Menemui Matematik
(Discovering Mathematics)

journal homepage: https:/ / myjms.mohe.gov.my/ index.php/ dismath/

Diagonal Newton Method for Solving Large-Scale Linear Systems

Nur Amirah Izzati Mustapa[1], Cynthia Mui Lian Kon[2], Fong Peng Lim[3] and Wah June Leong[4*]

[1, 3, 4]*Department of Mathematics and Statistics, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor.*
[2]*Swinburne University of Technology Sarawak Campus, Jalan Simpang Tiga, 93350 Kuching, Sarawak.*

[1]201118@student.upm.edu.my, [2]ckon@swinburne.edu.my, [3]fongpeng@upm.edu.my, [4]leongwj@upm.edu.my
*Corresponding author

Received: 2 July 2024
Accepted: 24 October 2024

ABSTRACT

This paper presents the Diagonal Newton (DN) method, an optimization-based approach for efficiently solving large-scale linear systems. By approximating the Hessian matrix with its diagonal elements, the method reduces computational complexity while retaining essential second-order information for optimization. We integrate the Armijo Line Search to ensure a sufficient decrease in the objective function during iterations, thus enhancing convergence. The study involves generating large-scale linear systems, converting them into matrix form, and minimizing the corresponding objective function. Through numerical tests, the efficiency of the DN method is evaluated by analyzing function values and final solutions. The results demonstrate the method's effectiveness in solving large-scale linear systems, offering a promising alternative for computational optimization.

Keywords: Large-scale linear system, unconstrained optimization, diagonal Newton method, Armijo line search

INTRODUCTION

Solving large-scale linear systems presents significant challenges, especially in fields like engineering, science, mathematics and statistics (Sim et al. (2019), Al-Hakeem et al. (2023), Lim et al. (2023)), and economics (Sim et al. (2018, 2023). These problems are fundamental in numerical computation and form the basis for many algorithms in scientific computing and machine learning (Sim et al., 2022). Linear systems can be solved using direct or iterative methods. Direct methods, such as Cramer's rule and Gaussian Elimination, provide solutions in a fixed number of steps and work well for small systems. On the other hand, iterative methods, like Newton's method (Waziri et al., 2012), approximate the solution gradually and are more suitable for large-scale systems.

For small-scale problems, techniques such as LU Decomposition, Eigenvalue Decomposition (EVD), and Singular Value Decomposition (SVD) are efficient. However, large-scale systems, particularly in optimization, become more complex and costly to compute. Methods like Gaussian Elimination may suffer from numerical instability when applied to large systems (Yuan et al., 2010).

Newton's method (Andrei (2019), Eagan et al. (2014), Gordon & Tibshirani (2012)) also known as the Newton-Raphson method, is an effective iterative technique for finding function roots or solving optimization problems. It relies on the idea of linearization and uses the gradient to improve the initial guess iteratively. This method can also be generalized to multivariable systems. Its key property, quadratic convergence, means that the number of correct digits doubles with each iteration, provided certain conditions are met.
For twice continuously differentiable objective function $f: R^n \to R$, Newton method utilizes quadratic approximation based on the first three terms of Taylor series expansion about x_k. In the context of a n increment vector, $\Delta x = x_{k+1} - x_k$, we have:

$$f(x) = f(x_k) + [\nabla f(x_k)]^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 f(x_k)\Delta x, \tag{1.1}$$

where $\nabla f(x_k)$ and $\nabla^2 f(x_k)$ denote the gradient and Hessian, respectively.

Hence, the increment vector, Δx is the solution of the following linear equation if $f(x)$ is minimized.

$$\nabla f(x_k) + \nabla^2 f(x_k)\Delta x = 0, \tag{1.2}$$

which corresponding to

$$\Delta x = -\nabla^2 f(x_k)^{-1} \cdot \nabla f(x_k).$$

Thus, the Newton iterate can be given by

$$x_{k+1} = x_k - \alpha_k \nabla^2 f(x_k)^{-1} \nabla f(x_k), \tag{1.3}$$

where α_k is the step length or step size and always equal to 1 (Leong et al., 2021).

A key disadvantage of the Newton method for large-scale optimization problems is its high computational cost, particularly due to the need to compute and invert the Hessian matrix, where the Hessian matrix can be extremely large and dense, making both its computation and inversion expensive in terms of time and memory.

To address this, the Diagonal Newton (DN) method is developed as a modification of the standard Newton method. This approach simplifies computation, reducing costs while maintaining accuracy in solving large-scale systems. We can then combine the DN method with Armijo Line Search to improve convergence toward the minimum of the objective function, even when the standard Newton method fails (Yagishita & Nakayama, 2024).

DIAGONAL NEWTON METHOD FOR SOLVING LARGE-SCALE LINEAR SYSTEMS

Consider a linear system,

$$Ax = b \tag{2.1}$$

Where $A \in R^{m \times n}$ is the coefficient matrix, $x \in R^n$ is the vector of variables and $b \in R^n$ be the vector of right-hand side constants. Linear systems can be solved by reformulating them as

optimization problems, where the goal is to minimize a corresponding objective function, such as the squared error between the left- and right- hand sides of the equations:

$$\min_x \frac{1}{2}\left|\left|Ax-b\right|\right|_2^2 \tag{2.2}$$

where $\|\cdot\|_2$ denotes the Euclidean norm.

Solving linear systems through optimization approaches, such as least-squares minimization (2.2), can be more efficient for large-scale or complex systems, leveraging iterative methods that converge to a solution without requiring direct inversion of matrices. Moreover, by converting linear systems into optimization problems, one can apply gradient descent or quasi-Newton methods, which offer alternatives to traditional direct solvers, especially when dealing with large matrices or non-square systems.

Note that the least squares problem (2.2) is a convex quadratic optimization problem, as the objective function, which minimizes the sum of squared residuals, is quadratic in nature and has a convex shape:

$$f(x)=\frac{1}{2}\left|\left|Ax-b\right|\right|_2^2=\frac{1}{2}x^TA^TAx-b^TAx+b^Tb, \tag{2.3}$$

where the gradient vector, $\nabla f(x)=A^TAx-b^TA$, and the Hessian matrix, $\nabla^2 f(x)=A^TA$, which is positive (semi-) definite, ensuring that the Hessian is non-negative, and the function has a single global minimum.

In the Diagonal Newton Method, the search direction is similar to that of Newton method, except that the search direction at k −iteration is computed as:

$$d_k=-{D_k}^{-1}\nabla f(x_k), \tag{2.4}$$

where D is the diagonal of the Hessian matrix. By using only the diagonal elements of the Hessian, the method reduces the computational burden while still capturing important second-order information. With the search direction (2.4), the DN method iterates through updates of the form:

$$x_{k+1}=x_k-\alpha_k{D_k}^{-1}\nabla f(x_k),$$

where α_k is the step size, which may be chosen using line search techniques such as the Armijo or backtracking line search to ensure sufficient decrease in the objective function.

Moreover, by using only the diagonal elements of the Hessian, the Diagonal Newton Method simplifies matrix inversion and reduces memory requirements, making it well-suited for large-scale optimization problems.

To link D with the Hessian matrix, the following result can be stated;

Theorem 1: Least Change Diagonal Form of the Hessian Matrix.
Let $D=diag\left(\frac{\partial^2 f}{\partial {x_i}^2}\right)$, where $i=1,2,\dots,n$. Then D is the unique solution of the following minimization problem:

$$\min_D \ \|\nabla^2 f - D\|_F^2,$$

where $\|\cdot\|_F^2$ is Frobenius norm.

Proof:

For an $n \times n$ Hessian matrix:

$$\nabla^2 f = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

and a diagonal matrix D with diagonal elements $d_{11}, d_{22}, \ldots, d_{nn}$;

$$D = \begin{bmatrix} d_{11} & 0 & \cdots & 0 \\ 0 & d_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_{nn} \end{bmatrix}.$$

The Frobenius norm of the difference is given by

$$F(D) = \|\nabla^2 f - D\|_F^2 = \sum_{i=1}^n \sum_{j=1}^n (a_{ij} - d_{ij})^2, \tag{2.5}$$

where $d_{ij} = 0$ for $i \neq j$. Therefore, it can be simplified to:

$$\|\nabla^2 f - D\|_F^2 = \sum_{i=1}^n (a_{ij} - d_{ij})^2 + \sum_{i \neq j} a_{ij}^2.$$

To obtain the optamility condition, we differentiate with respect to d_{ii} and set the derivatives to zero:

$$\frac{\partial F}{\partial d_{ii}} = 2(a_{ii} - d_{ii}) = 0 \Longrightarrow d_{ii} = a_{ii}.$$

Thus, the diagonal matrix D that minimizes the Frobenius norm of the difference is:

$$D = diag(a_{11,} a_{22,} \ldots, a_{nn}) = diag\left(\frac{\partial^2 f}{\partial {x_i}^2}, \ldots, \frac{\partial^2 f}{\partial {x_n}^2}\right). \tag{2.6}$$

□

To ensure convergence of the DN method, the step size a_k is obtained by using the Armijo line search:
Find the largest $0 < a_k < 1$, such that the following inequality holds.

$$f(x_k - a_k {D_k}^{-1} \nabla f_k) \leq f(x_k) + \theta a_k \nabla f(x_k)^T {D_k}^{-1} \nabla f(x_k), \tag{2.7}$$

for a chosen constant $\theta \in (0,1)$.

We can now state the detailed algorithm of DN method:

DN Algorithm:

- STEP 1: Set $k = 0$. Choose an initial guess x_0.
- STEP 2: Compute D by (2.6).
- STEP 3: Compute $f(x_k), \nabla f(x_k)$, and $\|\nabla f(x_k)\|$.
- STEP 4: Determine appropriate step size, a_k using conditions that satisfy Armijo Line Search (2.7).
- STEP 5: Calculate $x_{k+1} = x_k - a_k D^{-1} \nabla f(x_k)$.
- STEP 6: Check if $|f(x_{k+1})| > 0.0001$ or $k < 50$.
- STEP 7: If yes, then update$k := k + 1$ and repeat STEP 4. Else, terminate the iteration.

Theorem 2. Let the objective function, f be defined as (2.3). The DN algorithm, combined with Armijo Line Search, converges to a minimizer, x^* of (2.3), i.e.

$$\lim_{k\to\infty} \|\nabla f(x_k)\|^2 = 0.$$

Proof. Firstly, note that (2.3) is a convex quadratic function with a unique minimizer x^*. Since (2.3) is bounded below, we have

$$\lim_{k\to\infty} f\left(x_k - \alpha_k {D_k}^{-1} \nabla f_k\right) - f(x_k) \leq \theta \lim_{k\to\infty} \nabla f(x_k)^T \, {D_k}^{-1} \nabla f(x_k)$$

For some constant $\theta \in (0,1)$ and $\alpha_k \leq 1$. Given $D_k = diag(A^T A)$. Then, the boundedness of f implies that

$$0 = \lim_{k\to\infty} f\left(x_k - \alpha_k {D_k}^{-1} \nabla f_k\right) - f(x_k) \leq \theta \, [Tr(A^T A)]^{-1} \lim_{k\to\infty} \|\nabla f(x_k)\|^2,$$

which also gives

$$\lim_{k\to\infty} \|\nabla f(x_k)\|^2 = 0.$$

RESULTS AND DISCUSSION

Conditions for Numerical Experiment

In our numerical experiments, we utilized a HP Laptop modelled 15s-eq0xxx with 4.00 GB installed Random Access Memory (RAM) which able to manage in using multiple software operations at the same time. Moreover, it also has processor of AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx that has ability to compute numerical results achieved from MATLAB R2024a software.

Numerical Examples

Two examples with randomly generated A and b are given as follow:

Example 1: The matrix A is generated by using the command $randi(10,30,30)$ and matrix B is created by using the command $randi(10,30,1)$ in the MATLAB software. The $randi(10,30,30)$ will generate a 30 by 30 sized matrix A with random integers between 1 and 10 while $randi(10,30,1)$ will generate 30 by 1 sized matrix B with random integers between 1 and 10:

$$A = \begin{bmatrix} 5 & 1 & 2 & 2 & 1 & \cdots & 5 & 5 & 1 & 2 & 1 \\ 8 & 5 & 5 & 10 & 1 & \cdots & 2 & 10 & 5 & 4 & 2 \\ 1 & 10 & 7 & 5 & 1 & \cdots & 10 & 4 & 10 & 3 & 5 \\ 4 & 6 & 5 & 6 & 3 & \cdots & 9 & 10 & 4 & 9 & 2 \\ 2 & 7 & 1 & 5 & 9 & \cdots & 5 & 7 & 8 & 9 & 7 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 9 & 1 & 9 & 8 & 2 & \cdots & 2 & 10 & 9 & 8 & 4 \\ 1 & 6 & 5 & 8 & 3 & \cdots & 4 & 3 & 2 & 6 & 8 \\ 1 & 2 & 10 & 10 & 8 & \cdots & 2 & 7 & 5 & 10 & 6 \\ 2 & 6 & 7 & 8 & 6 & \cdots & 5 & 7 & 5 & 1 & 9 \\ 9 & 7 & 7 & 2 & 1 & \cdots & 3 & 3 & 5 & 10 & 4 \end{bmatrix}_{30\times 30}, B = \begin{bmatrix} 3 \\ 10 \\ 1 \\ 10 \\ 6 \\ \vdots \\ 5 \\ 9 \\ 6 \\ 5 \\ 3 \end{bmatrix}_{30\times 1}$$

Example 2: The matrix A is generated by using the command $randi([-10,10],30)$ and matrix B is created by using the command $randi([-10,10],30,1)$ in the MATLAB software. The $randi([-10,10],30)$ will generate a 30 by 30 sized matrix A with random integers between -10 and 10 while $randi([-10,10],30,1)$ will generate a 30 by 1 sized matrix B with random integers between -10 and 10. After that, we save the current state of the random number generator of matrix A and B using the command $rng(0)$ in the MATLAB software. This command tends to set the random seed for reproducibility.

$$A = \begin{bmatrix} 7 & 4 & 5 & -9 & -8 & \cdots & -1 & 4 & 5 & -8 & -9 \\ 9 & -10 & -5 & -9 & 10 & \cdots & -8 & 1 & -6 & 6 & -10 \\ -8 & -5 & 0 & 1 & -10 & \cdots & 10 & -2 & 5 & -4 & -2 \\ 9 & -10 & 4 & 6 & 6 & \cdots & -4 & -9 & 10 & 2 & 3 \\ 3 & -8 & 8 & 9 & 7 & \cdots & -4 & 6 & 8 & 10 & 5 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 5 & 0 & -4 & 7 & -9 & \cdots & 1 & 3 & -6 & -4 & 4 \\ 5 & 10 & 5 & 1 & -5 & \cdots & 5 & -4 & 1 & 6 & 1 \\ -2 & -3 & 5 & 10 & -8 & \cdots & 4 & 3 & -9 & 4 & -4 \\ 3 & 2 & -3 & -9 & -7 & \cdots & 6 & 5 & -2 & -8 & -7 \\ -7 & -6 & 1 & -1 & -5 & \cdots & -4 & 2 & -8 & -8 & 3 \end{bmatrix}_{30\times 30}, B = \begin{bmatrix} 10 \\ -7 \\ -5 \\ -2 \\ -9 \\ \vdots \\ -3 \\ 1 \\ 5 \\ -2 \\ -1 \end{bmatrix}_{30\times 1}$$

Initial guess, $x_0 = 0$ is used for both examples.

Table 1: Example 1: 50 Iterations

Iteration, k	**Step size, a_k**	**Function value, $f(x_k)$**	**Gradient norm, $\|\nabla f(x_k)\|$**
0	0.0625	1173.000000	10207.737262
10	0.0625	195.871750	793.726477
20	0.1250	152.703764	343.164587
30	0.1250	128.140530	296.161242
40	0.1250	112.766115	273.830764
50	0.0625	102.475818	309.212148

Table 2: Example 2: 50 Iterations

Iteration, k	Step size, a_k	Function value, $f(x_k)$	Gradient norm, $\|\nabla f(x_k)\|$
0	1.00	748.000000	1243.300446
10	0.50	255.942120	365.385963
20	1.00	188.134151	180.055280
30	0.50	153.867710	245.021479
40	0.50	126.482128	129.039203
50	1.00	108.422822	106.089494

The results from the test problem, shown in the two tables, highlight the performance of the Diagonal Approximate Newton method applied to solve large-scale linear systems. In Tables 1 and 2, the method iterates 50 times for each optimization problem. The step size alternates between 0.0625 and 0.1250 in the Table 1, and between 1.00 and 0.50 in Table 2. This alternating step size pattern, determined by the Armijo line search, helps balance larger and smaller steps, ensuring stable convergence.

Furthermore, the gradient norm starts at very high values and decreases steadily in Table 1 and 2. These indicates that the method's effectiveness in reducing the steepness of the objective function f. Besides, the gradient norm continues continues to reduce through the iterations, shows that the approach's ability in reducing the gradient and approach a stationary point. Thus, the decreases in the function value align with the gradient norm reductions, proving the approach in showing effective minimization of the objective function f.

CONCLUSION

In conclusion, the DN method proves to be an efficient and practical approach for solving large-scale linear systems. By approximating the Hessian matrix using only its diagonal elements, the method significantly reduces computational complexity, making it well-suited for large-scale problems where direct computation of the full Hessian would be impractical. The use of Armijo line search to determine an appropriate step size ensures stable convergence by balancing step length, allowing the method to maintain sufficient descent in each iteration. The method has demonstrated its capability to minimize the objective function effectively, with consistent convergence toward the optimal solution. The results highlight that the DN method can achieve reliable performance in scenarios where traditional methods might struggle due to high computational costs or instability. Overall, this method offers a valuable tool for large-scale optimization problems, providing both efficiency and accuracy in solving linear systems.

REFERENCES

Andrei, N. (2019). A diagonal quasi-Newton updating method for unconstrained optimization. *Numerical Algorithms*, **81(2)**: 575-590.

Al-Hakeem, H. A., Arasan, J., Mustafa, M. S. B., and Peng, L. F. (2023), Parameter Estimation for the Generalized Exponential Distribution in the Presence of Interval Censored Data and Covariate. *Int. J. Nonlinear Anal. Appl.*, **14(1)**: 739–751.

Eagan, N., Hauser, G., & Flaherty, T. (2014). Newton's Method on a System of Nonlinear Equations. Pittsburgh: Carnegie Mellon University.

Gordon, G., & Tibshirani, R. (2012). Gradient descent revisited. *Optimization*, **10**: 1-31.

Leong, W. J., Enshaei, S. & Kek, S. L. (2021). Diagonal quasi-Newton methods via least change updating principle with weighted Frobenius norm. *Numer Algor*, **86**: 1225-1241.

Lim, F. P., Wong, L. L., Yap, H. K., and Yow, K. S. (2023), Identifying Outlier Subjects in Bioavailability Trials Using Generalized Studentized Residuals. *Sains Malays.*, **52(5)**: 1581–1593.

Sim, H. S., Leong, W. J., Chen, C. Y., & Ibrahim, S. N. I. (2018). Multi-step spectral gradient methods with modified weak secant relation for large scale unconstrained optimization. *Numerical Algebra, Control and Optimization*, **8(3)**: 377-387.

Sim, H. S., Leong, W. J., & Chen, C. Y. (2019). Gradient method with multiple damping for large-scale unconstrained optimization. *Optimization Letters,* **13(1)**: 617-632.

Sim, H. S., Chen, C. Y., Leong, W. J., & Li, J. (2022). Nonmonotone spectral gradient method based on memoryless symmetric rank-one update for large-scale unconstrained optimization. *Journal of Industrial & Management Optimization,* **18(6)**: 3975-3988.

Sim, H. S., Ling, W. S. Y., Leong, W. J., & Chen, C. Y. (2023). Proximal linearized method for sparse equity portfolio optimization with minimum transaction cost. *Journal of Inequalities and Applications*, **2023(1)**: 152.

Waziri, M. Y., Leong, W. J., & Hassan, M. A. (2012). Diagonal Broyden-like method for large-scale systems of nonlinear equations. *Malaysian Journal of Mathematical Sciences*, **6(1)**: 59-73.

Yuan, G., Lu, S., & Wei, Z. (2010). A line search algorithm for unconstrained optimization. *Journal of Software Engineering and Applications*, **3(05)**: 503.

Yagishita, S., & Nakayama, S. (2024). An acceleration of proximal diagonal Newton method. *JSIAM Letters*, **16**: 5-8.