrid shuffled complex evolution approach with pattern search for unconstrained optimization *Mathematics and Computers in Simulation* **81**(9) pp 1901-1909 doi:10.1016/j.matcom.2011.02.009

[8] Yosef S S and Bruce A B 1994 Optimization by pattern search *European Journal of Operational Research* **78**(3) pp 277-303 https://doi.org/10.1016/0377-2217(94)90041-8

[9] Napitupulu H, Sukono, Mohd B I, Hidayat Y and Suplan Yosef S S and Bruce A B 1994 Optimization by pattern search algorithm in 12th International Conference on Mathematics, Statistics and Their Applications (ICMSA), Banda Aceh pp 25-30. doi: 10.1109/ICMSA.2016.7954301

[10] Babaie-Kafaki S 2016 Emrouznejad A (eds) *Big Data Optimization: Recent Developments and Challenges. Studies in Big Data*, **18**. Springer DOI 10.1007/978-3-319-30265-2_17

[11] Moyi A U and Leong W J 2016 A sufficient descent three-term conjugate gradient method via symmetric rank-one update for largescale optimization *Optimization* **65** (1) pp 121-143.

[12] Aderibigbe F M, Adebayo K J and Dele-Rotimi A O 2015 On quasi-newton method for solving unconstrained optimization problems *American Journal of Applied Mathematics* **3**(2) pp 47-50.

[13] Andreas F and Praduym K S 2008 A Levenberg–Marquardt algorithm for unconstrained multicriteria optimization *Optimations Research Letters* **36**(5) pp 643–646

[14] Fasano G and Lucidi S 2009 A nonmonotone truncated Newton–Krylov method exploiting negative curvature directions, for large scale unconstrained optimization *Optimation Letters* **3**(4) pp 521-535 https://doi.org/10.1007/s11590-009-0132-y

[15] Dehghani R, Hosseini M M and Bidabadi N 2018 The modified quasi-Newton methods for solving unconstrained optimization problems *Int J Numer Model* **32**(1) https://doi.org/10.1002/jnm.2459

[16] Kou C X and Dai Y H 2015 A Modified Self-Scaling Memoryless Broyden-Fletcher-Goldfarb-Shanno Method for Unconstrained Optimization *J Optim Theory Appl* **165** 209-224 DOI 10.1007/s10957-014-0528-4

[17] Nocedal J and Wright S J 2000 *Numerical Optimization* 2nd edn. Springer-Verlag, Berlin

[18] Polyak B T 2007 Newton’s method and its use in optimization *European Journal of Operational Research* **181** pp 1086–1096

[19] Grapsa T N 2014 A modified Newton direction for unconstrained optimization *Optimization* **63**(7) pp 983-1004

[20] Shen C, Chen X and Liang Y 2012 A regularized Newton method for degenerate unconstrained optimization problems *Optim Lett* **6** pp 1913–1933 DOI 10.1007/s11590-011-0386-z

[21] Shi Y 2000 Globally Convergent Algorithms for Unconstrained Optimization *Computational Optimization and Applications* **16** pp 295-308

[22] Taheri S, Mammadov M and Seifollahi S 2015 Globally convergent algorithms for solving unconstrained optimization problems *Optimization* **64**(2) pp 249-263 DOI:10.1080/02331934.2012.745529

[23] Young D M 1954 Iterative methods for solving partial differential equations of elliptic type. *Trans. Amer. Math. Soc.* **76** 92-111 https://doi.org/10.1090/S0002-9947-1954-0059635-7

[24] Young D M 1971 *Iterative solution of large linear systems* Academic Press. London
[25] Eng J H, Saudi A and Sulaiman S 2017 Application of SOR Iteration for Poisson Image Blending International Conference on High Performance Compilation, Computing and Communications 60-64 DOI 10.1145/3069593.3069608

[26] Ali N A M, Rahman R, Sulaiman J and Ghazali K 2018 SOR iterative method with wave variable transformation for solving advection-diffusion equations AIP Conference Proceedings 2013, 020036 https://doi.org/10.1063/1.5054235

[27] Nocedal J and Wright S J 2000 Numerical Optimization 2nd ed. Springer-Verlag, Berlin

[28] Predrag S S, Vasilios N K and Dejan K 2019 Inversion and pseudoinversion of block arrowhead matrices Applied Mathematics and Computation 341 pp 379-401 https://doi.org/10.1016/j.amc.2018.09.006.

[29] Killingbeck J P and Grosjean A 2010 A Gauss elimination method for resonances J Math Chem 47 1027 DOI 10.1007/s10910-009-9622-5

[30] Yunfei L, Jun L and Xiaowei G 2016 The application of simultaneous elimination and backsubstitution method (SEBSM) in finite element method Engineering Computations 33(8) pp 2339-2355 https://doi.org/10.1108/EC-10-2015-0287

[31] Sulaiman J, Hasan M K, Othman M and Karim S A A 2014 Fourth-order solutions of nonlinear two-point boundary value problems by Newton-HSSOR iteration. AIP Conf. Proc. 1602 pp 69-75

[32] Ghazali K, Sulaiman J, Dasril Y and Gabda D 2019 Application of Newton-4EGSOR Iteration for Solving Large Scale Unconstrained Optimization Problems with a Tridiagonal Hessian Matrix. In: Alfred R., Lim Y, Ibrahim A and Anthony P (eds) Computational Science and Technology. Lecture Notes in Electrical Engineering 481 Springer, Singapore https://doi.org/10.1007/978-981-13-2622-6_39

[33] Sulaiman J, Hasan M K, Othman M and Karim S A A 2015 Application of Block Iterative Methods with Newton Scheme for Fisher’s Equation by Using Implicit Finite Difference. Jurnal Kalam. 8(1) pp 039-46

[34] Saad Y 1996 Iterative Methods for Sparse Linear Systems PWS Publishing Company, United States America

[35] Sulaiman J, Hasan M K, Othman M and Karim S A A 2013Numerical solutions of nonlinear second-order two-point boundary value problems using half-sweep SOR with Newton method J. Concr. Appl. Math. 11(1) pp 112–120

[36] Strang G 2009 Introduction to Linear Algebra (4th-Ed) Wellesley Cambridge Press, Wellesley, Ma

[37] Andrei N 2008 An unconstrained optimization test function collection Adv. Modeling and Opt. 10(1) pp 147-161.

[38] Ghazali K, Sulaiman J, Dasril Y and Gabda D 2018 Newton method with explicit group iteration for solving large scale unconstrained optimization problems IOP Conf. Series: J. Phys. Conf. Ser. 1132 (2018) 012056 doi:10.1088/1742-6596/1132/1/012056

Acknowledgments
The authors are grateful for the fund received from Universiti Malaysia Sabah upon publication of this paper (GUG0160-2/2017).