GRADIENT BASED ITERATIVE ALGORITHM WITH A SEQUENCE OF OPTIMAL CONVERGENT FACTORS FOR SOLVING LINEAR SYSTEMS

ADISORN KITTISOPAPORN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN APPLIED MATHEMATICS
DEPARTMENT OF MATHEMATICS FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2019
KMITL-2019-SC-M-001-039

GRADIENT BASED ITERATIVE ALGORITHM WITH A SEQUENCE OF OPTIMAL CONVERGENT FACTORS FOR SOLVING LINEAR SYSTEMS

ADISORN KITTISOPAPORN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN APPLIED MATHEMATICS

DEPARTMENT OF MATHEMATICS FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2019
KMITL-2019-SC-M-001-039

Thesis Title	Gradient based iterative algorithm with a sequence of optimal convergent factors for solving linear systems
Student Name	Mr. Adisorn Kittisopaporn
Student ID	60605014
Degree	Master of Science (Applied Mathematics)
Department	Mathematics
Year	2019
Thesis Advisor	Asst.Prof.Dr. Pattrawut Chansangiam

Abstract

In this research, we introduce a new iterative algorithm for solving systems of linear equations. The algorithm is based on a gradient and along with a sequence of optimal convergent factors. We show that the algorithm is applicable with any initial vector as long as the coefficient matrix is invertible. Then we analyse the convergent rate and error estimates. Furthermore, we demonstrate the numerical simulations comparing to well-known methods and recent methods.

Keywords : gradient, steepest descent, iterative algorithm, norm, linear system

Acknowledgements

I would like to express my special thanks of gratitude to all of those whom made this thesis becomes a reality with their kind supports and helps.

Foremost, I would like to express my gratitude towards my beloved family for the encouragement and the inspiration which helped me in fulfilment of this thesis. I am extremely thankful to my thesis advisor, Asst.Prof.Dr. Pattrawut Chansangiam, for imparting his knowledge and expertise in this study as well as his patience, motivation, and endless support. I am also appreciated and thankful to my thesis committees: Assoc.Prof.Dr. Jessada Tariboon and Asst.Prof.Dr. Nopparat Pochai. by providing the valuable guidance and suggestions during this thesis work.

Finally, I would not forget Miss. Rungpailin Kongyaksee for her encouragement and more over for her timely support till the completion of my thesis work.

Mr. Adisorn Kittisopaporn

Table of Contents

List of Tables

Table **Page**

List of Figures

Figure **Page**

Chapter 1

Introduction

1.1 Research Motivation

In mathematics, a linear system (or system of linear equations) is a collection of two or more linear equations involving the same set of variables. For example,

$$x_1 + 2x_2 + 4x_3 = 15$$
$$-3x_1 + x_2 + 5x_3 = 20$$
$$2x_1 + 3x_2 + x_3 = 7$$

is a system of three linear equations in three variables x_1, x_2, x_3. A solution to a linear system is an assignment of a value to each variable for which all the equations are simultaneously satisfied. The solution to the system above is given by

$$x_1 = -1, \quad x_2 = 2, \quad x_3 = 3$$

since it makes all three equations valid collectively.

The theory of linear systems is a basis and a fundamental part of linear algebra, a subject which is used in most areas of modern mathematics. The computational algorithms for finding solutions are an important part of numerical linear algebra, and play prominent roles in engineering, physics, chemistry, computer science, and also economics.

In computation, simultaneous equations must be transformed into matrix and vector forms and considered as a linear system $Ax = b$ where A is a coefficient matrix and b is a known constant vector. Obviously, such system has a unique solution if and only if A is invertible. The methods of solving the linear systems can be classified into three groups: direct methods, semi-direct methods, and iterative methods.

Direct methods are a way to solve for an exact solution. Most of direct methods are well-known in general linear algebra such as Gaussian Elimination, Inversion Matrix, Cramer's Rule, and several decompositions of matrix. Direct methods are advantage in availability for any invertible matrices except for a decomposition method which has to be considered at different types of matrices. A disadvantage of direct methods is that the solution occurs at the final step of computation. In other word, we must finalize the entire process in order to obtain the solution. Stopping at somewhere during the process does not offer even a numerical solution. In addition, miscalculation in some steps might have an effect to every step afterwards and the outcome solution eventually absolutely differ from the exact solution. Also, large linear systems require huge spaces of memory. Therefore, direct methods are suited for small linear systems.

Semi-direct methods are other ways to seek the exact solution from numerical solutions with an exact step number. At each iteration, the numerical solution gets closer and closer to the exact solution and the exact solution occurs at the final iteration. The important semi-direct method is the conjugate gradient method.

Iterative methods are numerical methods that create a sequence of numerical solutions which converges to the exact solution. By utilizing knowledge from linear algebra together with matrix analysis such as norms of vector and matrix, convergence of sequence of vectors and matrices, etc. The advantage of iterative methods is having less steps of computation but the result solution is considerably close to the exact solution. Moreover, the number of iteration is able to be calculated for which the iteration can stop when the approximated solution has reached the satisfactory error. Above all, the iterative methods are suited for large linear systems. However, each method guarantees the convergence of solutions for different kinds of the coefficient matrix. Examples of well-known iterative methods:

- Jacobi method,
- Gauss-Siedel (GS) method,
- Successive Over-Relaxation (SOR) method.

Also the methods that developed from SOR such as:

- Extrapolated Successive Over-Relaxation (ESOR) method,
- Jacobi Over-Relaxation (JOR) method,
- Accelarated Over-Relaxation (AOR) method.

In this research, we shall introduce a new iterative method for solving linear system. Our method is an enhancement of a gradient based iterative method. With a sequence of optimal convergent factors, we may fasten the rate of convergence and reduce the iteration numbers. Furthermore, without any conditions of coefficient matrix and initial vector, the sequence of approximate solutions converges to the exact solution.

1.2 Objectives of the study

1) Create a new gradient based iterative algorithm for solving linear systems.
2) Propose the sequence of optimal convergent factors that make the approximated solutions converge to the exact solution.
3) Analyze the convergence of the algorithm in order to obtain a convergent rate and error estimate.

4) Provide numerical simulations comparing to well-known algorithms and recent algorithms.

1.3 Scopes of the study

Consider the linear system

$$Ax = b$$

where A is an nxn invertible real matrix and b is a $n \times 1$ real constant vector. Our algorithm is in the form

$$x(k+1) = x(k) - \tau_{k+1}\nabla f(x(k))$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and is defined to be

$$f(x) := \frac{1}{2}\|Ax - b\|_F^2$$

also τ_k is an appropriate step size at each k iteration.

1.4 Benefits of the Study

Attain a new effective algorithm to solve linear systems.

1.5 Research methodology

1) Study advanced topics in Applied Linear Algebra and iterative methods.
2) Study advanced topics in Functional Analysis and Matrix Analysis.
3) Study topics of iterative algorithms based on gradients from research papers.
4) Propose a new gradient based iterative method together with a sequence of optimal convergence factor.
5) Analyze a convergence of the algorithm.
6) Provide numerical simulations.
7) Combine and summarize all findings then write the thesis and make suggestions for further studies.

Table 1.1*:* The research schedule

Activity	Time frame							
	2017		2018				2019	
	Aug.-Sep.	Oct.-Dec.	Jan.-Mar.	Apr.-Jun.	Jul.-Sep.	Oct.-Dec.	Jan.-Mar.	Apr.-Jun.
Step 1	⟷	⟷						
Step 2		⟷	⟷					
Step 3				⟷	⟷			
Step 4					⟷			
Step 5						⟷		
Step 6							⟷	
Step 7								⟷

Chapter 2

Preliminaries

In this chapter, we provide sufficiently some knowledge and tools relative to matrix analysis, functional analysis, convex function and linear iterative systems. We denote the set of m-by-n real matrices by $M_{m,n}(\mathbb{R})$ and we denote M_n for square matrices instead of $M_{n,n}$.

2.1 Elements in matrix and functional analysis

Definition 2.1. Let $A \in M_n(\mathbb{R})$. The *trace* of A is denoted by $\mathrm{tr}(A)$ and defined to be

$$\mathrm{tr}(A) = \sum_{i=1}^{n} a_{ii}.$$

Lemma 2.2. (see e.g.[14]) Let A, B, C, D be compatibly constant matrices, X be a variable matrix and $c \in \mathbb{R}$. The properties and derivatives of trace of matrix are as follows:

1. $\mathrm{tr}(A+B) = \mathrm{tr}(A) + \mathrm{tr}(B)$,
2. $\mathrm{tr}(cA) = c\mathrm{tr}(A)$,
3. $\mathrm{tr}(A) = \mathrm{tr}(A^T)$,
4. $\mathrm{tr}(AB) = \mathrm{tr}(BA)$,
5. $\frac{d\mathrm{tr}(AX)}{dX} = \frac{d\mathrm{tr}(X^T A^T)}{X} = A^T$,
6. $\frac{d\mathrm{tr}(XAX^T B)}{dX} = BXA + B^T X A^T$.

Definition 2.3. The *condition number* of an $m \times n$ matrix A is the ratio between its largest and smallest singular value:

$$\kappa(A) = \left(\frac{\lambda_{\max}(A^T A)}{\lambda_{\min}(A^T A)} \right)^{1/2}.$$

Definition 2.4. Let V be a vector space over the field $\mathbb{F}(\mathbb{F} = \mathbb{R}$ or $\mathbb{C})$. A function $\|\cdot\| : V \to \mathbb{R}$ is a norm if, for all $x, y \in V$ and all $c \in \mathbb{F}$,

1. $\|x\| \geqslant 0$
2. $\|x\| = 0$ if and only if $x = 0$
3. $\|cx\| = |c|\|x\|$
4. $\|x+y\| \leqslant \|x\| + \|y\|$

Definition 2.5. The *Frobenius norm* is a norm of an $m \times n$ matrix A defined as the square root of the matrix trace of AA^T:

$$\|A\|_F = \sqrt{\mathrm{tr}(AA^T)}.$$

Definition 2.6. The *Spectral norm* is a norm of an $m \times n$ matrix A defined as the square root of the largest singular value of A:

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)}.$$

Definition 2.7. Let $A \in M_n(\mathbb{R})$ be symmetric. Then A is called

- **Negative definite** if $x^T Ax < 0$ for all $x \in \mathbb{R}^n - \{0\}$,
- **Positive definite** if $x^T Ax > 0$ for all $x \in \mathbb{R}^n - \{0\}$,
- **Positvie semidefinite** if $x^T Ax \geqslant 0$ for all $x \in \mathbb{R}^n$.

Lemma 2.8. Let A be a symmetric matrix and let us denote $\lambda_{\min}$ ($\lambda_{\max}$, resp.) its smallest (largest, resp.) eigenvalue, then

$$\lambda_{\min} x^T x \leqslant x^T Ax \leqslant \lambda_{\max} x^T x, \ x \in \mathbb{R}^n.$$

Definition 2.9. The Löwner partial order $\preceq$ for real symmetric matrices is defined by

$$A \preceq B \Leftrightarrow B - A \text{ is positive semi-definite.}$$

Using the definition of positive semi-definite matrices, this can be reformulated as:

$$A \preceq B \Leftrightarrow x^T Ax \leqslant x^T Bx, \ x \in \mathbb{R}^n.$$

2.2 Convex function

Definition 2.10. A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if for all x, y in the domain of f, and θ with $0 \leqslant \theta \leqslant 1$, we have

$$f(\theta x + (1-\theta)y) \leqslant \theta f(x) + (1-\theta) f(y). \tag{2.1}$$

Definition 2.11. A twice-differentiable convex function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *strongly convex* if there exist constant $0 \leqslant m < M$ such that for all $x \in \mathbb{R}^n$,

$$mI \preceq \nabla^2 f(x) \preceq MI. \tag{2.2}$$

Using the definition of the partial order $\preceq$, this is equivalent to

$$my^T y \leqslant y^T \nabla^2 f(x) y \leqslant My^T y, \ x \in \mathbb{R}^n, y \in \mathbb{R}^n. \tag{2.3}$$

In other words, m (M, resp.) is a lower (upper, resp.) bound on the smallest (largest, resp.) eigenvalue of $\nabla^2 f(x)$ for all $x \in \mathbb{R}^n$.

Lemma 2.12. (see e.g.[15]) If f is strongly convex on $\mathbb{R}^n$, then for $x, y \in \mathbb{R}^n$ we have

$$f(y) \geqslant f(x) + \nabla f(x)^T (y - x) + \frac{m}{2} \|y - x\|_F^2, \tag{2.4}$$

$$f(y) \leqslant f(x) + \nabla f(x)^T (y - x) + \frac{M}{2} \|y - x\|_F^2. \tag{2.5}$$

2.3 Linear Iterative Systems

We consider the linear system

$$Ax = b \tag{2.6}$$

where $A \in M_n(\mathbb{R})$ is an invertible matrix, $b \in \mathbb{R}^n$ is a known constant vector, and $x \in \mathbb{R}^n$ is an unknown vector to be solved. Indeed the equation has the exact solution $x^* = A^{-1}b$.

We shall be attempting to solve (2.6) by replacing it with a form of iterative system

$$x(k+1) = Tx(k) + c, \quad x(0) = x_0$$

in which T is an $n \times n$ matrix, and called an *iteration matrix*, and c is a vector.

Next, we will turn our attention to the three most classical iterative methods, Jacobi method, Gauss-Seidel method and SOR method, also, the methods developed from SOR method such as ESOR, JOR, and AOR.

2.3.1 Jacobi method

Jacobi method was created and named after the influential nineteenth century German analyst Carl Jacobi. It is the iterative method to find the solution of linear system $Ax = b$ under condition that each entry in the main diagonal of coefficient A must not be zero.

It is instructive to rederive the Jacobi method in a direct matrix form. We begin by decomposing the coefficient matrix

$$A = D - L - U$$

into the sum of a strictly lower triangular matrix L, a diagonal matrix D, and a strictly upper triangular matrix U, each of which is uniquely specified. Now, we have the Jacobi iteration matrix as follows:

$$T_J = D^{-1}(L + U),$$
$$c_J = D^{-1}b.$$

The Jacobi method is guaranteed to converge if the coefficient matrix A possesses some good properties.

Definition 2.13. A square matrix A is called *strictly diagonally dominant* if

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad \text{for all } i = 1, \ldots, n.$$

In other words, strict diagonal dominance requires each diagonal entry to be larger, in absolute value, than the sum of the absolute values of all the other entries in its row. For example, the matrix

$$\begin{bmatrix} 3 & -1 & 1 \\ 1 & -4 & 2 \\ -2 & -1 & 5 \end{bmatrix}$$

is strictly diagonally dominant since

$$|3| > |-1| + |1|, \ |-4| > |1| + |2|, \ |5| > |-2| + |-1|.$$

Diagonally dominant matrices appear frequently in numerical solution methods for both ordinary and partial differential equations. As we shall see, they are the most common class of matrices to which iteration solution methods can be successfully applied.

Theorem 2.14. [2] If A is strictly diagonally dominant, then the associated Jacobi iteration method converges for any choice of initial approximate vector $x(0)$.

2.3.2 Gauss-Seidel (GS) method

Gauss-Siedel method, also known as the Liebmann method or the method of successive displacement, was named after the German mathematicians Carl Friedrich Gauss and Philipp Ludwig Von Seidel, and is similar to the Jacobi method. It was only mentioned in a private letter from Gauss to his student Gerling in 1823. A publication was not delivered before 1874 by Seidel.

The Gauss-Seidel iteration matrix is given as:

$$T_{GS} = (D - L)^{-1}U,$$
$$c_{GS} = (D - L)^{-1}b.$$

Theorem 2.15. [2] If A is strictly diagonally dominant, then the associated Gauss-Seidel iteration method converges for any choice of initial approximate vector $x(0)$.

2.3.3 Successive Over-Relaxation (SOR) method

The method of successive over-relaxation (SOR) is a variant of the Gauss-Seidel method for solving a system of linear equations. By using the technique of relaxation, it is resulting in faster convergence. It was devised simultaneously by David M. Young, Jr. and by Stanley P. Frankel in 1950 for the purpose of automatically solving linear systems on digital computers.

In addition to the idea of Gauss-Seidel method, SOR has added a new variable

ω called *weighted factor*. The SOR iteration matrix is viewed as:

$$T_{SOR} = (D - \omega L)^{-1} (\omega U + (1 - \omega) D),$$
$$c_{SOR} = (D - \omega L)^{-1} b.$$

The following theorems are used to consider whether the SOR method converges or not.

Theorem 2.16. [16] If A is a positive definite matrix and $0 < \omega < 2$, then the SOR method converges for any choice of initial approximate vector $x(0)$.

Theorem 2.17. [17] Suppose $a_{ii} \neq 0$, for each $i = 1, 2, \ldots, n$, then the SOR method can converge for any choice of initial approximate vector $x(0)$ if $0 < \omega < 2$.

2.3.4 Extrapolated Successive Over-Relaxation (ESOR) method

As far as the iterative methods are concerned, it is not possible to indicate the best general purpose iteration method for the solution of linear systems (in the sense of converging faster than any other methods). However, trying to find methods that are best in the sense of being at least convergent for a maximum set of problems. Consequently, the Extrapolated Successive Over-Relaxation (ESOR) was introduced. The ESOR is particularly useful compared to the SOR if the SOR diverges or its optimum relaxed parameter ω cannot be determined but even if ω is known the ESOR may be faster than the SOR method. The iteration matrix of ESOR is given by:

$$T_{ESOR} = (D - \omega L)^{-1} ((\tau - \omega) L + \tau U + (1 - \tau) D),$$
$$c_{ESOR} = \tau (D - \omega L)^{-1} b.$$

Theorem 2.18. [5] For properly chosen $\omega \in (0, 2)$ and $\tau = \tau(\omega)$ with $|\tau| \leqslant \omega$, the ESOR method converges for any choice of initial approximate vector $x(0)$ if and only if each eigenvalue λ of T_{ESOR} satisfy either Re $\lambda < 1$ or Re $\lambda > 1$.

2.3.5 Jacobi Over-Relaxation (JOR) method

In 1971, David M. Young has introduced a generalization of the Jacobi method, that is, the Jacobi over-relaxation (or JOR), in which, having introduced a relaxation parameter ω. The JOR iteration matrix is as follows:

$$T_{JOR} = D^{-1} (\omega L + \omega U + (1 - \omega) D),$$
$$c_{JOR} = \omega D^{-1} b.$$

This method can be reduced to Jacobi method when $\omega = 1$. Convergence results for JOR are as follows:

Theorem 2.19. [6] If A is a symmetric positive definite matrix, then the JOR method is convergent for any choice of initial approximate vector $x(0)$ if $0 < \omega < \rho(D^{-1}A)$.

Theorem 2.20. [6] If the Jacobi method is convergent, then the JOR method converges if $0 < \omega \leqslant 1$.

2.3.6 Accelerated Over-Relaxation (AOR) method

The Accelerated Over-Relaxation (AOR) method is a two-parameter iterative method which is a generalization of the conventional Jacobi, Gauss-Seidel, and SOR techniques. AOR method was introduced by Hadjidimos in 1978. The iteration matrix of AOR is viewed as:

$$T_{AOR} = (D + \alpha L)^{-1} \left((\alpha - \beta)L - \beta U + (1 - \beta)D\right),$$
$$c_{AOR} = \beta(D + \alpha L)^{-1} b.$$

The AOR method contains the following standards as special cases:

- Jacobi method : $\alpha = 0$ and $\beta = 1$,
- Gauss-Seidel method : $\alpha = 1$ and $\beta = 1$,
- JOR method : $\alpha = 0$,
- SOR method : $\alpha = \beta$.

Theorem 2.21. [7] AOR iterations are convergent only if $0 \leqslant \alpha < 2$ and $0 < \beta < 2$.

2.3.7 Barzilai and Borwein (BB) method

Another study of solving linear systems is in the field of unconstrained convex optimization where the gradient method searches along with the steepest descent is used.

Suppose we would like to minimize a continuously differentiable function f on $\mathbb{R}^n$. To do this, let $x(k)$ be the current iterate point, and $g_k = g(x(k)) = \nabla f(x(k))$ be the gradient vector at $x(k)$. The steepest descent method defines the next iteration by

$$x(k+1) = x(k) - \alpha_k g_k$$

where $\alpha_k > 0$ satisfies

$$f\left(x(k) - \alpha_k g_k\right) = \min_{\alpha > 0} f\left(x(k) - \alpha g_k\right).$$

In 1988, Barzilai and Borwein [19] approached their step size in the current iteration. For the iterative equation $x(k+1) = x(k) - \alpha_k g_k$, the step size α_k can be chosen either (2.7) or (2.8) as follows:

$$\alpha_k = \frac{S_{k-1}^T y_{k-1}}{\|y_{k-1}\|_2^2}, \tag{2.7}$$

$$\alpha_k = \frac{\|S_{k-1}\|_2^2}{S_{k-1}^T y_{k-1}} \tag{2.8}$$

where $S_{k-1} = x(k) - x(k-1)$ and $y_{k-1} = g_k - g_{k-1}$.

Definition 2.22. Let $\{x_k\}$ converges to x^*. We say that the convergence is of order $q \geqslant 1$ and with factor $\gamma > 0$, if $\exists k_0$ such that $\forall k \geqslant k_0$,

$$\|x_{k+1} - x^*\| \leqslant \gamma \|x_k - x^*\|^q.$$

Theorem 2.23. [19] Let $f : \mathbb{R}^n \to \mathbb{R}$ be a strictly convex quadratic function defined by

$$f(k) := x^T Ax + b^T x,$$

where A is a symmetric positive definite $n \times n$ matrix and $b \in \mathbb{R}^n$. The gradient method with BB step size (2.7) and (2.8) almost always converges R-superlinearly, i.e., the order of convergence $q > 1$.

2.3.8 Gradient and least-squares based iterative (GI and LS) methods

In the recent decade, many researchers have been developing gradient based iterative algorithms for linear matrix equations based on the techniques of hierarchical identification and minimization of associated norm-error functions. See e.g. Feng and Tongwen [9], [10], [11], and [12] and Feng et al. [13].

We recall the following algorithms:

Theorem 2.24. [9] Suppose that the linear system (2.6) has a unique solution x^*. Let $0 < \mu < \frac{2}{\|A\|_2^2}$ or $0 < \mu < \frac{2}{\|A\|_F^2}$. Then the iterative solution $x(k)$ given by the following gradient based iterative (GI) algorithm:

$$x(k+1) = x(k) + \mu A^T (b - Ax(k))$$

converges to x^* for any initial value $x(0)$.

Theorem 2.25. [9] Suppose the system (2.6) has a unique solution x^*. Let $0 < \mu < 2$. Then the iterative solution $x(k)$ given by the following least-squares based iterative (LS) algorithm

$$x(k+1) = x(k) + \mu \left(A^T A\right)^{-1} A^T (b - Ax(k))$$

leads to $\lim_{k\to\infty} x(k) = x^*$ for any initial value $x(0)$.

Chapter 3

Proposing the gradient based iterative algorithm with a sequence of optimal convergent factors for solving linear systems

In this chapter, we introduce a new method for solving linear systems based on a gradient and we provide an appropriate sequence of convergent factors which minimizes an error at each iteration.

Let us turn our attention to the linear system

$$Ax = b \tag{3.1}$$

where $A \in M_n(\mathbb{R})$ is a nonsingular square matrix, $b \in \mathbb{R}^n$ is a known constant vector, and $x \in \mathbb{R}^n$ is an unknown vector to be solved. Indeed, the equation has the exact solution $x^* = A^{-1}b$.

We firstly define the quadratic norm-error function $f : \mathbb{R}^n \to \mathbb{R}$ by

$$f(x) := \frac{1}{2}\|Ax - b\|_F^2 \quad \text{where } \|x\|_F^2 = \mathrm{tr}(xx^T). \tag{3.2}$$

We assume that the consistent system (3.1) has a unique solution and therefore, an optimal vector x^* of f exists. We shall start by having an arbitrary initial vector $x(0)$ and then at every step $k > 0$ we iteratively move to the next vector $x(k+1)$ with an appropriate direction, i.e., the negative gradient of f, together with a suitable step size τ_{k+1}. The gradient based iterative method thus can be described through the following recursive rule:

$$x(k+1) = x(k) - \tau_{k+1}\nabla f(x(k)). \tag{3.3}$$

3.1 Searching a direction

We shall minimize the function f by applying Lemma 2.2, we then obtain

$$\begin{aligned}
\nabla f(x) = \frac{df(x)}{dx} &= \frac{1}{2}\frac{d}{dx}\mathrm{tr}\left((Ax-b)(Ax-b)^T\right) \\
&= \frac{1}{2}\frac{d}{dx}\mathrm{tr}\left(Axx^TA^T - bx^TA^T - Axb^T + bb^T\right) \\
&= \frac{1}{2}\left(\frac{d}{dx}\mathrm{tr}(x^TA^TAx) - \frac{d}{dx}\mathrm{tr}(x^TA^Tb) - \frac{d}{dx}\mathrm{tr}(b^TAx) + \frac{d}{dx}\mathrm{tr}(bb^T)\right) \\
&= \frac{1}{2}\left(A^TAx + A^TAx - A^Tb - A^Tb\right) \\
&= A^TAx - A^Tb \\
&= A^T(Ax - b).
\end{aligned}$$

Thus, our new iterative equation is in the form

$$x(k+1) = x(k) + \tau_{k+1} A^T \left(b - Ax(k)\right). \tag{3.4}$$

3.2 Choosing a step size

To choose the best step size at each iteration, we minimize an error which occurs with a new vector, $x(k+1)$. Then for each $k \in \mathbb{N} \cup \{0\}$, we define $\phi_{k+1} : [0, \infty) \to \mathbb{R}$ by

$$\phi_{k+1}(\tau) := \frac{1}{2} \| A \left(x(k) + \tau A^T \left(b - Ax(k)\right)\right) - b \|_F^2. \tag{3.5}$$

Now, we shall minimize the function $\phi_{k+1}(\tau)$ by applying Lemma 2.2. Before then, we may transform (3.5) into a convenient form.
Let $\tilde{b} = AA^T(b - Ax(k))$ and $\tilde{c} = -\left(Ax(k) - b\right)$. Then

$$\begin{aligned}
\phi_{k+1}(\tau) &= \frac{1}{2} \| A \left(x(k) + \tau A^T \left(b - Ax(k)\right)\right) - b \|_F^2 \\
&= \frac{1}{2} \| \tau AA^T(b - Ax(k)) + Ax(k) - b \|_F^2 \\
&= \frac{1}{2} \| \tau \tilde{b} - \tilde{c} \|_F^2.
\end{aligned}$$

Consider

$$\begin{aligned}
\frac{d\phi_{k+1}(\tau)}{d\tau} &= \frac{1}{2} \frac{d}{d\tau} \mathrm{tr} \left((\tau \tilde{b} - \tilde{c})(\tau \tilde{b} - \tilde{c})^T \right) \\
&= \frac{1}{2} \frac{d}{d\tau} \mathrm{tr} \left(\tau \tilde{b} \tau \tilde{b}^T - \tau \tilde{b} \tilde{c}^T - \tilde{c} \tau \tilde{b}^T + \tilde{c} \tilde{c}^T \right) \\
&= \frac{1}{2} \left(\frac{d}{d\tau} \tau^2 \mathrm{tr}(\tilde{b} \tilde{b}^T) - \frac{d}{d\tau} \tau \mathrm{tr}(\tilde{b} \tilde{c}^T) - \frac{d}{d\tau} \tau \mathrm{tr}(\tilde{c} \tilde{b}^T) + \frac{d}{d\tau} \mathrm{tr}(\tilde{c} \tilde{c}^T) \right) \\
&= \frac{1}{2} \left(2\tau \mathrm{tr}(\tilde{b} \tilde{b}^T) - 2\mathrm{tr}(\tilde{b} \tilde{c}^T) \right).
\end{aligned}$$

Assume that ϕ_{k+1} does not change with respect to τ, i.e.,

$$\frac{d\phi_{k+1}(\tau)}{d\tau} = 0.$$

This gives $\tau = \frac{\mathrm{tr}(\tilde{b} \tilde{c}^T)}{\mathrm{tr}(\tilde{b} \tilde{b}^T)}$, and hence the minimizer of function $\phi_{k+1}(\tau)$ is

$$\tau_{k+1} = \frac{\mathrm{tr}\left(AA^T(b - Ax(k))(b - Ax(k))^T\right)}{\|AA^T(b - Ax(k))\|_F^2} = \frac{\|A^T(b - Ax(k))\|_F^2}{\|AA^T(b - Ax(k))\|_F^2}.$$

We call the sequence $\{\tau_{k+1}\}_{k=0}^{\infty}$ that the sequence of optimal convergent factors.

3.3 The algorithm

Now, we summarize the search direction and optimal step size altogether and provide "The gradient based iterative algorithm with a sequence of optimal convergent factors".

Algorithm 3.1

Input step: Input matrix $A \in M_n(\mathbb{R})$ and vector $b \in \mathbb{R}^n$. Given any small positive number ϵ as an error.

Initializing step: Choose an initial vector $x(0) \in \mathbb{R}^n$. Set $k := 0$

Stopping rule: If $\delta_k := \|Ax(k) - b\|_F < \epsilon$, stop. Otherwise, go to the next step.

Updating step:

$\tau_{k+1} = \|A^T(b - Ax(k))\|_F^2 / \|AA^T(b - Ax(k))\|_F^2$

$x(k+1) = x(k) + \tau_{k+1}A^T\,(b - Ax(k))$

Set $k := k + 1$ and return to Stopping rule.

On the Algorithm 3.1, it seems that we have to triple-compute $A^T(b - Ax(k))$. Moreover, there are plenty of matrix-multiplications to compute. We then attempt to reform the Algorithm 3.1 and reduce any overlapping computations. Therefore, we obtain the well-applicable algorithm as follows:

Algorithm 3.1A

Input step: Input matrix $A \in M_n(\mathbb{R})$ and vector $b \in \mathbb{R}^n$. Given any small positive number ϵ as an error.

Initializing step: Choose an initial vector $x(0) \in \mathbb{R}^n$. Set $k := 0$. Compute $A_1 = A^T b$, $A_2 = A^T A$, $A_3 = AA_1$, and $A_4 = AA_2$.

Stopping rule: Compute $E(k) = b - Ax(k)$. If $\|E(k)\|_F < \epsilon$, stop. Otherwise, go to the next step.

Updating step:

$\tau_{k+1} = \sum_{i=1}^n \left(A_1(i) - \sum_{j=1}^n A_2(i,j)x(j)\right)^2 / \sum_{i=1}^n \left(A_3(i) - \sum_{j=1}^n A_4(i,j)x(j)\right)^2$,

$x(k+1) = x(k) + \tau_{k+1}A^T E(k)$.

Set $k := k + 1$ and return to Stopping rule.

Chapter 4

Convergence analysis of the proposed algorithm

In this chapter, we analyse our proposed algorithm by proving that Algorithm 3.1 converges to the exact solution. In addition, we provide the convergent rate and error estimates. Furthermore, we state the number of iterations correspond to a given satisfactory error. From now on, we denote $\kappa = \kappa(A)$, the condition number of A.

4.1 Convergence of the algorithm

Theorem 4.1. If (3.1) has a unique solution x^*, then the iterative sequence $\{x(k)\}$ generated by Algorithm 3.1 converge to x^* for any initial $x(0)$, i.e., $x(k) \to x^*$ as $k \to \infty$.

Proof. If $\nabla f(x(k)) = A^T(Ax(k) - b) = 0$ for some k, then $x(k) = x^*$ and the result holds. So assume that $\nabla f(x(k)) \neq 0$ for all k. Since $\nabla^2 f(x) = A^T A$ is a symmetric matrix, by Lemma 2.8, we have

$$\lambda_{\min}(A^T A)I \preceq A^T A \preceq \lambda_{\max}(A^T A)I. \tag{4.1}$$

The inequality (4.1) implies that the function f is strongly convex.

For convenience, we may write $\lambda_{\max}$ and $\lambda_{\min}$ instead of $\lambda_{\max}(A^T A)$ and $\lambda_{\min}(A^T A)$ respectively. We consider the function $\phi_{k+1}(\tau) := f(x(k) + \tau A^T(b - Ax(k)))$. From the inequality (2.5) in Lemma 2.12, with substituting $y = x(k) + \tau A^T(b - Ax(k))$ and $x = x(k)$, we obtain

$$\phi_{k+1}(\tau) \leqslant f(x(k)) - \tau\|\nabla f(x(k))\|_F^2 + \frac{\lambda_{\max}\tau^2}{2}\|\nabla f(x(k))\|_F^2. \tag{4.2}$$

Minimize over τ both sides. The right-hand side is minimized by $\tau = 1/\lambda_{\max}$, and has minimum value $f(x(k)) - (1/(2\lambda_{\max}))\|\nabla f(x(k))\|_F^2$. Therefore, we have

$$f(x(k+1)) = \phi_{k+1}(\tau_{k+1}) \leqslant f(x(k)) - \frac{1}{2\lambda_{\max}}\|\nabla f(x(k))\|_F^2. \tag{4.3}$$

Consider inequality (2.4) in Lemma 2.12, with substituting $y = x(k) + \tau A^T(b - Ax(k))$ and $x = x(k)$, we obtain

$$\phi_{k+1}(\tau) \geqslant f(x(k)) - \tau\|\nabla f(x(k))\|_F^2 + \frac{\lambda_{\min}\tau^2}{2}\|\nabla f(x(k))\|_F^2. \tag{4.4}$$

Minimize over τ both sides, we can find that $\tau = 1/\lambda_{\min}$ minimizes the right-hand side. Therefore, we have

$$f(x(k+1)) = \phi_{k+1}(\tau_{k+1}) \geqslant f(x(k)) - \frac{1}{2\lambda_{\min}}\|\nabla f(x(k))\|_F^2.$$

Since this holds for any $x \in \mathbb{R}^n$, we have

$$0 \geqslant f(x(k)) - \frac{1}{2\lambda_{\min}}\|\nabla f(x(k))\|_F^2.$$

Hence,

$$\|\nabla f(x(k))\|_F^2 \geqslant 2\lambda_{\min} f(x(k)). \tag{4.5}$$

Substituting (4.5) into (4.3) we have

$$\begin{aligned} f(x(k+1)) &\leqslant f(x(k)) - \frac{1}{2\lambda_{\max}} 2\lambda_{\min} f(x(k)) \\ &= \left(1 - \frac{\lambda_{\min}}{\lambda_{\max}}\right) f(x(k)). \end{aligned}$$

Notice that $1 - \lambda_{\min}/\lambda_{\max} = 1 - \kappa^{-2}$, hence

$$f(x(k+1)) \leqslant \left(1 - \kappa^{-2}\right) f(x(k)).$$

Since A is invertible, A^TA is positive definite i.e. $\lambda > 0$ for all $\lambda \in \sigma(A^TA)$. So $\kappa^{-2} > 0$ and thus

$$f(x(k+1)) \leqslant \left(1 - \kappa^{-2}\right) f(x(k)). \tag{4.6}$$

By induction, we obtain

$$f(x(k)) \leqslant \left(1 - \kappa^{-2}\right)^k f(x(0)) \tag{4.7}$$

which shows that $f(x(k))$ converges to 0 as $k \to \infty$. □

Theorem 4.2. Let $\{x(k)\}_{k=1}^{\infty}$ be the sequence of vector generated by Algorithm 3.1. For any $\epsilon > 0$ we have that $\|Ax(k) - b\|_F < \epsilon$ after k^* iterations for any k^* that respects:

$$k^* > \frac{\log \epsilon - \log f(x(0))}{\log\left(1 - \kappa^{-2}\right)}. \tag{4.8}$$

Proof. From (4.7)

$$f(x(k)) \leqslant \left(1 - \kappa^{-2}\right)^k f(x(0)) \to 0 \quad \text{as } k \to \infty.$$

By definition of convergence, we have that for all $\epsilon > 0$ there is a positive integer N such that for all $k \geqslant N$

$$\left(1 - \kappa^{-2}\right)^k f(x(0)) < \epsilon.$$

Taking logarithm both sides, we obtain

$$\begin{aligned} \log\left(\left(1 - \kappa^{-2}\right)^k f(x(0))\right) &< \log \epsilon \\ k \log\left(1 - \kappa^{-2}\right) + \log f(x(0)) &< \log \epsilon \\ k &> \frac{\log \epsilon - \log f(x(0))}{\log\left(1 - \kappa^{-2}\right)}. \end{aligned}$$

□

Corollary 4.3. An approximated solution, $x(k)$ will have an accuracy to decimal digit n after k^* iterations for any k^* that respects:

$$k^* > \frac{\log 0.5 - \log f(x(0)) - n}{\log (1 - \kappa^{-2})}. \tag{4.9}$$

Proof. The proof is straightforward by substituting $\epsilon = 0.5 \times 10^{-n}$ in (4.8). □

Now, with Corollary 4.3 we can have an improvement of Algorithm 3.1 by cutting out the stopping rule step and setting the exact iteration time k^* as follows:

Algorithm 3.1B

Input step: Input matrix $A \in M_n(\mathbb{R})$ and vector $b \in \mathbb{R}^n$. Given m as a decimal digit accuracy.

Initializing step: Choose an initial vector $x(0) \in \mathbb{R}^n$. Set $k := 0$. Compute $A_1 = A^T b$, $A_2 = A^T A$, $A_3 = AA_1$, $A_4 = AA_2$ and $k^* = \lceil (\log 0.5 - \log f(x(0)) - m)/(\log(1 - \kappa^{-2})) \rceil$.

Updating step:

For $k = 0, 1, \ldots, k^*$,

$\tau_{k+1} = \sum_{i=1}^{n} \left(A_1(i) - \sum_{j=1}^{n} A_2(i,j)x(j) \right)^2 / \sum_{i=1}^{n} \left(A_3(i) - \sum_{j=1}^{n} A_4(i,j)x(j) \right)^2$,

$x(k+1) = x(k) + \tau_{k+1} A^T E(k)$.

Set $k := k + 1$ and continue the process.

Note that $\lceil x \rceil$ is the ceiling function of x.

4.2 convergent rate and error estimates

We now discuss the rate of convergence and error estimates of Algorithm 3.1. According to the proof of Theorem 4.1 equations (4.6) and (4.7) give us the following error estimates:

$$\|Ax(k) - b\|_F \leqslant \left(1 - \kappa^{-2}\right)^{\frac{1}{2}} \|Ax(k) - b\|_F, \tag{4.10}$$

$$\|Ax(k) - b\|_F \leqslant \left(1 - \kappa^{-2}\right)^{\frac{k}{2}} \|Ax(0) - b\|_F. \tag{4.11}$$

Moreover, we can see that $f(x(k)) \leqslant (1 - \kappa^{-2}) f(x(k-1)) < f(x(k-1))$.

Theorem 4.4. Assume that the equation (3.1) has a unique solution x^*, then the Algorithm 3.1 converges for any initial vector with its convergent rate (with respect to the error $\|Ax(k) - b\|_F$) is governed by $\sqrt{1 - \kappa^{-2}}$. Moreover, the error estimates $\|Ax(k) - b\|_F$ compared to the previous step and the first step are provided by (4.10) and (4.11), respectively. In particular, the relative error at each iteration gets smaller than the previous one.

Now, to establish another estimation of error, we recall the following properties.

Lemma 4.5. [14] For any matrices A and B with proper sizes, we have

i) $\|A^T\|_2 = \|A\|_2$,

ii) $\|A^T A\|_2 = \|A\|_2^2$,

iii) $\|AB\|_F \leqslant \|A\|_2 \|B\|_F$.

Theorem 4.6. Assume that the equation (3.1) has a unique solution x^*, then the error estimates $\|x(k) - x^*\|_F$ compared to the previous step and the first step of Algorithm 3.1 are given as follows:

$$\|x(k) - x^*\|_F \leqslant \kappa \left(1 - \kappa^{-2}\right)^{\frac{1}{2}} \|x(k-1) - x^*\|_F, \tag{4.12}$$

$$\|x(k) - x^*\|_F \leqslant \kappa \left(1 - \kappa^{-2}\right)^{\frac{k}{2}} \|x(0) - x^*\|_F. \tag{4.13}$$

In particular, the convergent rate of the algorithm is governed by $\sqrt{1 - \kappa^{-2}}$.

Proof. Applying Lemma 4.5 to the equation (4.11), we obtain

$$\begin{aligned}
\|x(k) - x^*\|_F &= \|A^{-1}Ax(k) - A^{-1}Ax^*\|_F \\
&= \|A^{-1}\left(Ax(k) - b\right)\|_F \\
&\leqslant \|A^{-1}\|_2 \|Ax(k) - b\|_F \\
&< \|A^{-1}\|_2 \left(1 - \kappa^{-2}\right)^{\frac{k}{2}} \|Ax(0) - b\|_F \\
&\leqslant \|A^{-1}\|_2 \left(1 - \kappa^{-2}\right)^{\frac{k}{2}} \|A\|_2 \|x(0) - x^*\|_F \\
&= \kappa \left(1 - \kappa^{-2}\right)^{\frac{k}{2}} \|x(0) - x^*\|_F.
\end{aligned}$$

Since the end behaviour of the above error depends on the term $\left(1 - \kappa^{-2}\right)^{\frac{k}{2}}$, the asymptotic rate of convergence for the algorithm is governed by $\sqrt{1 - \kappa^{-2}}$. Similarly, following from (4.10) and Lemma 4.5, we obtain

$$\begin{aligned}
\|x(k) - x^*\|_F &< \|A^{-1}\|_2 \left(1 - \kappa^{-2}\right)^{\frac{1}{2}} \|Ax(k-1) - b\|_F \\
&\leqslant \|A^{-1}\|_2 \left(1 - \kappa^{-2}\right)^{\frac{1}{2}} \|A\|_2 \|x(k-1) - x^*\|_F \\
&= \kappa \left(1 - \kappa^{-2}\right)^{\frac{1}{2}} \|x(k-1) - x^*\|_F,
\end{aligned}$$

and therefore, we have (4.12). □

Chapter 5

Numerical simulations

In this chapter, we illustrate the effectiveness and capability of our algorithm. We make the comparison reports of Algorithm 3.1 with the existing algorithms we have presented in Chapter 2. We include the table of iteration time and the CPU time as well as the error plot graphic in each example. All iterations have been carried out by MATLAB R2017a, Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz 2.60 GHz, RAM 8.00 GB. PC environment. For convenience, we use TauOpt, GI, LS, BB1, BB2, IT and CPU to represent Algorithm 3.1, the gradient based algorithm (Theorem 2.24), the least squares algorithm (Theorem 2.25), the BB step size type 1 (2.7) and type 2 (2.8), the iteration time and the CPU time (in seconds) respectively. At step kth of the iteration, we consider the following error:

$$\gamma_k := \|x(k) - x^*\|_F$$

where $x(k)$ is the kth approximated solution of the corresponding linear system.

Example 5.1. We consider the linear system $Ax = b$ where

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 5 \\ 14 \end{bmatrix}.$$

We choose an initial vector

$$x(0) = 10^{-6} \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

We run the algorithm 3.1 and we can see that from the table 5.1 that the approximated solutions converge to the exact solution

$$x^* = \begin{bmatrix} -3 \\ 4 \end{bmatrix}.$$

Furthermore, the algorithm 3.1 gives the fastest convergence as the figure shown.

Table 5.1*:* Iterative solution for Example 5.1.

k	x_1	x_2	γ_k
1	0.9714	2.3550	0.8597
2	-2.9926	3.9902	0.0025
3	-2.9902	3.9960	0.0021
4	-3.0000	4.0000	0.0000
Solution	-3.0000	4.0000	

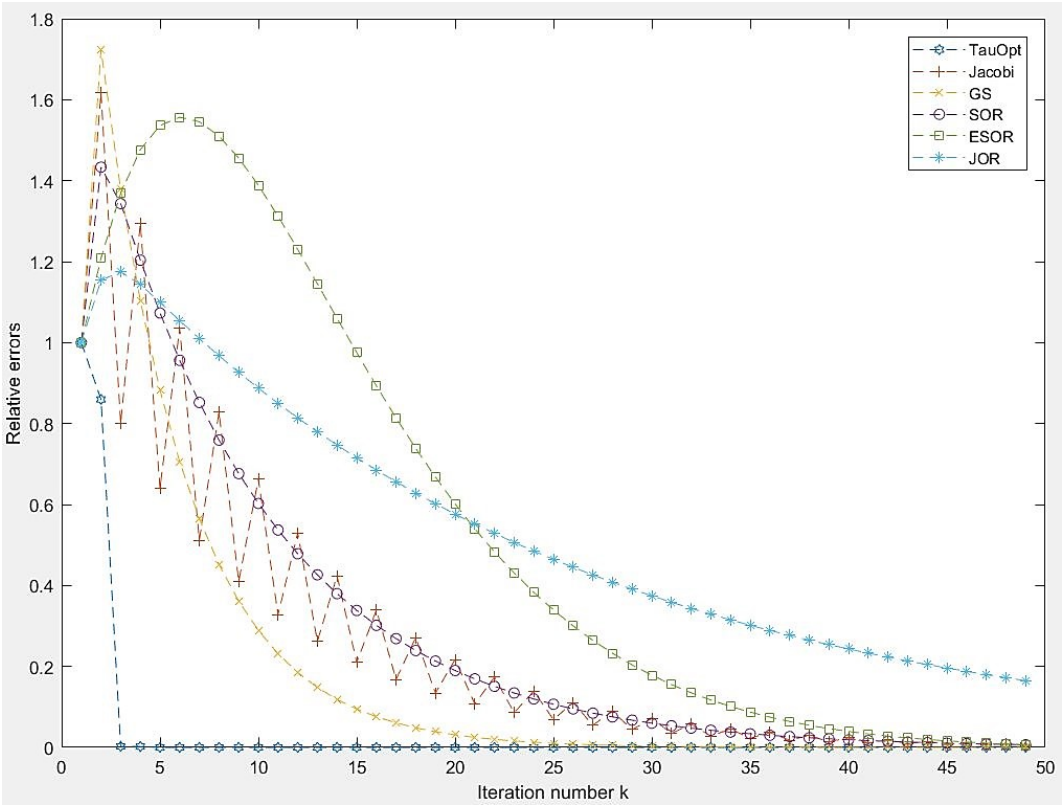

Figure 5.1: Relative errors for Example 5.1.

Table 5.2: Iteration time and cpu time for Example 5.1.

Method	IT	CPU
TauOpt	8	0.0156
Jacobi	134	0.0313
GS	71	0.0156
SOR	133	0.0313
ESOR	116	0.0625
JOR	345	0.0156

Example 5.2. We consider the larger linear system. We would like to show that for a coefficient matrix that does not have an appropriate property and make all approximated solutions from every method diverge, our algorithm still converges to the exact solution. Given

$$A = \begin{bmatrix} 1 & 5 & 8 & 4 & 8 & 5 \\ 5 & 2 & 7 & 7 & 6 & 5 \\ 8 & 7 & 9 & 8 & 6 & 4 \\ 4 & 7 & 8 & 6 & 7 & 1 \\ 8 & 6 & 6 & 7 & 2 & 0 \\ 5 & 5 & 4 & 1 & 0 & 2 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} -6 \\ -3 \\ -13 \\ 9 \\ -4 \\ -30 \end{bmatrix}.$$

We start with an initial vector

$$x(0) = 10^{-6} \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}^T.$$

As we can see from Figure 5.2 , Jacobi, Gauss-Seidel, SOR, ESOR, AOR and JOR diverge, but meanwhile, our algorithm continue to converge to 0.

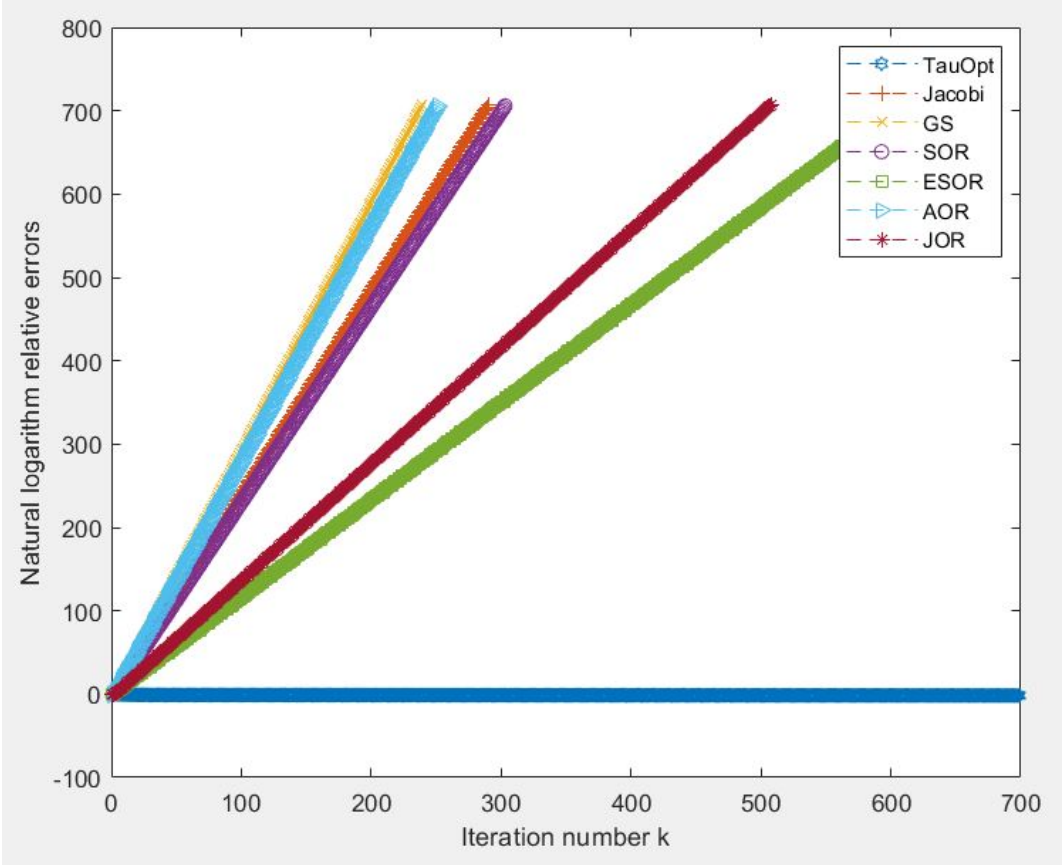

Figure 5.2*:* Natural logarithm relative errors for Example 5.2.

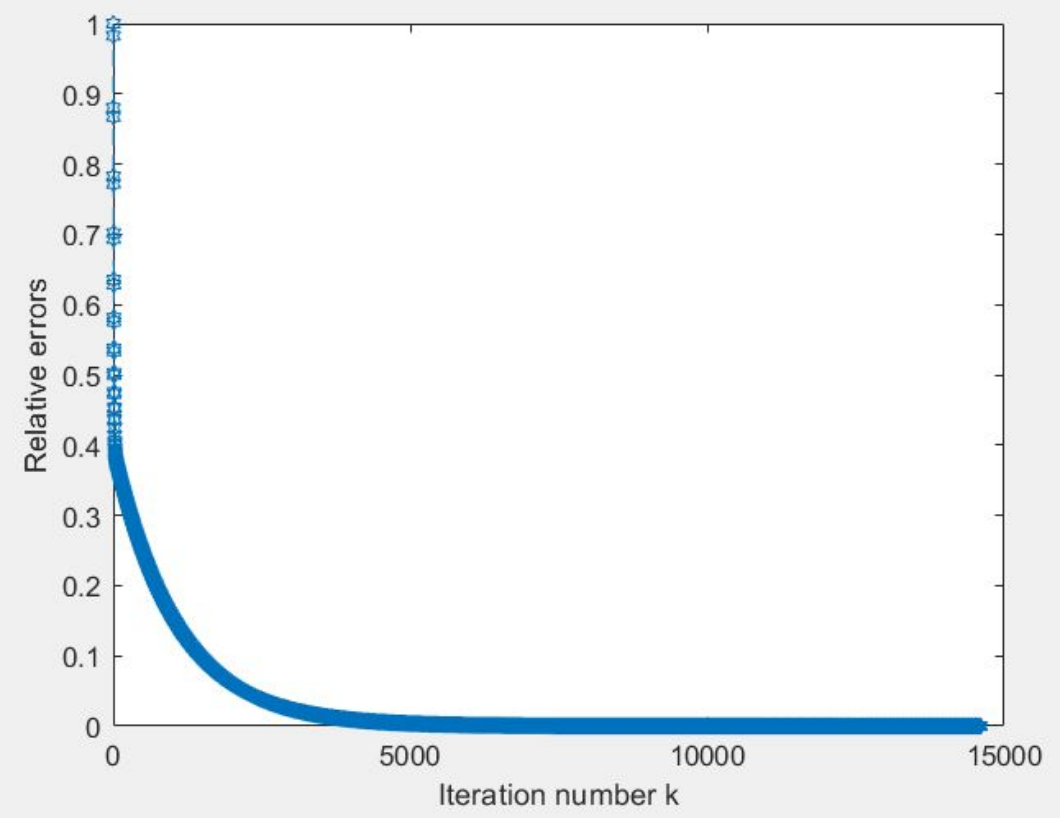

Figure 5.3*:* Relative errors of Algorithm 3.1 for Example 5.2.

As a result, the approximated solutions from our algorithm converges to the exact solution as shown in Figure 5.3

$$x^* = \begin{bmatrix} -1 \\ -3 \\ 0 \\ 2 \\ 4 \\ -6 \end{bmatrix}$$

with six decimal digits accuracy using 14612 iterations.

Example 5.3. This example we consider a 10×10 coefficient matrix. We compare Algorithm 3.1 with GI, LS and BB algorithms. Given

$$A = \begin{bmatrix} -1 & 2 & -3 & 7 & 6 & 9 & 0 & -5 & -8 & 5 \\ 1 & 5 & -4 & -1 & 0 & 3 & 5 & 8 & -7 & 3 \\ 3 & 4 & -7 & 6 & 0 & 3 & -1 & 7 & 4 & -5 \\ -1 & 1 & 7 & 4 & -9 & -1 & 0 & 0 & -5 & 3 \\ 1 & -7 & 3 & 2 & -4 & 1 & 0 & 5 & 9 & 3 \\ 3 & 1 & 4 & -4 & -6 & 3 & 3 & 6 & -9 & 4 \\ 6 & 1 & 8 & 2 & -3 & -8 & 7 & -4 & 2 & 6 \\ 8 & 1 & 5 & 2 & 3 & 3 & -2 & 8 & 7 & -9 \\ -9 & 5 & 4 & -1 & 0 & 6 & 4 & -8 & 5 & -3 \\ 0 & 1 & -3 & 1 & 6 & -1 & 9 & 5 & -1 & 0 \end{bmatrix} \text{ and } b = \begin{bmatrix} 23 \\ -88 \\ 100 \\ -93 \\ 28 \\ -156 \\ -100 \\ 148 \\ 160 \\ -2 \end{bmatrix}.$$

Choose the initial vector

$$x(0) = 10^{-6} \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}^T.$$

After running Algorithm 1, the exact solution is given by

$$x^* = \begin{bmatrix} -3 & 2 & 1 & 4 & 5 & 7 & -1 & -2 & 9 & -8 \end{bmatrix}^T.$$

To compare with GI and LS algorithms, we randomly choose the convergent factors $\mu = 0.0005$ and $\mu = 0.005$ for GI and LS, respectively. The natural logarithm relative errors shown in Figure 5.4 implies that Algorithm 3.1 is comparable to both of BB algorithms but still outperforms the GI and LS algorithms in the performances of convergence. Moreover, Table 5.3 shows that Algorithm 3.1 has reached the lowest of CPU time.

Table 5.3*:* Iteration time and CPU time for Example 5.3.

Method	IT	CPU
TauOpt	840	0.0313
GI	6020	0.2813
LS	2531	0.1250
BB1	77	0.0625
BB2	90	0.0625

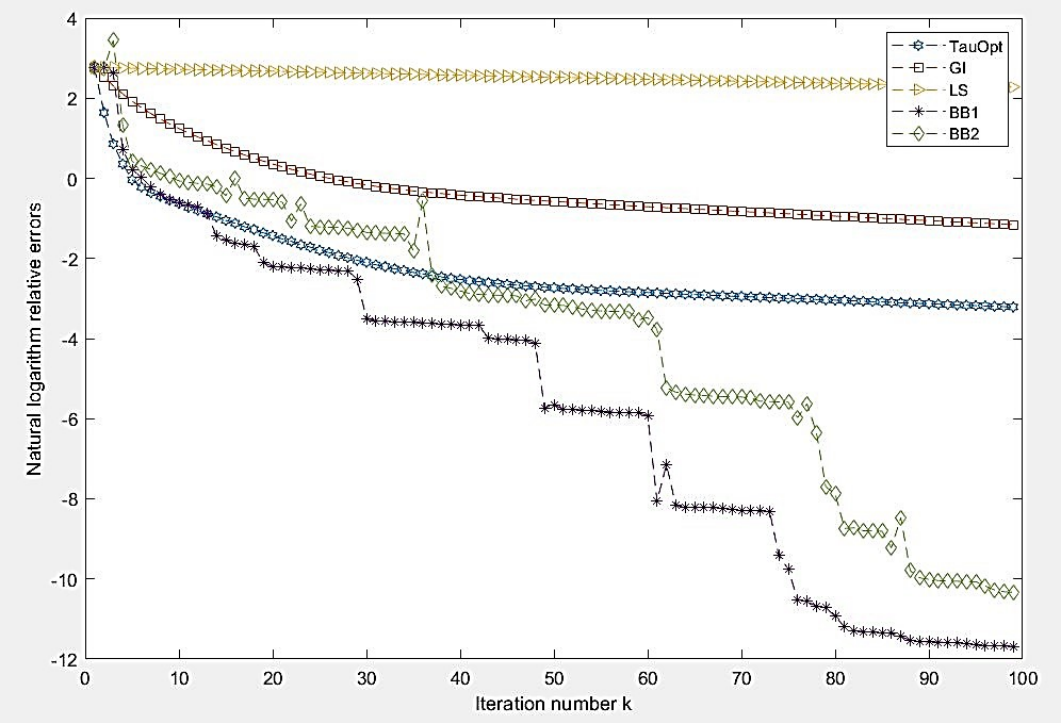

Figure 5.4*:* Natural logarithm relative errors for Example 5.3.

Example 5.4. This example we consider the convergence rate of the algorithm. We show the correctness of Theorem 4.4 and Theorem 4.6 which they imply that the closer to 1 the condition number, the faster the convergence of the algorithm. Let $a \in \mathbb{R}$ and consider

$$A = \begin{bmatrix} a & 1 \\ 2 & 3 \end{bmatrix}.$$

Thus, the condition number of the iteration matrix depends on a. By taking different values of a, we then obtain the results that are shown in Table 5.4 and Figure 5.5.

Table 5.4*:* Condition number, iteration time and CPU time for each a for Example 5.4.

a	$\kappa(A)$	IT	CPU
-3	1.3504	11	0.0469
7	2.9802	19	0.0625
10	3.8089	27	0.0781
15	5.9720	36	0.1094

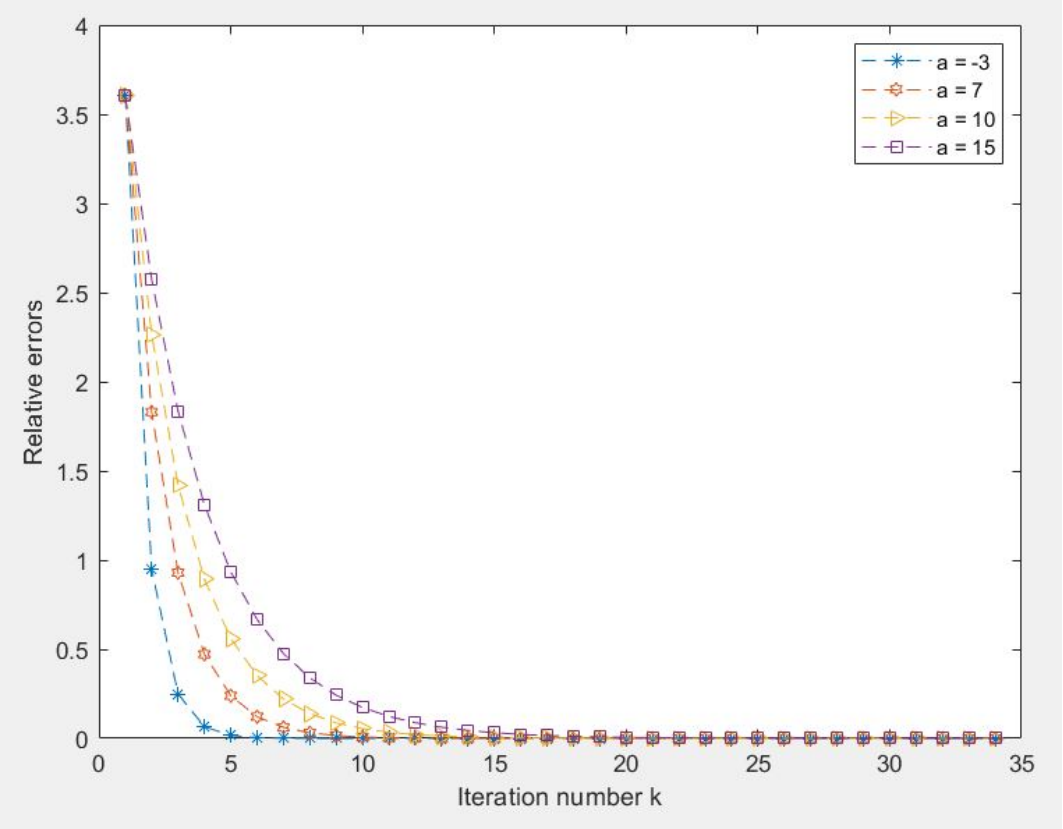

Figure 5.5: Relative errors for Example 5.4.

Chapter 6

Conclusion and suggestion

6.1 conclusion

A new algorithm, that is, the gradient based iterative algorithm with a sequence of optimal convergent factors, is established for solving linear systems. By utilizing knowledge in gradients, steepest descent, and convex optimization techniques, we hence obtain the effective algorithm that can be applicable for any linear systems without any conditions but the systems must have a unique solution. In addition, we introduce a new formula for a convergent factors and call it an optimal convergent factor. It therefore excellently improve the algorithm in performance of convergence. The asymptotic rate of convergence is governed by $\sqrt{1-\kappa^{-2}}$, where κ is the condition number of A. The numerical experiments reveal the competency of the proposed algorithm comparing to well-known and recent methods. The iteration time and the CPU time in each simulation indicate that our algorithm is a good choice for solving linear systems.

6.2 Suggestion

The techniques of gradients, steepest descent, and convex optimization might be useful for a class of matrix equations for examples:

- The Sylvester equation

$$AX + XB = C,$$

- The Lyapunov equation

$$AX - XA^T = Q,$$

- The Kalman-Yakubovich equation

$$AXB + X = C.$$

However, these topics require more studies and can be another further research.

References

[1] G.Strang, “Linear Algebra and Its Applications,” 4th ed. USA: Brooks Cole, 2006.

[2] P. J. Olver and C. Shakiban, “Applied Linear Algebra,” Upper Saddle River, USA: Pearson Education, 2006.

[3] B. N. Datta, “Numerical Linear Algebra and Applications,” 2nd ed. Philadelphia, USA: Society for Inductrial and Applied Mathematics, 2010.

[4] D. M. Young, “Iterative Methods for Solving Partial Difference Equations of Elliptical Type,” Ph.D. thesis, Harvard University, Cambridge, Massachusetts, 1950.

[5] P. Albrechtt and M. P. Klein, “extrapolated iterative methods for linear systems,” SIAM Journal on Numerical Analysis, vol. 21, no. 1, pp. 192-201, 1984.

[6] D. M. Young, “Iterative Solution of Large Linear Systems,” Academic Press, New York/London, 1971.

[7] A. J. H. Hallett, “The convergence of accelerated overrelaxation iterations,” mathematics of computation, vol. 47, no. 175,pp. 219-223, 1986.

[8] A. Hadjidimos, A. Psimami, and A. Yeyios, “On the convergence of some generalized iterative methods,” Linear Algebra and its Applications, vol.75, pp.117-132, 1986.

[9] D. Feng, and C. Tongwen, “Iterative least-squares solutions of coupled sylvester matrix equations,” Systems & Control Letters, vol. 54, pp. 95-107, 2005.

[10] D. Feng, and C. Tongwen, “Gradient based iterative algorithms for solving a class of matrix equations,” IEEE Transactions on Automatic Control, vol. 50, no. 8, pp. 1216-1221, 2005.

[11] D. Feng, and C. Tongwen, “Hierarchical gradient-based identification of multivariable discrete-time systems,” Automatica, vol. 41, no. 2, pp. 315-325, 2005.

[12] D. Feng, and C. Tongwen, “Hierarchical least squares identification methods for multivariable systems,” IEEE Transactions on Automatic Control, vol. 50, no. 3, pp. 397-402, 2005.

[13] D. Feng, P. X. Liu, J. Ding, “Iterative solutions of the generalized Sylvester matrix equations by using the hierarchical identification principle,” Applied Mathematics and Computation, vol. 197, pp. 41-50, 2008.

[14] R. A. Horn, C. R. Johnson, “Matrix Analysis,” 2nd edition, Cambridge University Press, New York, 1990.

[15] S. Boyd, L. Vandenberghe, “Convex Optimization,” Cambridge University Press, 2004.

[16] A. M. Ostrowski, “On the linear iteration procedures for symmetric matrices,” Rendiconti di Matematica e delle sue Applicazioni, vol. 14, pp. 140-163, 1954.

[17] W. Kahan, “Gauss-Seidel Methods of Solving Large Systems of Linear Equations,” Ph.D. thesis, University of Toronto, Toronto, Canada, 1958.

[18] E. Reich, “On the convergence of the classical iterative procedures for symmetric matrices,” The Annual of Mathematical Statistics, vol. 20, pp. 445-451, 1949.

[19] J. Barzilai, and J. M. Borwein, “Two-Point Step Size Gradient Methods,” IMA Journal of Numerical Analysis, vol. 8, no. 1, pp. 141-148, 1988.

Appendix

Appendix A

The research paper

The 24th Annual Meeting in Mathematics (AMM 2019)

Gradient based iterative algorithm with a sequence of optimal convergent factors for solving linear systems

Adisorn Kittisopaporn[1] and **Pattrawut Chansangiam**[2,*]

Department of Mathematics, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand; [1]a.kittisopaporn@gmail.com; [2]pattrawut.ch@kmitl.ac.th
*Correspondence: pattrawut.ch@kmitl.ac.th

Abstract: In this paper, we introduce a new iterative algorithm for solving linear systems. The algorithm is based on a gradient and along with a sequence of optimal convergent factors. We show that the algorithm is applicable with any initial vector as long as the coefficient matrix is invertible. Then we analyse the convergence rate and error estimates. Furthermore, we demonstrate the numerical simulations comparing to well-known methods and recent methods.

Keywords: linear system; iterative algorithm; gradient; condition number; norm

AMS Math Classification (2010) : 26B25, 65F10, 65F35

1 Introduction

Consider a linear system

$$Ax = b \tag{1}$$

where $A \in M_n(\mathbb{R})$ and $b \in \mathbb{R}^n$ are given. Here, we denote the set of n-by-n real matrices by $M_n(\mathbb{R})$. Such system has a unique solution if and only if A is invertible. The methods of solving the linear systems are classified into two groups: direct methods, and iterative methods.

A direct method is a way to solve for an exact solution. Most of the direct methods are well-known in general linear algebra such as Gaussian elimination, inversion matrix, Cramer's rule, and several matrix decompositions. The direct method is advantage in availability for any invertible matrices except for the decomposition method which has to be considered at different types of matrices. A disadvantage of the direct method is that the solution occurs at the final step of computation. In other word, we must finalize the entire process in order to obtain the solution. Stopping at somewhere during the process does not offer even a numerical solution. In addition, miscalculation in some steps might have an effect to every step afterwards and the outcome solution eventually absolutely differ from the exact solution. Also, large linear systems require huge spaces of memory. Therefore, the direct method is suited for small linear systems.

On the other hand, an iterative method create a sequence of numerical solutions so that starting from an initial approximation with sufficient iteration number finally comes an accurate solution. A group of methods for solving (1), called stationary iterative methods, can be expressed in the simple form

$$x^{(k+1)} = Tx^{(k)} + c.$$

The Jacobi (J) method and the Gauss-Seidel (GS) method are two classical stationary iterative methods deriving by splitting

$$A = L + D + U$$

where D is a diagonal matrix and L (U) is a lower (upper) triangular matrix. The iteration matrices for the Jacobi and GS methods are respectively given by

$$T_{\text{J}} = -D^{-1}(L+U),\ c_{\text{J}} = D^{-1}b,$$
$$T_{\text{GS}} = -(D+L)^{-1}U,\ c_{\text{GS}} = (D+L)^{-1}b.$$

In [1] David Young Jr. developed the Successive Over-Relaxation (SOR) method. The method can be derived from the GS method by introducing an extrapolation parameter ω and its iteration matrix is viewed as

$$T_{\text{SOR}} = (D+\omega L)^{-1}\left((1-\omega)D - \omega U\right),$$
$$c_{\text{SOR}} = \omega(D+\omega L)^{-1}b, \qquad 0 < \omega < 2.$$

The SOR method has received much attention and has been evolved continually into new iterative methods. We recall well-known methods that have been developed from the SOR method as follows:

- Jacobi Over-Relaxation (JOR) method [2]

$$T_{\text{JOR}} = D^{-1}\left((1-\alpha)D - \alpha(L+U)\right),$$
$$c_{\text{JOR}} = \alpha D^{-1}b, \qquad \alpha > 0.$$

- Extrapolated SOR (ESOR) method [3]

$$T_{\text{ESOR}} = (D+\omega L)^{-1}\left((\omega-\tau)L - \tau U + (1-\tau)D\right),$$
$$c_{\text{ESOR}} = \tau(D+\omega L)^{-1}b, \qquad 0 < \omega < 2,\ |\tau| < \omega.$$

- Accelerated Over-Relaxation (AOR) method [4]

$$T_{\text{AOR}} = (D-\alpha L)^{-1}\left((\beta-\alpha)L + \beta U + (1-\beta)D\right),$$
$$c_{\text{AOR}} = \beta(D-\alpha L)^{-1}b, \qquad 0 < \beta < 2,\ 0 < \alpha < \beta.$$

Now, let us get back to the linear system (1). If A is assumed to be invertible, then the consistent system (1) has a unique solution

$$x^* = A^{-1}b.$$

The hierarchical identification technique and the minimization of quadratic norm-error functions yield gradient based iterative algorithms. Many researchers have been developing such iterative method and utilizing for a class of matrix equations, see Feng and Tongwen [5], [6], [7], [8], Feng et al. [9], Niu et al. [10], Wang et al. [11], Xie and Ma [12] and Zhang and Sheng [13]. Convergence analysis for such algorithms relies on two famous matrix norms, namely, the Frobenius norm $\|\cdot\|_F$ and the spectral norm $\|\cdot\|_2$. We recall the following algorithms:

Proposition 1. *(Feng, [8]) Suppose that the linear system* (1) *has a unique solution* x^*. *Let* $0 < \mu < \frac{2}{\|A\|_2^2}$ *or* $0 < \mu < \frac{2}{\|A\|_F^2}$. *Then the iterative solution* $x(k)$ *given by the following gradient based iterative (GI) algorithm:*

$$x(k+1) = x(k) + \mu A^T\left(b - Ax(k)\right) \tag{2}$$

converges to x^* *for any initial value* $x(0)$.

Proposition 2. *(Feng, [8]) Suppose the system* (1) *has a unique solution* x^*. *Let* $0 < \mu < 2$. *Then the iterative solution* $x(k)$ *given by the following least-squares based iterative (LS) algorithm*

$$x(k+1) = x(k) + \mu\left(A^TA\right)^{-1}A^T\left(b - Ax(k)\right) \tag{3}$$

leads to $\lim_{k\to\infty} x(k) = x^*$ *for any initial value* $x(0)$.

Another technique for solving linear systems is to formulate them into an unconstrained convex optimization problem which can be written by: $\min_{x\in\mathbb{R}^n} f(x)$ where f is a continuously differentiable function. We apply gradient method searches along with the steepest descent. The steepest descent is a gradient scheme where the step size τ_k is chosen appropriately at each individual iteration so that the maximum amount of decrease of the objective function is achieved. To do this, let x_k be the current iterate point, and $v_k = v(x_k) = \nabla f(x_k)$ be the gradient vector at x_k. The steepest descent method defines the next iteration by

$$x_{k+1} = x_k - \tau_k^* v_k \tag{4}$$

where $\tau_k^* > 0$ satisfies

$$f(x_k - \tau_k^* v_k) = \min_{\tau>0}\{f(x_k) - \tau v_k\}.$$

In 1988, Barzilai and Borwein approached their step size in the current iteration. For the iterative equation (4), the step size τ_k can be chosen either (5) or (6) as follows:

$$\tau_k = \frac{s_{k-1}^T y_{k-1}}{\|y_{k-1}\|_2^2}, \tag{5}$$

$$\tau_k = \frac{\|s_{k-1}\|_2^2}{s_{k-1}^T y_{k-1}} \tag{6}$$

where $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = v_k - v_{k-1}$. We call the iterative method with a convergence factor chosen above that the BB method. For the convergence analysis of the above step sizes, see Barzilai and Borwein [14], and Yuan [15]. The results of Barzilai and Borwein has encouraged and brought about many researches on the gradient method, e.g. Fletcher [16], Dai et al. [17], Dai and Fletcher [18], Yuan [19], and Dai and Yuan [20].

In the present paper, we propose a new gradient based iterative algorithm with a sequence of optimal convergent factors for solving linear systems (see Section 2). Then, we make convergence analysis for the proposed algorithm, including the convergence rate and error estimates (see Section 3). Numerical experiments are provided in order to illustrate the capability and effectiveness of the proposed algorithm (see Section 4). It turns out that our algorithm has a better performance than that of all mentioned algorithms. Finally, we conclude the paper with some remarks.

In order to make convergence analysis, the spectral norm, the Frobenius norm, and the condition number of $A \in M_n(\mathbb{R})$ are used and respectively defined by

$$\begin{aligned} \|A\|_2 &= \sqrt{\lambda_{max}(A^T A)}, \\ \|A\|_F &= \sqrt{\mathrm{tr}(AA^T)}, \\ \mathrm{cond}(A) &= \left(\frac{\lambda_{max}(A^T A)}{\lambda_{min}(A^T A)}\right)^{1/2}. \end{aligned}$$

We recall the following properties.

Lemma 3. *(Horn, [21]) For any square matrices A and B of the same size, we have*

(i) $\|A^T A\|_2 = \|A\|_2^2$

(ii) $\|AB\|_F \leqslant \|A\|_2 \|B\|_F$.

2 The gradient based iterative algorithm with a sequence of optimal convergent factors

In this section, we introduce a new method for solving linear systems based on gradients and we provide an appropriate sequence of convergent factors which minimizes an error at each iteration.

Consider the linear system (1) where $A \in M_n(\mathbb{R})$ is an invertible matrix, $b \in \mathbb{R}^n$ is a known constant vector, and $x \in \mathbb{R}^n$ is an unknown vector. We firstly define the quadratic norm-error function

$$\varphi : \mathbb{R}^n \to \mathbb{R}, \quad \varphi(x) := \frac{1}{2}\|Ax - b\|_F^2. \tag{7}$$

Since A is invertible, the consistent system (1) has a unique solution and hence an optimal vector x^* of φ exists. We shall start by having an arbitrary initial vector $x(0)$ and then at every step $k > 0$ we iteratively move to the

next vector $x(k+1)$ with an appropriate direction, i.e., the negative gradient of φ, together with a suitable step size μ_{k+1}. The gradient based iterative method thus can be described through the following recursive rule:

$$x(k+1) = x(k) - \mu_{k+1}\nabla\varphi(x(k)). \tag{8}$$

In order to minimize the function φ, we start by searching a direction. We will deduce their gradients in detail. Recall the following gradient formula:

$$\frac{d}{dX}\text{tr}(AX) = \frac{d}{dX}\text{tr}(X^TA^T) = A^T.$$

Now, we get

$$\begin{aligned}\nabla\varphi(x) &= \frac{1}{2}\frac{d}{dx}\text{tr}\left((Ax-b)(Ax-b)^T\right)\\ &= \frac{1}{2}\frac{d}{dx}\text{tr}\left(Axx^TA^T - bx^TA^T - Axb^T + bb^T\right)\\ &= \frac{1}{2}\left(A^TAx + A^TAx - A^Tb - A^Tb\right)\\ &= A^T(Ax-b).\end{aligned}$$

Thus, our new iterative equation is in the form

$$x(k+1) = x(k) + \mu_{k+1}A^T(b - Ax(k)).$$

To generate the best step size at each iteration, we minimize an error which occurs at the next iteration, $x(k+1)$. We then define a new function $\psi : [0,\infty) \to \mathbb{R}$ by for each $k \in \mathbb{N}\cup\{0\}$,

$$\begin{aligned}\psi_{k+1}(\mu) &:= \varphi(x(k+1))\\ &= \frac{1}{2}\|A\left(x(k) + \mu A^T(b - Ax(k))\right) - b\|_F^2.\end{aligned}$$

Putting $\tilde{c} = b - Ax(k)$ and $\tilde{b} = AA^T\tilde{c}$, we get

$$\psi_{k+1}(\mu) = \frac{1}{2}\|\mu_{k+1}\tilde{b} - \tilde{c}\|_F^2.$$

Since

$$\begin{aligned}\frac{d}{d\mu}\psi_{k+1}(\mu) &= \frac{1}{2}\frac{d}{d\mu}\text{tr}\left((\mu\tilde{b}-\tilde{c})(\mu\tilde{b}-\tilde{c})^T\right)\\ &= \frac{1}{2}\frac{d}{d\mu}\text{tr}\left(\mu\tilde{b}\mu\tilde{b}^T - \mu\tilde{b}\tilde{c}^T - \tilde{c}\mu\tilde{b}^T + \tilde{c}\tilde{c}^T\right)\\ &= \frac{1}{2}\left(2\mu\text{tr}(\tilde{b}\tilde{b}^T) - 2\text{tr}(\tilde{b}\tilde{c}^T)\right) = 0,\end{aligned}$$

we have $\mu = \text{tr}(\tilde{b}\tilde{c}^T)/\text{tr}(\tilde{b}\tilde{b}^T)$. Hence, the minimizer of function $\psi_{k+1}(\mu)$ is

$$\begin{aligned}\mu_{k+1} &= \frac{\text{tr}\left(AA^T(b-Ax(k))(b-Ax(k))^T\right)}{\text{tr}\left(AA^T(b-Ax(k))(b-Ax(k))^TAA^T\right)}.\\ &= \frac{\|A^T(b-Ax(k))\|_F^2}{\|AA^T(b-Ax(k))\|_F^2}.\end{aligned} \tag{9}$$

We call the sequence $\{\mu_{k+1}\}_{k=0}^{\infty}$ the sequence of optimal convergent factors.

Now, we summarize the search direction and optimal step size altogether and provide "The gradient based iterative algorithm with a sequence of optimal convergent factors".

Algorithm 1

Input step: Input matrix $A \in M_n(\mathbb{R})$ and vector $b \in \mathbb{R}^n$. Given any small positive number ϵ as an error.

Initializing step: Choose an initial vector $x(0) \in \mathbb{R}^n$. Set $k := 0$

Stopping rule: If $\delta_k := \|Ax(k) - b\|_F < \epsilon$, stop. Otherwise, go to the next step.

Updating step:

$\mu_{k+1} = \|A^T(b - Ax(k))\|_F^2 / \|AA^T(b - Ax(k))\|_F^2$

$x(k+1) = x(k) + \mu_{k+1}A^T(b - Ax(k))$

Set $k := k+1$ and return to Stopping rule.

3 Convergence analysis

In this section, we will show that Algorithm 1 converges to the exact solution. Moreover, we provide the convergent rate, the error estimates and the number of iterations corresponding to a given satisfactory error.

Our analysis will be based on a matrix partial order and strongly convex functions defined below. We then firstly begin by showing some important properties of strongly convex functions, and use these properties in the proof of the convergence of the algorithm.

Recall that the Löwner partial order $\preceq$ for real symmetric matrices is defined by

$$A \preceq B \Leftrightarrow B - A \text{ is positive semi-definite.}$$

Using the definition of positive semi-definite matrices, this can be reformulated as:

$$A \preceq B \Leftrightarrow x^T Ax \leqslant x^T Bx, \forall x \in \mathbb{R}^n.$$

Recall also that a twice-differentiable convex function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *strongly convex* if there exist constants $0 \leqslant m < M$ such that for all $x \in \mathbb{R}^n$,

$$mI \preceq \nabla^2 f(x) \preceq MI.$$

Using the definition of the partial order $\preceq$, this is equivalent to:

$$my^T y \leqslant y^T \nabla^2 f(x) y \leqslant My^T y, \forall x, y \in \mathbb{R}^n.$$

In other words, m (M, resp.) is a lower (upper, resp.) bound on the smallest (largest, resp.) eigenvalue of $\nabla^2 f(x)$ for all x.

Lemma 4. *(Stephen, [22]) If f is strongly convex on $\mathbb{R}^n$, then for $x, y \in \mathbb{R}^n$ we have*

$$f(y) \geqslant f(x) + \nabla f(x)^T (y - x) + \frac{m}{2}\|y - x\|_F^2,$$
$$f(y) \leqslant f(x) + \nabla f(x)^T (y - x) + \frac{M}{2}\|y - x\|_F^2.$$

Theorem 5. *If the system* (1) *is consistent and has a unique solution x^*, then the sequence $\{x(k)\}$ generated by Algorithm 1 converges to x^* for any initial vector $x(0)$.*

Proof. We shall show that $x(k) \to x^*$ as $k \to \infty$. In case of $\nabla\varphi(x(k)) = A^T(Ax(k) - b) = 0$ for some k, then $x(k) = x^*$ and the result holds. So assume that $\nabla\varphi(x(k)) \neq 0$ for all k. Since $\nabla^2\varphi(x) = A^T A$ is a positive semi-definite matrix, we have

$$\lambda_{min}(A^T A)I \leqslant A^T A \leqslant \lambda_{max}(A^T A)I. \tag{10}$$

Thus, φ is strongly convex. For convenience, we may write λ_{min} and λ_{max} instead of $\lambda_{min}(A^T A)$ and $\lambda_{max}(A^T A)$ respectively. We consider the function $\psi_{k+1}(\mu)$ of the step size τ. Applying Lemma 4, we obtain

$$\varphi(x(k+1)) \leqslant \varphi(x(k)) - \mu\|\nabla\varphi(x(k))\|_F^2 + \frac{\lambda_{max}\mu^2}{2}\|\nabla\varphi(x(k))\|_F^2.$$

Minimize the above inequality over μ both sides. The RHS is minimized by $\mu_{k+1} = 1/\lambda_{max}$, and

$$\varphi(x(k+1)) \leqslant \varphi(x(k)) - \frac{1}{2\lambda_{max}}\|\nabla\varphi(x(k))\|_F^2. \tag{11}$$

From another inequality in Lemma 4, we have

$$\varphi(x(k+1)) \geqslant \varphi(x(k)) - \mu\|\nabla\varphi(x(k))\|_F^2 + \frac{\lambda_{min}\mu^2}{2}\|\nabla\varphi(x(k))\|_F^2.$$

We find that $\mu = 1/\lambda_{min}$ minimizes the RHS i.e.,

$$\begin{aligned} 0 &\geqslant \varphi(x(k)) - \frac{1}{\lambda_{min}}\|\nabla\varphi(x(k))\|_F^2 + \frac{1}{2\lambda_{min}}\|\nabla\varphi(x(k))\|_F^2 \\ &= \varphi(x(k)) - \frac{1}{2\lambda_{min}}\|\nabla\varphi(x(k))\|_F^2. \end{aligned}$$

Hence,

$$\|\nabla\varphi(x(k))\|_F^2 \geqslant 2\lambda_{min}\varphi(x(k)). \tag{12}$$

Substituting (12) into (11) we have

$$\begin{aligned}\varphi(x(k+1)) &\leqslant \varphi(x(k)) - \frac{1}{2\lambda_{max}} 2\lambda_{min}\varphi(x(k)) \\ &= \left(1 - \frac{\lambda_{min}}{\lambda_{max}}\right)\varphi(x(k)).\end{aligned}$$

Since A is an invertible matrix, A^TA is a positive definite matrix i.e., $\lambda > 0$ for all $\lambda \in \sigma(A^TA)$. Hence, $c := 1 - \lambda_{min}/\lambda_{max} < 1$ and

$$\varphi(x(k+1)) \leqslant c\varphi(x(k)). \tag{13}$$

By induction we obtain

$$\varphi(x(k)) \leqslant c^k\varphi(x(0)) \tag{14}$$

which shows that $\varphi(x(k)) \to 0$, or equivalently $x(k) \to x^*$ as $k \to \infty$. □

We now discuss the convergence rate and error estimates of Algorithm 1.

Theorem 6. *Suppose that* (1) *has a unique solution* x^**, then Algorithm 1 converges for any initial vector with its convergence rate governing by*

$$\frac{\sqrt{\text{cond}^2(A) - 1}}{\text{cond}(A)}.$$

Moreover, the relative error estimates $\|Ax(k) - b\|_F$ *compared to the previous step and the first step are provided by* (15) *and* (16)*, respectively. In particular, the relative error at each iteration gets smaller than the previous one.*

Proof. According to the results of proof of Theorem 5, the relative error estimates are given by

$$\|Ax(k) - b\|_F \leqslant \frac{\sqrt{\text{cond}^2(A) - 1}}{\text{cond}(A)}\|Ax(k-1) - b\|_F, \tag{15}$$

$$\|Ax(k) - b\|_F \leqslant \left(\frac{\sqrt{\text{cond}^2(A) - 1}}{\text{cond}(A)}\right)^k \|Ax(0) - b\|_F. \tag{16}$$

□

Theorem 7. *Assume that* (1) *has a unique solution* x^**, then Algorithm 1 converges for any initial vector with its convergence rate governing by*

$$\sqrt{\text{cond}^2(A) - 1}$$

Furthermore, the error estimates $\|x(k) - x^*\|_F$ *compared to the previous step and the first step are given as follows:*

$$\|x(k) - x^*\|_F < \sqrt{\text{cond}^2(A) - 1}\|x(k-1) - x^*\|_F, \tag{17}$$

$$\|x(k) - x^*\|_F < \sqrt{\text{cond}^2(A) - 1}^k \|x(0) - x^*\|_F. \tag{18}$$

Proof. Deriving the equations (15) and (16) and applying Lemma 3 we obtain

$$\begin{aligned}
&\|x(k) - x^*\|_F \\
&= \|A^{-1}Ax(k) - A^{-1}Ax^*\|_F \\
&\leqslant \|A^{-1}\|_2 \|Ax(k) - b\|_F \\
&\leqslant \|A^{-1}\|_2 \frac{\sqrt{\text{cond}^2(A) - 1}}{\text{cond}(A)} \|Ax(k-1) - b\|_F \\
&\leqslant \|A^{-1}\|_2 \|A\|_2 \frac{\sqrt{\text{cond}^2(A) - 1}}{\text{cond}(A)} \|x(k-1) - x^*\|_F \\
&= \sqrt{\text{cond}^2(A) - 1} \|x(k-1) - x^*\|_F.
\end{aligned}$$

Hence for each $k \in \mathbb{N}$,

$$\|x(k) - x^*\|_F \leqslant \sqrt{\text{cond}^2(A) - 1}^{\,k} \|x(0) - x^*\|_F.$$

□

Theorem 8. *Let $\{x(k)\}_{k=1}^{\infty}$ be a sequence of vector generated by Algorithm 1. For any $\epsilon > 0$ we have that $\varphi(x(k)) < \epsilon$ after k^* iterations for any k^* that respects:*

$$k^* > \frac{\log \epsilon - \log \varphi(x(0))}{\log c} \tag{19}$$

where $c = 1 - \lambda_{min}/\lambda_{max}$.

Proof. From (14)

$$\varphi(x(k)) \leqslant c^k \varphi(x(0)) \to 0 \text{ as } k \to \infty.$$

This means precisely that for all $\epsilon > 0$, there is a positive integer N such that for all $k \geqslant N$,

$$c^k \varphi(x(0)) < \epsilon$$

Taking logarithm both sides, we obtain (19). □

Corollary 9. *An approximated solution, $x(k)$ will have an accuracy to decimal digit n after k^* iterations for any k^* that respects:*

$$k^* > \frac{\log 0.5 - \log \varphi(x(0)) - n}{\log c} \tag{20}$$

where $c = 1 - \lambda_{min}/\lambda_{max}$.

Proof. Substitute $\epsilon = 0.5 \times 10^{-n}$ in (19). □

Notice that the rate converges to ϵ both as a function of how far our initial point was from the optimal solution, as well as the condition number of the coefficient matrix. As it gets closer to 1, we have the algorithm converges faster as a result.

4 Numerical simulations

In this section, we illustrate the effectiveness and capability of Algorithm 1. We report the comparison of Algorithm 1 with the existing algorithms we have presented in the introduction. We include the table of iteration time and the CPU time as well as the error plot graphic in each example. All iterations have been carried out by MATLAB R2017a, Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz 2.60 GHz, RAM 8.00 GB. PC environment. For convenience, we use TauOpt, GI, LS, BB1, BB2, IT and CPU to represent Algorithm 1, the gradient based algorithm (Lemma 1), the least squares algorithm (Lemma 2), the BB step size type 1 (5)

and type 2 (6), the iteration time and the CPU time respectively. At step kth of the iteration, we consider the following error:

$$\gamma_k := \|x(k) - x^*\|_F$$

where $x(k)$ is the kth solution of the corresponding linear system.

Example 10. *We consider the linear system where*

$$A = \begin{bmatrix} 1 & 2 \\ -2 & -5 \end{bmatrix} \text{ and } b = \begin{bmatrix} -1 \\ 4 \end{bmatrix}.$$

We choose an initial vector

$$x(0) = 10^{-6} \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Running Algorithm 1, we see from Table 1 that the approximated solutions converge to the exact solution

$$x^* = \begin{bmatrix} 3 \\ -2 \end{bmatrix}.$$

Furthermore, Algorithm 1 gives the fastest convergence as Figure 1 and Table 2 shown. Besides, to compare with the GI and LS algorithms, we choose the convergence factor $\mu = 0.025$ and $\mu = 0.05$, respectively.

Example 11. *We consider a larger linear system. We would like to show that for a coefficient matrix that has no appropriate property and make all approximated solutions from every other method diverges, our method still converges to the exact solution. Given*

$$A = \begin{bmatrix} 2 & 4 & 9 & 8 & 1 & 2 \\ 5 & 2 & 7 & 7 & 6 & 5 \\ 9 & 4 & 1 & 2 & 0 & 8 \\ 4 & 7 & 8 & 6 & 7 & 1 \\ 8 & 6 & 6 & 7 & 2 & 0 \\ 6 & 9 & 1 & 0 & 3 & 5 \end{bmatrix} \text{ and } b = \begin{bmatrix} 92 \\ 70 \\ -25 \\ 116 \\ 80 \\ 16 \end{bmatrix}.$$

Table 1: Iterative solution for Example 10

k	x_1	x_2	γ_k
1	-0.2649	-0.6476	3.5339
2	2.9351	-1.9567	0.0780
3	2.9294	-1.9708	0.0764
4	2.9986	-1.9991	0.0017
5	2.9985	-1.9994	0.0017
6	3.0000	-2.0000	0
Solution	3.0000	-2.0000	

Table 2: Iteration time and CPU time for Example 10

Method	IT	CPU
TauOpt	6	0.0469
Jacobi	104	0.0625
GS	54	0.0781
SOR	102	0.0781
ESOR	97	0.0625
JOR	265	0.0469
GI	6713	0.2656
LS	196	0.0469

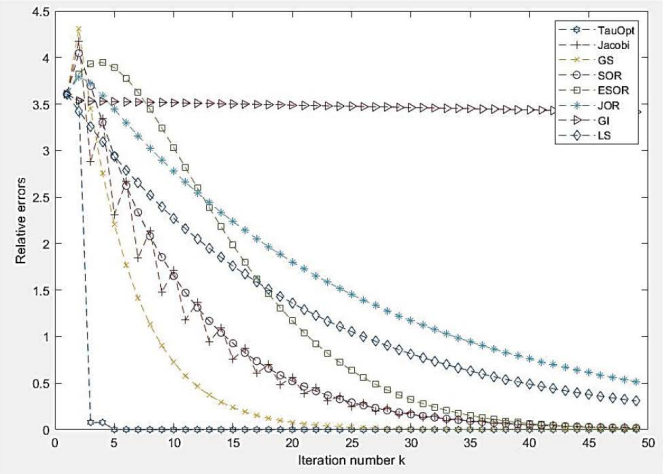

Figure 1: Relative errors for Example 10

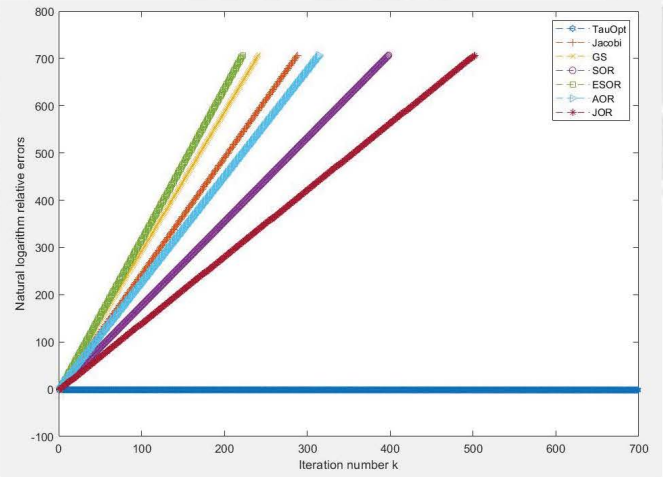

Figure 2: Natural logarithm relative errors for Example 11

We start with an initial vector

$$x(0) = 10^{-6}\begin{bmatrix}1 & -1 & 1 & -1 & 1 & -1\end{bmatrix}^T.$$

In this case, we use natural logarithm error, $\log(\gamma_k)$*, to emphasize how most of the methods but ours diverge as we can see from Figure 2.*

As a result, the approximated solutions from our method converges to the following exact solution with six decimals accuracy using 17119 iterations as shown in Figure 3

$$x^* = \begin{bmatrix}-1 & -3 & 8 & 2 & 4 & -5\end{bmatrix}^T.$$

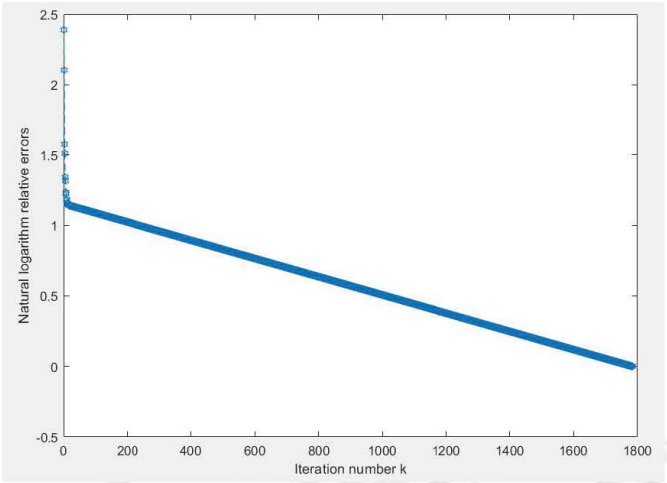

Figure 3: Natural logarithm relative errors of Algorithm 1 for Example 11

Example 12. *This example we consider a* 10×10 *coefficient matrix. We compare Algorithm 1 with GI, LS and BB algorithms. Given*

$$A = \begin{bmatrix} -1 & 2 & -3 & 7 & 6 & 9 & 0 & -5 & -8 & 5 \\ 1 & 5 & -4 & -1 & 0 & 3 & 5 & 8 & -7 & 3 \\ 3 & 4 & -7 & 6 & 0 & 3 & -1 & 7 & 4 & -5 \\ -1 & 1 & 7 & 4 & -9 & -1 & 0 & 0 & -5 & 3 \\ 1 & -7 & 3 & 2 & -4 & 1 & 0 & 5 & 9 & 3 \\ 3 & 1 & 4 & -4 & -6 & 3 & 3 & 6 & -9 & 4 \\ 6 & 1 & 8 & 2 & -3 & -8 & 7 & -4 & 2 & 6 \\ 8 & 1 & 5 & 2 & 3 & 3 & -2 & 8 & 7 & -9 \\ -9 & 5 & 4 & -1 & 0 & 6 & 4 & -8 & 5 & -3 \\ 0 & 1 & -3 & 1 & 6 & -1 & 9 & 5 & -1 & 0 \end{bmatrix} \text{ and } b = \begin{bmatrix} 23 \\ -88 \\ 100 \\ -93 \\ 28 \\ -156 \\ -100 \\ 148 \\ 160 \\ -2 \end{bmatrix}.$$

Choose the initial vector

$$x(0) = 10^{-6} \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}^T.$$

After running Algorithm 1, the exact solution is given by

$$x^* = \begin{bmatrix} -3 & 2 & 1 & 4 & 5 & 7 & -1 & -2 & 9 & -8 \end{bmatrix}^T.$$

To compare with GI and LS algorithms, we choose the convergent factors $\mu = 0.0005$ *and* $\mu = 0.005$ *for GI and LS, respectively. The natural logarithm relative errors shown in Figure 4 implies that Algorithm 1 is comparable to both of BB algorithms but still outperforms the GI and LS algorithms in the performances of convergence. Moreover, Table 3 shows that Algorithm 1 has reached the lowest of CPU time.*

5 Conclusion

A new algorithm, gradient based iterative algorithm with a sequence of optimal convergent factors, is proposed for solving linear systems. By applying gradients and fundamentals of convex optimization, we thus have the algorithm that can be applicable for any linear systems without any conditions but the coefficient matrix must be of invertible. In addition, we introduce a new formula for a convergent factor and call it an optimal convergent factor. It consequently excellently enhance the algorithm in performance of convergence. According

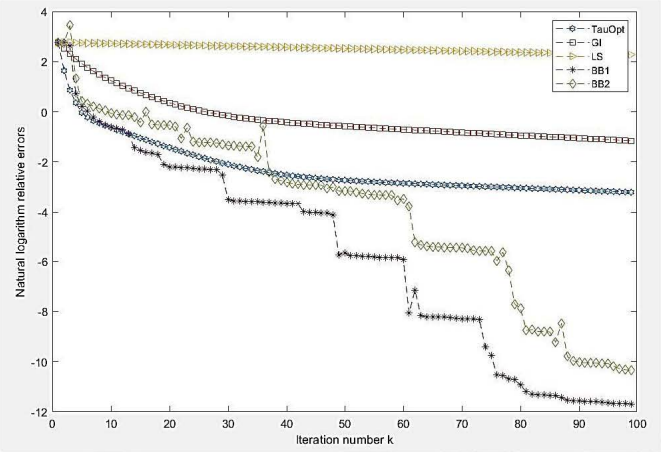

Figure 4: Natural logarithm relative errors for Example 12

to the numerical simulations in Section 4, they illustrate that the proposed algorithm outperforms distinctly all other algorithms mentioned in this paper. Finally, the techniques of gradients and convex optimization might be useful for a class of matrix equations such as Sylvester equation, Lyapunov equation, etc. However, these topics require more studies and can be another further research.

References

[1] Young DM. Iterative Methods for Solving Partial Difference Equations of Elliptical Type. Cambridge, Massachusetts; 1950.

[2] Young DM. Iterative Solution of Large Linear Systems. New York/London: Academic Press; 1971.

[3] Albrechtt P, Klein MP. extrapolated iterative methods for linear systems. SIAM Journal on Numerical Analysis. 1984;21(1):192–201.

[4] Hallett AJH. The convergence of accelerated overrelaxation iterations. mathematics of computation. 1986;47(175):219–223.

[5] Feng D, Tongwen C. Gradient based iterative algorithms for solving a class of matrix equations. IEEE Transactions on Automatic Control. 2005;50(8):1216–1221.

[6] Feng D, Tongwen C. Hierarchical gradient-based identification of multivariable discrete-time systems. Automatica. 2005;41(2):315–325.

[7] Feng D, Tongwen C. Hierarchical least squares identification methods for multivariable systems. IEEE Transactions on Automatic Control. 2005;50(3):397–402. DOI: 10.1109/TAC.2005.843856.

Table 3: Iteration time and CPU time for Example 12

Method	IT	CPU
TauOpt	840	0.0313
GI	6020	0.2813
LS	2531	0.1250
BB1	77	0.0625
BB2	90	0.0625

[8] Feng D, Tongwen C. Iterative least-squares solutions of coupled Sylvester matrix equations. Systems & Control Letters. 2005;54:95–107. DOI: 10.1016/j.sysconle.2004.06.008.

[9] Feng D, Peter XL, Jie D. Iterative solutions of the generalized Sylvester matrix equations by using the hierarchical identification principle. Applied Mathematics and Computation. 2008;197(1):41–50. DOI: 10.1016/j.amc.2007.07.040.

[10] Niu Q, Wang X, Lu Lz. A relaxed gradient based algorithm for solving Sylvester equation. Asian Journal of Control. 2011;13(3):461–464.

[11] Wang X, Dai L, Liao D. A modified gradient based algorithm for solving Sylter equation. Applied Mathematics and Computation. 2012;218:5620–5628.

[12] Xie Y, Ma CF. The accelerated gradient based iterative algorithm for solving a class of generalized Sylvester-transpose matrix equation. Applied Mathematics and Computation. 2016;273:1257–1269. DOI:10.1016/j.amc.2015.07.022.

[13] Zhang X, Sheng X. The relaxed gradient based iterative algorithm for the symmetric (skew symmetric) solution of the Sylvester equation AX+XB=C. Mathematical Problem in Engineering. 2017;2017:1–8. DOI: 10.1155/2017/1624969.

[14] Barzilai J, Borwein J. Two point step size gradient methods. IMA Journal of Numerical Analysis. 1988;8(1):141–148. DOI: 10.1093/imanum/8.1.141.

[15] Yuan Yx. Step-sizes for the gradient method. AMS IP studies in Advanced Mathematics. 2008;42(2):785.

[16] Fletcher R. On the Brazilar-Borwein method. Scotland, UK: University of Dundee; 2001. Research report.

[17] Dai YH, Yuan JY, Yuan Y. Modified two-point step-size gradient methods for unconstrained optimization. Computational Optimization and Applications. 2002;22:103–109.

[18] Dai YH, Fletcher R. On the asymnptotic behaviour of some new gradient methods. Scotland, UK: Department of Mathematics Universityh of Dundee; 2003. Numerical Analysis Report NA/212.

[19] Yuan Y. A new stepsize for the steepest descent method. Chinese Academy of Sciences, China: Institute of Computional Mathematics and Scientific/Engineering Computing; 2004. Research report.

[20] Dai YH, Yuan Y. Analysis of monotone gradient methods. J Industrial and Management Optimization. 2005;1:181–192.

[21] Horn R, Johnson C. Matrix Analysis. New York: Cambridge University Press; 1990.

[22] Stephen PB, Lieven V. Convex Optimization. Cambridge: Cambridge University Press; 2004.

Author Biography

Name	Mr. Adisorn Kittisopaporn
Date of Birth	8 February 1995
Address	116/59 Moo 4, Bangmuang, Muang, Samutprakarn, 10270
Education	2013-2016 Bachelor of Science in Applied Mathematics. GPA 3.92 (First Class Honors) King Mongkut's Institute of Technology Ladkrabang
	2017-2018 Master of Science in Applied Mathematics. GPA 4.00 King Mongkut's Institute of Technology Ladkrabang
Scholarship	2013-2016 King Mongkut's Institute of Technology Ladkrabang Academic Excellent Scholarship
	2017-2018 Faculty of science, King Mongkut's Institute of Technology Ladkrabang Graduate Scholarship

Academic Publication(s)

1. Kittisopaporn, A. and Chansangiam, P. 2019. "Gradient Based Iterative Algorithm with a Sequence of Optimal Convergent Factors for Solving Linear Systems." The 24th Annual Meeting in Mathematics (AMM 2019) Proceedings, p521-532.