Newton Method with AOR Iteration for Finding Large Scale Unconstrained Minimizer with Tridiagonal Hessian Matrices

K Ghazali\textsuperscript{a,1}, J Sulaiman\textsuperscript{a,2}, Y Dasril\textsuperscript{b,3}, and D Gabda\textsuperscript{a,4}

\textsuperscript{a}Mathematics with Economics Programme, Universiti Malaysia Sabah, 88400 Kota Kinabalu, Sabah, Malaysia
\textsuperscript{b}Faculty of Electronic and Computer Engineering, Universiti Teknikal Malaysia Melaka, 76100 Melaka, Malaysia.

\textsuperscript{1}khadizah@ums.edu.my, \textsuperscript{2}jumat@ums.edu.my, \textsuperscript{3}yosza@utem.edu.my, \textsuperscript{4}darmesah@ums.edu.my

Abstract. Finding the large scale unconstrained minimizer using Newton method has required the calculation of large and complicated linear systems results from solving the Newton direction. Therefore, in this paper, we propose a method for solving large scale unconstrained optimization problems with tridiagonal Hessian matrices to reduce the complexity of calculating Newton direction. Our proposed method was a combination of Newton method and Accelerated Over Relaxation (AOR) iterative method. To evaluate the performance of the proposed method, combination of Newton method with Gauss-Seidel iteration and Newton method with Successive Over Relaxation (SOR) iteration were used as reference method. Finally, the numerical experiment illustrated that the proposed method produce results that are more efficient compared to the reference methods with less execution time and minimum number of iterations.

1. Introduction
Unconstrained optimization problems shows an important role in many areas of mathematics, engineering, medical, computational economics and management. As the scale of unconstrained optimization problems is getting large, we need faster optimization methods that not only work well in theory but also work well in practice by using computing skill. Therefore, developing an efficient methods to solve large scale unconstrained optimization problems is a very important and active research area.

Generally, unconstrained optimization problems can be solved using either direct search methods or gradient descent methods. For this paper, we choose gradient descent methods for solving large scale unconstrained optimization problems. There are various existing gradient descent methods that have been used by previous researcher such as steepest descent method [1], Newton’s method [2], conjugate gradient method [3], Quasi-Newton method [4] and Levenberg-Marquardt’s algorithm [5]. Out of these methods, Newton’s method as one of the most well-known classical methods of solution for unconstrained optimization problems to be discussed in this paper. This is due to the fact that Newton method has an excellent rate of convergence namely quadratic [6] and it also represents the most reliable method for solving large problem [7]. However, this method requires an initial guess to be selected closer to the optimal solution in order to guarantee its convergence. Another problem is
that the Newton’s method can be very costly in calculating and storing the inverse of the Hessian especially for large scale problems.

Thus in this paper, motivated by how to avoid the computational complexity of the inverse Hessian, we propose a new approach for finding large scale unconstrained minimizer. The new approach use a combination of Newton method with AOR iterative method and namely as Newton-AOR method. This combination uses AOR iterative method as inner iteration in finding the Newton direction then Newton’s method is uses as the outer iteration to estimate the solution of problem. The AOR iterative method has been introduced by Hadjidimos [8] using two parameter generalization of the SOR method. This AOR method classified as one of the numerical methods that has an advantage of the efficient point iteration for solving any linear systems [9-11]. To evaluate the performance of the Newton-AOR method, this paper also considers a combination of Newton method with Gauss-Seidel iteration and Newton method with SOR iteration as reference method and they are named as Newton-GS method and Newton-SOR method respectively.

To investigate the capability of Newton-AOR method, let us consider a large scale unconstrained optimization problem being expressed as

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

where \( f: \mathbb{R}^n \rightarrow \mathbb{R} \) is the objective function and at least twice continuously differentiable. In the process of estimating the solution using Newton’s method, a Newton direction is required which can be obtained by solving the linear system resulting from problem (1). Solving this linear system for higher dimension directly becomes computationally too expensive for calculating and storing the Hessian matrix. Thus, using AOR iterative method to discard the disadvantages of Newton’s method can be reduced the computational cost of Newton Direction.

2. Approaches to Newton Scheme with a Tridiagonal 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, \( f(x) \) in problem (1) at the current point \( x^{(k)} \) using the first three terms of Taylor series expansion as:

\[
\begin{align*}
 f(x) & \approx f(x^{(k)}) + \left[ \nabla f(x^{(k)}) \right]^T (x - x^{(k)}) + \frac{1}{2} (x - x^{(k)})^T \nabla^2 f(x^{(k)}) (x - x^{(k)}), \\
 \end{align*}
\]  

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

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

Thus, we can simplify equation (3) and setting \( d^{(k)} = x^{(k+1)} - x^{(k)} \), then we can formally define the Newton direction by:

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

where \([H(x^{(k)})]^{-1}\) is the inverse of the Hessian matrix \( H(x^{(k)}) \). This the Newton direction (4) can be obtained by solving the Newton equation;

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

The Newton direction (4) is a descent direction as long as \( H(x^{(k)}) \) is positive definite. Since \( H(x^{(k)}) \) is positive definite, then its inverse must exist and is positive definite. Notice that, we are interested in solving problem (1) for a large scale unconstrained optimization. Thus, the Hessian matrix, \( H(x^{(k)}) \) involved in this study is a large sparse matrix. As a particularly interesting case, Hessian of a tridiagonal matrix of order \( n \) can be considered [12];
with $h_{i,i} > 0 \forall i, h_{i,j} < 0$ for $i \neq j, h_{i,j} = 0$ if $|i - j| > 1$ and $i, j = 1, 2, ..., n$. Since matrix (6) is symmetric, hence we have

$$a_i = c_{i-1}, i = 2, 3, ..., n.$$  

(7)

Furthermore, this tridiagonal matrix $H(x^{(k)})$ has a diagonal entries that satisfies the inequality

$$|b_i| \geq |a_i| + |c_i|, \forall i = 0, 1, 2, ..., n,$$  

(8)

so that the condition of positive definite can be fulfilled.

3. Formulation of proposed Iterative Method

As mention in the previous section, the Hessian of a tridiagonal matrix, $H(x^{(k)})$ is large and sparse. Therefore, finding the inverse of $H(x^{(k)})$ in high dimensions can be very expensive arithmetic operation. Hence, solving equation (5) by using direct method as in [13,14] will lead to a solution involving very trivial work. To overcome this problem, we propose an iterative method as in [15-17] for solving linear system of equation (5). Let the linear system (5) be rewritten in general form as

$$Hd = f$$  

(9)

where, $H$ is as in the form of matrix (6), $d^T = [d_1, d_2, d_3, ..., d_n]$ and $f^T = [f_1, f_2, f_3, ..., f_n]$. To derive the formulation of proposed iterative method, we decompose the real coefficient matrix $H$ of the linear system (9) as;

$$H = D - L - U$$  

(10)

in which $D$ is the nonzero diagonal part, $L$ is strictly lower triangular part and $U$ is strictly upper part, of $H$. By applying the decomposition in equation (10) into linear system (5), the iterative formulation of the SOR method can be stated in vector form as [18];

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

(11)

For the implementation of SOR point iterations, each component $d_i^{(k+1)}$ can be computed as;

$$d_i^{(k+1)} = (1 - \omega)d_{i-1}^{(k+1)} + \frac{\omega}{a_{ii}}\left(f_i - \sum_{j=1}^{i-1}a_{ij}d_j^{(k+1)} - \sum_{j=i+1}^{n}a_{ij}d_j^{(k)}\right), i = 1, 2, ..., n$$  

(12)

where $\omega$ represents as a relaxation parameter with the optimal value in the range of [1,2] and selected based on the smallest number of inner iterations. In any case, the optimal value is given by [18];

$$\omega_{OPT} = \frac{2}{1 + (1 - \rho^2)^{1/2}}$$  

with $\rho$ denotes as the spectral radius of the iteration matrix. In analogy to what was done for SOR iterations and by considering the implementation of two different relaxation parameters, which are $r$ as an acceleration parameter and $\omega$ as an overrelaxation parameter. Therefore, we derived the formulation of AOR method [18] with the iterative scheme is given as;

$$d_i^{(k+1)} = (1 - \omega)d_{i-1}^{(k+1)} + \frac{\omega}{a_{ii}}\left(f_i - \sum_{j=1}^{i-1}a_{ij}d_j^{(k+1)} - \sum_{j=i+1}^{n}a_{ij}d_j^{(k)}\right) - \frac{r}{a_{ii}}\sum_{j=1}^{i-1}a_{ij}(d_j^{(k+1)} - d_j^{(k)}), i = 1, 2, ..., n$$  

(13)

By using the formulation of AOR iterative method to calculate the Newton direction (4) in Newton equation (5), we proposed the reliable algorithm of Newton-AOR scheme for solving problem (1), as stated in Algorithm 1. Note that for $\omega = r = 1$, equation (13) is reduced to the GS method and if $\omega = r$, then equation (13) coincides with SOR method. Hereby, to evaluate the performance of our proposed iterative method, the Newton-GS and Newton-SOR iterative methods are used as the reference method.
Algorithm 1: Newton-AOR Scheme

i. Initialize
   Set up the objective function: \( f(x) \), \( f'(x) \), \( \mathbb{R} \), \( x^{(0)} \), \( n \leftarrow \{1000, 5000, 10000, 20000, 30000\} \), \( \varepsilon_1 \leftarrow 10^{-6} \), \( \varepsilon_2 \leftarrow 10^{-6} \).

ii. Assign the optimal value of \( \omega \) and \( r \).

iii. For \( j = 1, 2, \ldots, n \), implement
   a. Set \( d^{(0)} \leftarrow 0 \)
   b. Calculate \( f'(x^{(k)}) \)
   c. For \( i = 1, 2, \ldots, n \), calculate iteratively; solve equation (9) by using equation (13)
   d. Check the convergence test, \( \|d^{(k+1)} - d^{(k)}\| < \varepsilon_2 \). If yes, go to step (e). Otherwise go back to step (b)
   e. For \( i = 1, 2, \ldots, n \), calculate: \( x^{(k+1)} = x^{(k)} + d^{(k)} \)
   f. Check the convergence test, \( \|f'(x^{(k)})\| \leq \varepsilon_1 \). If yes, go to (iii). Otherwise go back to step (a)

iv. Display approximate solutions

4. Numerical Experiments
Next we tested our algorithm using three different test functions taken from [19-21] and we implemented the algorithm in C language to solve problem (1). For each test function we considered five numerical experiments with dimensions varied from 1000 to 30000 variables as listed in Table 1 and tested with 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. In our numerical experiments, the termination criterion are \( \|d^{(k+1)} - d^{(k)}\| < 10^{-3} \) for inner iteration and \( \|f'(x)\| < 10^{-6} \) for outer iteration. For the sake of comparison, we used combination Newton method with other two similar subroutines (GS 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 (generally used for unconstrained optimization test problem) with its Hessian of the Newton direction (4) that leads a tridiagonal matrix as follows:

Example 1: Dixon and Price Function [19]
\[
f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} (2x_i^2 - x_{i-1})^2
\]
This function has a global minimum, \( f^* = 0 \) at \( x_i^* = 2 \left[ \frac{2i^2}{i^2} \right] \), for \( i = 1, 2, \ldots, n \). The used starting points, \( x^{(0)} \) were
(a) \( x^{(0)} = (0.6, 0.6, \ldots, 0.6) \), (b) \( x^{(0)} = (1.0, 1.0, \ldots, 1.0) \) and (c) \( x^{(0)} = (0.6, 1.0, \ldots, 0.6, 1.0) \)

Example 2: NONSCOMP Function [20]
\[
f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} (4x_i - x_{i-1})^2
\]
This function has a global minimum, \( f^* = 0 \) at \( x_i^* = 1 \), for \( i = 1, 2, \ldots, n \). The used starting points, \( x^{(0)} \) were:
(a) \( x^{(0)} = (1.5, 1.5, \ldots, 1.5) \), (b) \( x^{(0)} = (3.0, 3.0, \ldots, 3.0) \) and (c) \( x^{(0)} = (1.5, 1.0, \ldots, 1.5, 1.0) \)

Example 3: Cube Function [21]
\[
f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n-1} 100(x_{i+1} - x_i)^2
\]
This function has a global minimum, \( f^* = 0 \) at \( x_i^* = 1 \), for \( i = 1, 2, \ldots, n \). The used starting points, \( x^{(0)} \) were:
(a) $\mathbf{x}^{(0)} = (-1.2, -1.2, ..., -1.2),$ (b) $\mathbf{x}^{(0)} = (0.9, 0.9, ..., 0.9)$ and (c) $\mathbf{x}^{(0)} = (1.2, 1.0, ..., 1.2, 1.0)$

The numerical results for 45 test cases stated in Table 1. The efficiency comparison results for the execution time (seconds) and the number of iteration are presented in Table 1. From Table 1 also, it is obvious that the number of inner iteration for our proposed method reduce well compared from the referenced method. Whereas, Table 2 shows that the decrement percentage of the number of inner iterations and the execution time for Newton-AOR method is superior that the Newton-SOR method compared to Newton-GS method.

**Table 1.** Comparison of number of inner iteration and execution time (second) for Newton-GS, Newton-SOR and Newton-AOR method.

| Example | Method      | 1000 | 5000 | 10000 | 20000 | 30000 |
|---------|-------------|------|------|-------|-------|-------|
|         | Number of inner iterations (Number of outer iterations) / Execution time |      |      |       |       |       |
| (a)     | Newton-GS   | 193(20)/0.01 | 190(26)/0.05 | 202(29)/0.10 | 204(31)/0.20 | 206(33)/0.31 |
|         | Newton-SOR  | 100(11)/0.00 | 103(14)/0.03 | 104(15)/0.06 | 105(16)/0.12 | 106(17)/0.20 |
|         | Newton-AOR  | 91(10)/0.00 | 94(13)/0.02 | 95(14)/0.05 | 96(15)/0.10 | 97(16)/0.16 |
| (b)     | Newton-GS   | 282(16)/0.01 | 288(22)/0.05 | 290(24)/0.11 | 293(27)/0.23 | 294(28)/0.34 |
|         | Newton-SOR  | 147(8)/0.01 | 150(11)/0.05 | 152(13)/0.07 | 154(14)/0.13 | 154(15)/0.19 |
|         | Newton-AOR  | 139(7)/0.00 | 142(10)/0.03 | 143(11)/0.06 | 144(12)/0.12 | 145(13)/0.17 |
| (c)     | Newton-GS   | 276(18)/0.02 | 282(24)/0.11 | 286(24)/0.11 | 287(29)/0.23 | 288(30)/0.35 |
|         | Newton-SOR  | 157(9)/0.01 | 160(12)/0.03 | 162(14)/0.07 | 163(15)/0.14 | 164(16)/0.21 |
|         | Newton-AOR  | 138(8)/0.00 | 140(10)/0.03 | 142(12)/0.12 | 143(13)/0.12 | 144(14)/0.19 |

**Table 2.** Decrement percentage of the number of inner iterations and the execution time compared to Newton-GS method.

| $\mathbf{x}^{(0)}$ | Example | Number of inner Iterations (%) | Execution Time (%) |
|-------------------|---------|--------------------------------|-------------------|
|                   |         | 1 | 2 | 3 | 1 | 2 | 3 |
| (a)               | Newton-SOR | 48.19 – 48.54 | 95.86 – 95.91 | 4.99 | 35.48 – 100.00 | 95.31 – 95.44 | 1.79 – 40.00 |
|                   | Newton-AOR | 52.76 – 52.97 | 97.29 | 6.45 | 48.39 – 100.00 | 96.81 – 97.02 | 5.36 – 60.00 |
| (b)               | Newton-SOR | 47.59 – 47.92 | 99.12 – 99.27 | 5.07 | 0.00 – 44.12 | 99.02 – 99.25 | 2.38 – 25.00 |
|                   | Newton-AOR | 50.68 – 50.85 | 99.44 | 8.70 | 40.00 – 100.00 | 99.33 – 99.51 | 7.14 – 50.00 |
| (c)               | Newton-SOR | 42.96 – 43.26 | 95.97 | 11.69 | 36.36 – 50.00 | 94.99 – 95.44 | 1.75 – 25.00 |
|                   | Newton-AOR | 50.00 – 50.35 | 97.39 | 13.25 | 45.45 – 100.00 | 96.98 – 97.22 | 3.51 – 50.00 |

5. Conclusion
As presented in this paper, the combination of Newton method with AOR point iterative method has speed up the process for solving large scale unconstrained optimization problems with a tridiagonal Hessian matrix. This can be seen through the execution time and the number of iterations given in Table 1 as a result of our implementation proposed algorithm. In addition to that, the numerical results stated in Table 2 clearly has showed that our proposed method has reduced the number of inner iterations better than the reference algorithm with less execution time (second). Therefore, it can be concluded that our proposed iterative method (Newton-AOR) is able to show substantial improvement in the number of inner iterations and execution time compared to the Newton-GS and Newton-SOR point iterative methods. In future, this work will investigate the efficiency of the combination of Newton method with block iterative method as in [16, 22].

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

7. References
[1] Napitupulu H, Sukono, Mohd B I, Hidayat Y and Suplan S 2018 Steepest descent method implementation on solving optimization problem using C++ program. *Materials Sci. and Eng.* 332 012024.
[2] 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
[3] 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) 121-143.
[4] Aderibigbe F M, Adebayo K J and Dele-Rotimi A O 2015 *America Journal of Applied Mathematics* 6 47-50.
[5] Andreas F and Pradyunn K S 2008 *Operations Research Letters* 36 643–646.
[6] Taheri S, Mammadov M and Seifollahi S 2015 *Optimization*, 64(2) 249-263.
[7] Noceval J 1992 Theory of Algorithms for Unconstrained Optimization. *Acta Numerica*, 1, 199-242.
[8] Hadhidiemons A 1978 Accelerated Overrelaxation Method. *Mathematics of Computation*, 32(141), 149–157.
[9] Yousif W S and Martins M M 2008 Explict De-couple Group AOR method for solving elliptic partial differential equations. *Neural Parallel and Scientific Computations* 16(4) 531-542.
[10] Akhir M K M and Sulaiman J 2017 Accelerated over relaxation iterative method using triangle element approximation for solving 2D Helmholtz Equations. *J. Phys: Conf. Ser.* 890 012060. DOI:10.1088/1742-6596/890/1/012060
[11] Basran N A, Eng J H, Saudi A, and Sulaiman J 2018 Numerical analysis of AOR iteration for anisotropic diffusion equation in image blurring. *AIP Conf. Proc.*, 020040. https://doi.org/10.1063/1.5054239
[12] Li H-B, Huang T-Z, Lui X-P and Li H 2010 On the inverses of general tridiagonal matrices. *Linear Algebra and its Applications* 433. 965-983.
[13] Killingbeck J P and Grosjean A 2010 A Gauss elimination method for resonances. *J Math Chem* 47. 1027.
[14] Yunfei Liu, Jun Lv and Xiaowei Gao 2016 "The application of simultaneous elimination and backsstitution method (SEBSM) in finite element method". *Engineering Computations*, 33(8). 2339-2355. https://doi.org/10.1108/EC-10-2015-0287
[15] 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. 69-75.
[16] 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., 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
[17] 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. *Journal Kalam*, 8(1) 039-46
[18] Young D M 1954 Iterative methods for solving partial difference equations of elliptic type. *Trans. Amer. Math. Soc.* 76. 92-111 https://doi.org/10.1090/S0002-9947-1954-0059635-7
[19] Laguna, M. and Marti, R 2005 Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions *Journal of Global Optimization* 33(2). 235-55
[20] Andrei N 2008. An unconstrained optimization test function collection, *Adv. Modeling and Opt.*, 10(1): 147-161.

[21] Jamil M and Yang X 2013 A literature survey of benchmark functions for global optimization problems. *Int. J. Mathematical Modelling and Numerical Opt.* 4(2): 150-194.

[22] 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. Ser.: J. Phys. Conf. Ser.* 1132 012056.