Advances in Difference Equations
a SpringerOpen Journal

RESEARCH Open Access

The steepest descent of gradient-based iterative method for solving rectangular linear systems with an application to Poisson's equation

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

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

Abstract

We introduce an effective iterative method for solving rectangular linear systems, based on gradients along with the steepest descent optimization. We show that the proposed method is applicable with any initial vectors as long as the coefficient matrix is of full column rank. Convergence analysis produces error estimates and the asymptotic convergence rate of the algorithm, which is governed by the term $\sqrt{1-\kappa^{-2}}$, where κ is the condition number of the coefficient matrix. Moreover, we apply the proposed method to a sparse linear system arising from a discretization of the one-dimensional Poisson equation. Numerical simulations illustrate the capability and effectiveness of the proposed method in comparison to the well-known and recent methods.

MSC: 15A12; 15A60; 26B25; 65F10; 65N22

Keywords: Rectangular linear system; Iterative method; Gradient; Steepest descent; Condition number; Poisson's equation

1 Introduction

Linear systems play an essential role in modern applied mathematics, including numerical analysis, statistics, mathematical physics/biology, and engineering. In this paper, we develop an effective algorithm for solving rectangular linear systems. The proposed algorithm can be applied for most of the scientific models involving differential equations. As a model problem, we concern Poisson's equation, which arises in many applications, for example, electromagnetics, fluid mechanics, heat flow, diffusion, and quantum mechanics.

Let us consider a (square) linear system $Ax = b$ with given $A \in M_n(\mathbb{R})$ and $b \in \mathbb{R}^n$. Here we denote the set of m-by-n real matrices by $M_{m,n}(\mathbb{R})$, and for square matrices, we set $M_n(\mathbb{R}) = M_{n,n}(\mathbb{R})$. For solving linear systems, iterative methods have received much attention. In principle, iterative methods create a sequence of numerical solutions so that starting from an initial approximation, an iteration with sufficiently large number finally becomes an accurate solution. A group of methods for solving the linear system, called

stationary iterative methods, can be expressed in the simple form

$$x(k+1) = Bx(k) + c,$$

where B is the associated iteration matrix derived from the coefficient matrix A, and c is the vector derived from A and b. The Jacobi method, the Gauss–Seidel (GS) method, and the successive over-relaxation (SOR) method are three classical ones (see, e.g., [1, Ch. 10]) derive by splitting

$$A = D - L - U,$$

where D is a diagonal matrix, and L (U) is a lower (upper) triangular matrix. The SOR method has received much attention and has been evolved continually into new iterative methods, for example:

- Jacobi over-relaxation (JOR) method [2]

$$B = D^{-1}\big(\alpha L + \alpha U + (1-\alpha)D\big), \qquad c = \alpha D^{-1} b, \quad \alpha > 0.$$

- Extrapolated SOR (ESOR) method [3]

$$B = (D - \omega L)^{-1}\big((\tau - \omega)L + \tau U + (1-\tau)D\big),$$
$$c = \tau(D - \omega L)^{-1} b, \quad 0 < |\tau| < \omega < 2.$$

- Accelerated over-relaxation (AOR) method [4]

$$B = (D + \alpha L)^{-1}\big((\alpha - \beta)L - \beta U + (1-\beta)D\big),$$
$$c = \beta(D + \alpha L)^{-1} b, \quad 0 < \alpha < \beta < 2.$$

In the recent decade, many researchers developed gradient-based iterative algorithms for solving matrix equations based on the techniques of hierarchical identification and minimization of associated norm-error functions; see, for example, [5–24]. Convergence analysis for such algorithms relies on the Frobenius norm $\|\cdot\|_F$, the spectral norm $\|\cdot\|_2$, and the condition number respectively defined for each $A \in M_{m,n}(\mathbb{R})$ by

$$\|A\|_F = \sqrt{\operatorname{tr}\big(AA^T\big)}, \qquad \|A\|_2 = \sqrt{\lambda_{\max}\big(A^T A\big)}, \qquad \kappa(A) = \left(\frac{\lambda_{\max}(A^T A)}{\lambda_{\min}(A^T A)}\right)^{1/2}.$$

Moreover, such techniques can be employed for any rectangular linear system of the form

$$Ax = b, \quad A \in M_{m,n}(\mathbb{R}), b \in \mathbb{R}^m. \tag{1}$$

If A is assumed to be of full column rank, then the consistent system (1) has a unique solution $x^* = (A^T A)^{-1} A^T b$. The following algorithms are derived from such techniques.

Proposition 1.1 ([25, 26]) *Consider the linear system* (1) *with full-column-rank matrix* A. *Let* $0 < \mu < 2/\|A\|_2^2$ *or* $0 < \mu < 2/\|A\|_F^2$. *Then the iterative solution* $x(k)$ *given by the gradient-based iterative* (*GI*) *algorithm*

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

converges to a unique solution for any initial value $x(0)$.

Proposition 1.2 ([25, 26]) *Consider the linear system* (1) *with full-column-rank matrix* A. *Let* $0 < \mu < 2$. *Then the iterative solution* $x(k)$ *given by the least-squares iterative* (*LS*) *algorithm*

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

converges to a unique solution for any initial value $x(0)$.

Another study of solving linear systems considers unconstrained convex optimization, where the gradient method along with the steepest descent is used (see, e.g., [27]). The steepest descent is a gradient algorithm where the step size α_k is chosen at each individual iteration to achieve the maximum amount of decrease of the objective function. 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 let $g_k = \nabla f(x_k)$ be the gradient vector at x_k. The steepest descent method defines the next iteration by

$$x_{k+1} = x_k - \alpha_k^* g_k, \tag{4}$$

where $\alpha_k^* > 0$ satisfies

$$f\big(x_k - \alpha_k^* g_k\big) = \min_{\alpha>0} f(x_k - \alpha g_k).$$

Barzilai and Borwein [28] approached the step size in the current iteration. For the iterative equation (4), the step size α_k can be chosen as

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

or

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

where $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$. We call such iterative method the BB method. Convergence analysis of the BB method is provided in [28, 29]. This idea has encouraged and brought about many researches on the gradient method; see, for example, [30–34].

In the present paper, we propose a new gradient-based iterative algorithm with a sequence of optimal convergent factors for solving rectangular linear systems (see Sect. 2).

Then we make convergence analysis for the proposed algorithm, including the convergence rate and error estimates (see Sect. 3). Numerical experiments are provided to illustrate the capability and effectiveness of the proposed algorithm in comparison to all mentioned algorithms (see Sect. 4). We also apply the algorithm to a sparse linear system arising from a discretization of the one-dimensional Poisson equation (see Sect. 5). Finally, we conclude the paper with some remarks in Sect. 6.

2 Proposing the algorithm

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

Consider the rectangular linear system (1) where $A \in M_{m,n}(\mathbb{R})$ is a full-column-rank matrix, $b \in \mathbb{R}^m$ is a known constant vector, and $x \in \mathbb{R}^n$ is an unknown vector. We first define the quadratic norm-error function

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

Since A is of full column rank, the consistent system (1) has a unique solution, and hence an optimal vector x^* of f exists. We will start by having an arbitrary initial vector $x(0)$, and then at every step $k > 0$, we iteratively move to the next vector $x(k+1)$ in an appropriate direction, that is, the negative gradient of f together with a suitable step size τ_{k+1}. Thus the gradient-based iterative method can be described through the following recursive rule:

$$x(k+1) = x(k) - \tau_{k+1}\nabla f\big(x(k)\big). \tag{8}$$

To minimize the function f, we will deduce their gradients in detail. Indeed, we get

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

Thus our iterative equation is of the form

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

To generate the best step size at each iteration, we recall the technique of the steepest descent, which minimizes the error occurring at each iteration. Thus we consider the error $f(x(k+1))$ as a function of $\tau \geq 0$:

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

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

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

To obtain the critical point, we make the differentiation:

$$\begin{aligned} 0 = \frac{d}{d\tau}\phi_{k+1}(\tau) &= \frac{1}{2}\frac{d}{d\tau}\operatorname{tr}\big((\tau\tilde{b}-\tilde{c})(\tau\tilde{b}-\tilde{c})^T\big) \\ &= \frac{1}{2}\frac{d}{d\tau}\operatorname{tr}\big(\tau\tilde{b}\tau\tilde{b}^T - \tau\tilde{b}\tilde{c}^T - \tilde{c}\tau\tilde{b}^T + \tilde{c}\tilde{c}^T\big) \\ &= \frac{1}{2}\big(2\tau\operatorname{tr}\big(\tilde{b}\tilde{b}^T\big) - 2\operatorname{tr}\big(\tilde{b}\tilde{c}^T\big)\big), \end{aligned}$$

which gives $\tau = \operatorname{tr}(\tilde{b}\tilde{c}^T)/\operatorname{tr}(\tilde{b}\tilde{b}^T)$. Note that the second derivative of $\phi_{k+1}(\tau)$ is given by $\operatorname{tr}(\tilde{b}\tilde{b}^T)$, which is positive. Hence the minimizer of the function $\phi_{k+1}(\tau)$ is

$$\tau_{k+1} = \frac{\operatorname{tr}(AA^T(b-Ax(k))(b-Ax(k))^T)}{\operatorname{tr}(AA^T(b-Ax(k))(b-Ax(k))^TAA^T)} = \frac{\|A^T(b-Ax(k))\|_F^2}{\|AA^T(b-Ax(k))\|_F^2}. \tag{9}$$

We call $\{\tau_{k+1}\}_{k=0}^{\infty}$ the sequence of optimal convergent factors. Now we summarize the search direction and optimal step size.

Algorithm 2.1 The steepest descent of gradient-based iterative algorithm.

Input step: Input $A \in M_{m,n}(\mathbb{R})$ and $b \in \mathbb{R}^m$.

Initializing step: Choose an initial vector $x(0) \in \mathbb{R}^n$. Set $k := 0$. Compute $c = A^Tb$, $C = A^TA$, $d = Ac$, $D = AC$.

Updating step:

$\tau_{k+1} = \sum_{i=1}^n (c_i - \sum_{j=1}^n C_{ij}x_j(k))^2 / \sum_{i=1}^m (d_i - \sum_{j=1}^n D_{ij}x_j(k))^2$,

$x(k+1) = x(k) + \tau_{k+1}(c - Cx(k))$.

Set $k := k+1$ and repeat the updating step.

Here we denote by c_i the ith entry of c and by C_{ij} the (i,j)th entry of C. In case of stopping the algorithm, a stopping criteria is necessary and can be described as $\|b - Ax(k)\|_F < \epsilon$, where ϵ is a small positive number. Note that we introduce the vectors c, d and the matrices C, D to avoid duplicated computations.

3 Convergence analysis

In this section, we show that Algorithm 2.1 converges to the exact solution for any initial vector. Moreover, we provide the convergence rate, error estimates, and the iteration number corresponding to a given satisfactory error.

3.1 Convergence of the algorithm

The convergence analysis is based on a matrix partial order and strongly convex functions. Recall that the Löwner partial order $\preceq$ for real symmetric matrices is defined by $A \preceq B$ if $B-A$ is a positive semidefinite matrix or, equivalently, $x^TAx \le x^TBx$ for all $x \in \mathbb{R}^n$. A twice-differentiable convex function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be strongly convex if there exist constants $0 \le m < M$ 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

$$my^Ty \le y^T\nabla^2 f(x)y \le My^Ty \quad \text{for all } x, y \in \mathbb{R}^n.$$

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

Lemma 3.1 ([35]) *If f is strongly convex on $\mathbb{R}^n$, then for any $x, y \in \mathbb{R}^n$,*

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

Theorem 3.2 *If system* (1) *is consistent and A is of full column rank, then the sequence $\{x(k)\}$ generated by Algorithm* 2.1 *converges to a unique solution for any initial vector $x(0)$.*

Proof The hypothesis implies the existence of a unique solution x^* for the system. We will show that $x(k) \to x^*$ as $k \to \infty$. In case the gradient $\nabla f(x(k))$ becomes the zero vector for some k, we have $x(k) = x^*$, 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 positive semidefinite matrix, we have

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

Thus f is strongly convex. For convenience, we write $\lambda_{\min}$ and $\lambda_{\max}$ instead of $\lambda_{\min}(A^T A)$ and $\lambda_{\max}(A^T A)$, respectively. We consider the function $\phi_{k+1}(\tau)$ of the step size τ. Applying Lemma 3.1, we obtain

$$f\big(x(k+1)\big) \leq f\big(x(k)\big) - \tau \big\|\nabla f\big(x(k)\big)\big\|_F^2 + \frac{\lambda_{\max}\tau^2}{2}\big\|\nabla f\big(x(k)\big)\big\|_F^2.$$

Minimize this inequality over τ. The right-hand side (RHS) is minimized by $\tau_{k+1} = 1/\lambda_{\max}$, and thus

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

From the other inequality in Lemma 3.1 we have

$$f\big(x(k+1)\big) \geq f\big(x(k)\big) - \tau \big\|\nabla f\big(x(k)\big)\big\|_F^2 + \frac{\lambda_{\min}\tau^2}{2}\big\|\nabla f\big(x(k)\big)\big\|_F^2.$$

We find that $\tau = 1/\lambda_{\min}$ minimizes the RHS, that is,

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

Now $\|\nabla f(x(k))\|_F^2 \geq 2\lambda_{\min} f(x(k))$, and hence by (11) we have

$$f\big(x(k+1)\big) \leq \left(1 - \frac{\lambda_{\min}}{\lambda_{\max}}\right) f\big(x(k)\big).$$

Since A is a full-column-rank matrix, the matrix A^TA is positive definite, that is, $\lambda > 0$ for all $\lambda \in \sigma(A^TA)$. It follows that $c := 1 - \lambda_{\min}/\lambda_{\max} < 1$ and

$$f\big(x(k+1)\big) \le cf\big(x(k)\big). \tag{12}$$

By induction we obtain that for any $k \in \mathbb{N}$,

$$f\big(x(k)\big) \le c^k f\big(x(0)\big), \tag{13}$$

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

3.2 Convergence rate and error estimates

From now on, denote $\kappa = \kappa(A)$, the condition number of A. According to the proof of Theorem 3.2, bounds (12) and (13) give rise to the following estimates:

$$\big\|Ax(k) - b\big\|_F \le \big(1 - \kappa^{-2}\big)^{\frac{1}{2}} \big\|Ax(k-1) - b\big\|_F, \tag{14}$$

$$\big\|Ax(k) - b\big\|_F \le \big(1 - \kappa^{-2}\big)^{\frac{k}{2}} \big\|Ax(0) - b\big\|_F. \tag{15}$$

Theorem 3.3 *Assume the hypothesis of Theorem 3.2. The asymptotic convergence rate of the Algorithm 2.1 (with respect to the certain 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 (14) and (15), respectively. In particular, the relative error at each iteration gets smaller than the previous (nonzero) one.*

Now we recall the following properties.

Lemma 3.4 (e.g. [1]) *For any matrices A and B of proper sizes, we have*

(i) $\|A^T\|_2 = \|A\|_2$,
(ii) $\|A^TA\|_2 = \|A\|_2^2$,
(iii) $\|AB\|_F \le \|A\|_2\|B\|_F$.

Theorem 3.5 *Assume the hypothesis of Theorem 3.2. Then the error estimates $\|x(k) - x^*\|_F$ compared to the previous step and the first step of Algorithm 2.1 are given as follows*:

$$\big\|x(k) - x^*\big\|_F \le \kappa\sqrt{\kappa^2 - 1}\big\|x(k-1) - x^*\big\|_F, \tag{16}$$

$$\big\|x(k) - x^*\big\|_F \le \kappa^2\big(1 - \kappa^{-2}\big)^{\frac{k}{2}} \big\|x(0) - x^*\big\|_F. \tag{17}$$

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

Proof By (15) and Lemma 3.4 we obtain

$$\begin{aligned}
\big\|x(k) - x^*\big\|_F &= \big\|\big(A^TA\big)^{-1}\big(A^TA\big)x(k) - \big(A^TA\big)^{-1}\big(A^TA\big)x^*\big\|_F \\
&\le \big\|\big(A^TA\big)^{-1}\big\|_2\big\|A^T\big\|_2\big\|Ax(k) - b\big\|_F \\
&\le \big(1 - \kappa^{-2}\big)^{\frac{k}{2}}\big\|\big(A^TA\big)^{-1}\big\|_2\big\|A^T\big\|_2\big\|Ax(0) - Ax^*\big\|_F
\end{aligned}$$

$$
\begin{aligned}
&\leq \left(1-\kappa^{-2}\right)^{\frac{k}{2}} \left\|\left(A^TA\right)^{-1}\right\|_2 \left\|A^T\right\|_2 \|A\|_2 \left\|x(0)-x^*\right\|_F \\
&= \left(1-\kappa^{-2}\right)^{\frac{k}{2}} \frac{\lambda_{\max}(A^TA)}{\lambda_{\min}(A^TA)} \left\|x(0)-x^*\right\|_F \\
&= \kappa^2\left(1-\kappa^{-2}\right)^{\frac{k}{2}} \left\|x(0)-x^*\right\|_F.
\end{aligned}
$$

Since the end behavior of this error depends on the term $(1-\kappa^{-2})^{\frac{k}{2}}$, the asymptotic rate of convergence for the algorithm is governed by $\sqrt{1-\kappa^{-2}}$. Similarly, from (14) and Lemma 3.4 we have

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

and thus we get (16). □

Hence the condition number κ of the coefficient matrix determines the asymptotic rate of convergence, as well as how far our initial point was from the exact solution. As κ gets closer to 1, the algorithm converges faster.

Proposition 3.6 *Let $\{x(k)\}_{k=1}^{\infty}$ be the sequence of vectors generated by Algorithm 2.1. For each $\epsilon > 0$, we have $\|Ax(k)-b\|_F < \epsilon$ after k^* iterations for any*

$$
k^* > \frac{\log\epsilon - \log\|Ax(0)-b\|_F}{\log\left(1-\kappa^{-2}\right)}. \tag{18}
$$

Besides, for each $\epsilon > 0$, we have $\|x(k)-x^*\|_F < \epsilon$ after k^* iterations for any

$$
k^* > \frac{2(\log\epsilon - 2\log\kappa - \log\|x(0)-x^*\|_F)}{\log\left(1-\kappa^{-2}\right)}. \tag{19}
$$

Proof From (13) we have $\|Ax(k)-b\|_F \leq (1-\kappa^{-2})^k \|Ax(0)-b\|_F \to 0$ as $k \to \infty$. This means precisely that for each $\epsilon > 0$, there is a positive integer N such that for all $k \geq N$,

$$
\left(1-\kappa^{-2}\right)^k \left\|Ax(0)-b\right\|_F < \epsilon.
$$

Taking the logarithm on both sides, we obtain (18). Another result can be obtained in a similar manner; here we start with approximation (17). □

From Proposition 3.6 we obtain the iteration number such that the relative error $\|Ax(k)-b\|_F$ and the exact error $\|x(k)-x^*\|_F$ have an accuracy to a decimal digit after iterations. Indeed, if p is a satisfactory decimal digit, then the desired iteration number is obtained by substituting $\epsilon = 0.5 \times 10^{-p}$.

Remark 3.7 A sharper bound for error estimation is obtained when the coefficient matrix A is a square matrix. However, the convergence rate is governed by the same value. Indeed,

the condition that A is of full column rank is equivalent to the invertibility of A. Using Lemma 3.4, we have the following bound:

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

Similarly, we get $\|x(k)-x^*\|_F \le \sqrt{\kappa^2-1}\|x(k-1)-x^*\|_F$. Since $\kappa \ge 1$, these bounds are sharper than those in (16) and (17).

4 Numerical simulations for linear systems

In this section, we illustrate the effectiveness and capability of our algorithm. We report the comparison of TauOpt, our proposed algorithm, with the existing algorithms we have presented in the introduction, that is, GI (Proposition 1.1), LS (Proposition 1.2), BB1 (5), and BB2 (6). All iteration results have been carried out by MATLAB R2018a in Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz, RAM 8.00 GB PC environment. To measure the computational time taken for each program, we apply the *tic* and *toc* functions in MATLAB and abbreviate them CT. The readers are recommended to consider all reported results, such as errors, CTs, figures, while comparing the performance of any algorithms. For each example, unless otherwise stated, we consider the following error at the kth iteration step:

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

Example 4.1 We consider the linear system $Ax = b$ with

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

We choose an initial vector $x(0) = 10^{-6}[1\ -1]^T$. Running Algorithm 2.1, we see from Table 1 that the approximated solutions converge to the exact solution $x^* = [-3\ 4]^T$. Fig. 1 and Table 2 show the results when running 100 iterations. We can conclude that Algorithm 2.1 gives the fastest convergence.

Table 1 Iterative solution for Ex. 4.1

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	

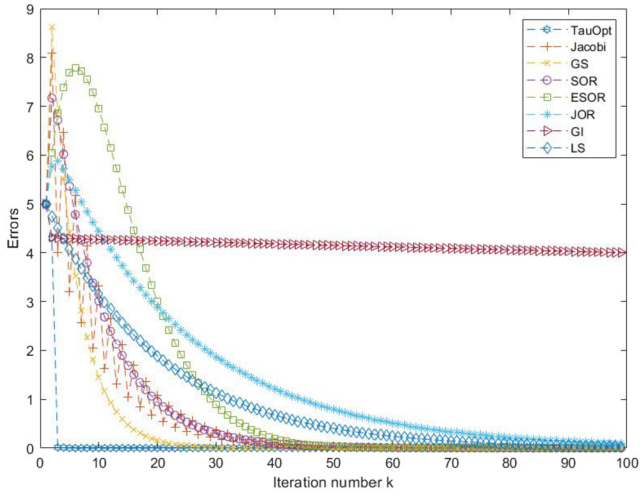

Figure 1 Error for Ex. 4.1

Table 2 Error and CT for Ex. 4.1

Method	Error	CT
TauOpt	8.8818e-16	0.0095
Jacobi	8.9203e-05	0.0050
GS	3.4281e-09	0.0041
SOR	1.0415e-04	0.0045
ESOR	7.7386e-05	0.0050
JOR	0.0954	0.0190
GI	4.0029	0.0025
LS	0.0328	0.0038

Table 3 Condition number, iteration time (IT), and CT for Ex. 4.2

a	$\kappa(A)$	IT	CT
−3	1.3504	11	1.7944e-12
7	2.9802	19	3.0440e-11
10	3.8089	27	0.0013
15	5.9720	36	0.0049

Example 4.2 In this example, we consider the convergence rate 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 shown in Table 3 and Fig. 2. The simulations reveal that the closer to 1 the condition number, the faster the convergence of the algorithm. This shows the correctness of Theorems 3.3 and 3.5.

Example 4.3 We consider a larger linear system. We would like to show that for a coefficient matrix that has no appropriate property and makes all approximated solutions from

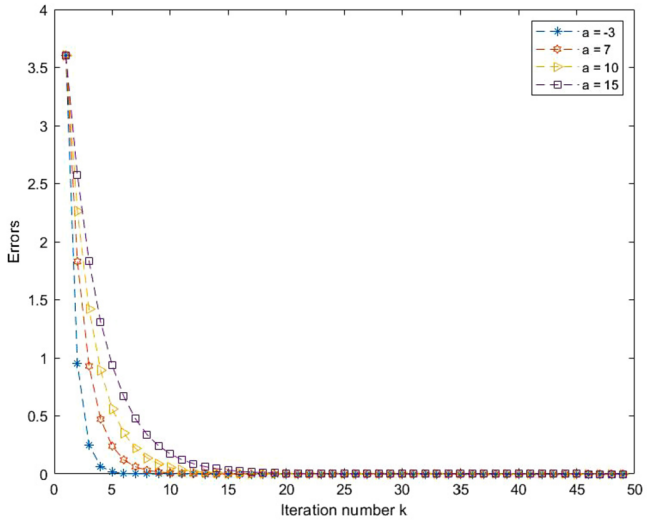

Figure 2 Error for Ex. 4.2

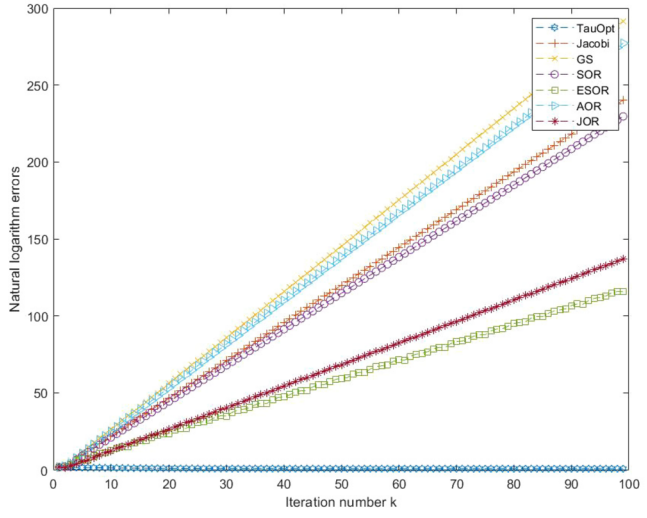

Figure 3 Natural logarithm errors for Ex. 4.3

every other method diverge, our method still converges to the exact solution. Let

$$A = \begin{bmatrix} 1 & 5 & 8 & 4 & 8 & 5 \\ 5 & 2 & 7 & 7 & 6 & 5 \\ 8 & 7 & 9 & 8 & 6 & 4 \\ 4 & 7 & 8 & 6 & 7 & 1 \\ 8 & 6 & 6 & 7 & 2 & 0 \\ 5 & 5 & 4 & 1 & 0 & 2 \end{bmatrix}, \qquad b = \begin{bmatrix} -6 \\ -3 \\ -13 \\ 9 \\ -4 \\ -30 \end{bmatrix}, \qquad x(0) = 10^{-6} \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}.$$

As we can see from Fig. 3, the natural logarithm errors $\log \|x(k) - x^*\|_F$ for Jacobi, GS, SOR, ESOR, AOR, and JOR diverge, whereas those for our method continue to converge

to 0. As a result, the approximated solutions from Algorithm 2.1 converge to the exact solution

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

with six decimals accuracy using 14,612 iterations and CT = 0.3627.

Example 4.4 In this example, we consider a rectangular linear system where its coefficient matrix is of full column rank. We compare Algorithm 2.1 with GI, LS, and BB algorithms. Let

$$A = \begin{bmatrix} 1 & 3 & -2 & 9 & 0 & 4 & 3 & -9 \\ 2 & -3 & 1 & 0 & 8 & 4 & -1 & 6 \\ 3 & 4 & 5 & 1 & 0 & 0 & 7 & -8 \\ -4 & 1 & 3 & 5 & 9 & 4 & -1 & -2 \\ -9 & 8 & 3 & 0 & -5 & 4 & 1 & -3 \\ 4 & 1 & 1 & 5 & 8 & -5 & 4 & 9 \\ 11 & 3 & 5 & 7 & -7 & 3 & 5 & 2 \\ -4 & 3 & 1 & 0 & -1 & 2 & 7 & 5 \\ 2 & 1 & 3 & 5 & 7 & 12 & -9 & -3 \\ 1 & 2 & 3 & -4 & 1 & 0 & 5 & 7 \end{bmatrix}, \quad b = \begin{bmatrix} 34 \\ 52 \\ 35 \\ 33 \\ -98 \\ 15 \\ 28 \\ -67 \\ 93 \\ -26 \end{bmatrix}, \quad x(0) = 10^{-6} \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}.$$

The exact solution of the system $Ax = b$ is given by

$$x^* = \begin{bmatrix} 7 & -4 & 1 & 0 & 5 & 2 & -1 & -4 \end{bmatrix}^T.$$

The results of running 100 iterations are provided in Fig. 4 and Table 4. Both show that Algorithm 2.1 outperforms the GI and LS algorithms. On the other hand, both types 1 and 2 of the BB algorithm of are comparable with ours. The BB algorithm gives a better iteration time; however, our algorithm gives a better computation time.

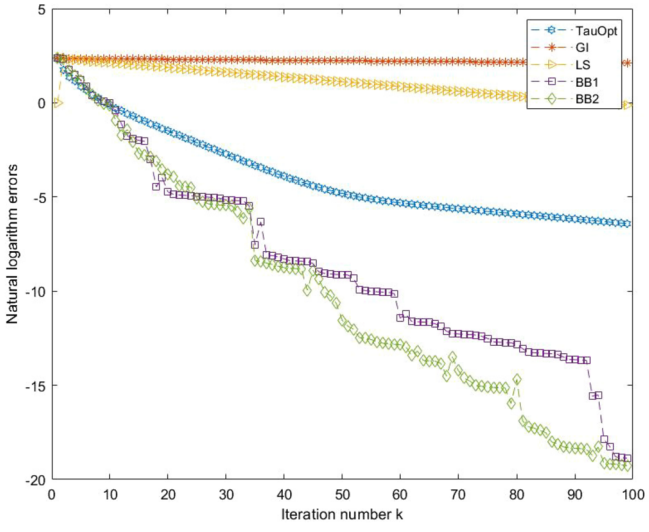

Figure 4 Errors for Ex. 4.4

Table 4 Error & CT for Ex. 4.4

Method	Error	CT
TauOpt	0.0016	6.4740e-04
GI	8.3179	0.0063
LS	0.8852	0.0140
BB1	6.4026e-09	0.0072
BB2	4.2386e-09	0.0027

Table 5 Error & CT for Ex. 4.5

Method	Error	CT
TauOpt	6.9368	0.2359
GI	4.7083e+03	0.1328
LS	1.5904e+03	0.6062

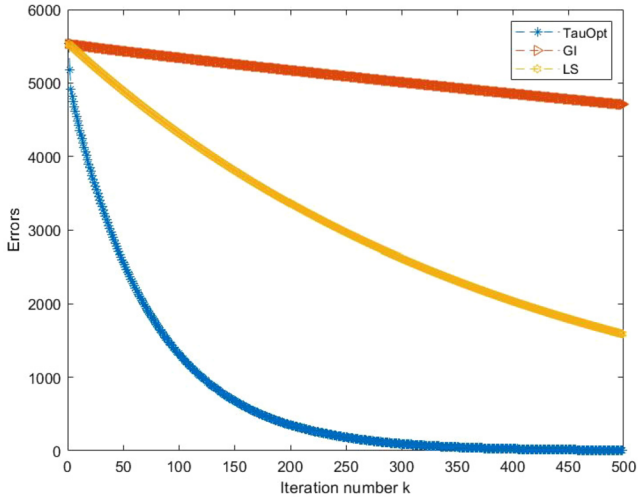

Figure 5 Relative errors for Ex. 4.5

Example 4.5 For this example, we use the sparse 100×100 matrix

$$A = M + 2rN + \frac{100}{(n+1)^2} I,$$

where $M = \text{tridiag}(-1, 2, -1)$, $N = \text{tridiag}(0.5, 0, -0.5)$, $r = 0.01$, and $n = 16$ as in [36]. We choose an initial vector $x(0) = [x_i] \in \mathbb{R}^{100}$, where $x_i = 10^{-6}$ for all $i = 1, \ldots, 100$. We take a random vector $b \in \mathbb{R}^{100}$ with every element in $[-100, 100]$. Since the exact solution is not yet known, it is appropriate to consider the relative error $\|b - Ax(k)\|_F$. The numerical results after 500 iterations in comparing our algorithm with GI and LS algorithm are shown in Table 5 and Fig. 5. They reveal that our algorithm performs better than GI and LS methods.

5 Application to one-dimensional Poisson's equation

We now discuss an application of the proposed algorithm to a sparse linear system arising from the one-dimensional Poisson equation

$$-u'' = f. \tag{20}$$

Here $u : (\alpha, \beta) \to \mathbb{R}$ is an unknown function to be approximated, and $f : (\alpha, \beta) \to \mathbb{R}$ is a given function. The function u must satisfy the Dirichlet boundary conditions $u(\alpha) = u(\beta) = 0$. We discretize the problem to solve an approximate solution at N partitioned points x_i between α and β: $x_i = \alpha + ih$, where $h = (\beta - \alpha)/(N + 1)$ and $0 \le i \le N + 1$. We denote $u_i = u(x_i)$ and $f_i = f(x_i)$. By the centered 2nd-order finite difference approximation we obtain

$$-\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = f(x_i), \quad i = 1, \dots, N.$$

Now we can put it into a linear system $T_N u = h^2 f$, where $u = [u_1\ u_2\ \dots\ u_N]$ is an unknown vector, and the coefficient matrix $T_N = \text{tridiag}(-1, 2, -1) \in M_N(\mathbb{R})$ is a tridiagonal matrix. Now the proposed algorithm for this sparse linear system is presented as follows.

Algorithm 5.1 The steepest descent of gradient-based iterative algorithm for solving one-dimensional Poisson's equation.

Input step: Input $N \in \mathbb{N}$ as a number of partition.

Initializing step: Let $h = (\beta - \alpha)/(N + 1)$. For each $i = 1, \dots, N$, set $x(i) = \alpha + ih$ and $f(i) = f(x(i))$. Compute $g = h^2 f$, $s = T_N g$, $S = T_N^2$, $t = T_N s$, and $T = T_N S$, where $T_N = \text{tridiag}(-1, 2, -1) \in M_N(\mathbb{R})$. Choose an initial vector $u(0) \in \mathbb{R}^N$ and set $k := 0$.

Updating step:

$\tau_{k+1} = \sum_{i=1}^N (s_i - \sum_{j=1}^N S_{ij} u_j(k))^2 / \sum_{i=1}^N (t_i - \sum_{j=1}^N T_{ij} u_j(k))^2$,

$u(k + 1) = u(k) + \tau_{k+1}(s - Su(k))$.

Set $k := k + 1$ and repeat the updating step.

Here the stopping criteria is $\|g - T_N u(k)\|_F < \epsilon$, where ϵ is a small positive number. Since the coefficient matrix T_N is a sparse matrix, the error norm can be described more precisely:

$$\begin{aligned}
\left\|g - T_N u(k)\right\|_F^2 &= \|g\|_F^2 - 2\,\text{tr}\left(g^T T_N u(k)\right) + \left\|T_N u(k)\right\|_F^2 \\
&= \|g\|_F^2 - h^2\left(2\sum_{i=1}^{N} u_i(k) f_i - \sum_{i=1}^{N-1} u_i(k) f_{i+1} - f_i u_{i+1}(k)\right) \\
&\quad + 5u_1^2(k) + 5u_N^2(k) + 6\sum_{i=2}^{N-1} u_i^2(k) - 8\sum_{i=1}^{N-1} u_i(k) u_{i+1}(k) \\
&\quad + 2\sum_{i=1}^{N-2} u_i(k) u_{i+2}(k).
\end{aligned}$$

The eigenvalues of T_N are given by $\lambda_j = 2(1 - \cos\frac{j\pi}{N+1})$ for $j = 1, \dots, N$; see, for example, [1]. The smallest eigenvalues of T_N can be approximated by the second-order Taylor approximation:

$$\lambda_1 = 2\left(1 - \cos\frac{\pi}{N+1}\right) \approx \left(\frac{\pi}{N+1}\right)^2.$$

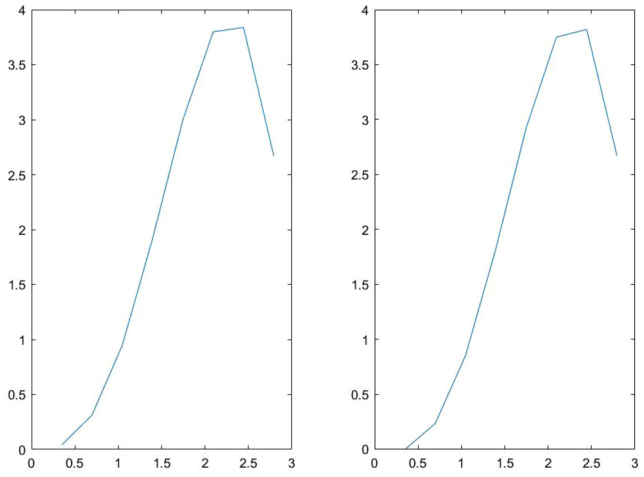

Figure 6 The analytical solution (left) and the numerical solution (right) for Ex. 5.3

Thus T_N is positive definite with condition number

$$\kappa = \frac{\lambda_N}{\lambda_1} = \frac{1-\cos\frac{N\pi}{N+1}}{1-\cos\frac{\pi}{N+1}} \approx \frac{4}{\pi^2}(N+1)^2 \quad \text{for large } N. \tag{21}$$

Now, according to Remark 3.7, the convergence analysis of Algorithm 5.1 can be described as follows.

Corollary 5.2 *The discretization $T_N u = h^2 f$ of the Poisson equation (20) can be solved using Algorithm 5.1 so that the approximated solution $u(k)$ converges to the exact solution u^* for any initial vector $u(0)$. The asymptotic convergence rate of the algorithm is governed by $\sqrt{1-\kappa^{-2}}$, where κ is given by (21).*

Thus the convergence rate of the algorithm depends on the number N of partition.

Example 5.3 We consider an application of our algorithm to the one-dimensional Poisson equation (20) with

$$f(x) = \left(x^2 - 2\right)\sin x - 4x\cos x, \quad 0 < x < \pi,$$

and $u = 0$ on the boundary of $[0, \pi]$. We choose an initial vector $u(0) = 2 \times$ ones, where ones is the matrix that contains 1 at every position. We run Algorithm 5.1 with 8 partitioned points, so that the size of the matrix T_N is 64×64. The analytical solution is

$$u^*(x) = x^2 \sin x.$$

Figure 6 shows the result of our algorithm (right) compared to the analytical solution (left) after running 1000 iterations with CT = 0.0112 seconds.

6 Conclusion

A new algorithm, the steepest descent of gradient-based iterative algorithm, is proposed for solving rectangular linear systems. The algorithm is applicable for any rectangular

linear systems and any initial points without any conditions, but the coefficient matrix is of full column rank. We use an optimization technique to obtain a new formula for a convergence factor, so that it excellently enhances 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 the coefficient matrix. The numerical simulations in Sect. 4 illustrate the applicability and efficiency of the algorithm compared to all other algorithms mentioned in this paper. The iteration number and the CT indicate that our algorithm is a good choice for solving linear systems. Moreover, the sparse linear system arising from the one-dimensional Poisson equation can be solved efficiently using this algorithm. In our opinion, the techniques of gradients, steepest descent, and convex optimization might be useful for a class of matrix equations such as Lyapunov equation, Sylvester equation, and so on. However, these topics require more studies and can be another further research.

Acknowledgements
This work was supported by Thailand Science Research and Innovation (Thailand Research Fund).

Funding
This second author expresses his gratitude to Thailand Science Research and Innovation (Thailand Research Fund), Grant No. MRG6280040, for financial supports.

Availability of data and materials
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
Both authors contributed equally and significantly in writing this article. Both authors read and approved the final manuscript.

Publisher's Note

Received: 26 February 2020 Accepted: 21 May 2020 Published online: 01 June 2020

References

1. James, W.D.: Applied Numerical Linear Algebra. Society for Industrial and Applied Mathematics, Philadelphia (1997)
2. Young, D.M.: Iterative Solution of Large Linear Systems. Academic Press, New York (1971)
3. Albrechtt, P., Klein, M.P.: Extrapolated iterative methods for linear systems. SIAM J. Numer. Anal. **21**(1), 192–201 (1984)
4. Hallett, A.J.H.: The convergence of accelerated overrelaxation iterations. Math. Comput. **47**(175), 219–223 (1986). https://doi.org/10.2307/2008090
5. Ding, F., Chen, T.: Gradient based iterative algorithms for solving a class of matrix equations. IEEE Trans. Autom. Control **50**(8), 1216–1221 (2005). https://doi.org/10.1109/TAC.2005.852558
6. Ding, F., Chen, T.: Hierarchical gradient-based identification of multivariable discrete-time systems. Automatica **41**(2), 315–325 (2005). https://doi.org/10.1016/j.automatica.2004.10.010
7. Ding, F., Chen, T.: Hierarchical least squares identification methods for multivariable systems. IEEE Trans. Autom. Control **50**(3), 397–402 (2005). https://doi.org/10.1109/TAC.2005.843856
8. Ding, F., Liu, P.X., Ding, J.: Iterative solutions of the generalized Sylvester matrix equations by using the hierarchical identification principle. Appl. Math. Comput. **197**(1), 41–50 (2008). https://doi.org/10.1016/j.amc.2007.07.040
9. Niu, Q., Wang, X., Lu, L.Z.: A relaxed gradient based algorithm for solving Sylvester equation. Asian J. Control **13**(3), 461–464 (2011). https://doi.org/10.1002/asjc.328
10. Wang, X., Dai, L., Liao, D.: A modified gradient based algorithm for solving Sylvester equation. Appl. Math. Comput. **218**(9), 5620–5628 (2012). https://doi.org/10.1016/j.amc.2011.11.055
11. Xie, Y., Ma, C.F.: The accelerated gradient based iterative algorithm for solving a class of generalized Sylvester-transpose matrix equation. Appl. Math. Comput. **273**(15), 1257–1269 (2016). https://doi.org/10.1016/j.amc.2015.07.022
12. Zhang, X., Sheng, X.: The relaxed gradient based iterative algorithm for the symmetric (skew symmetric) solution of the Sylvester equation $AX + XB = C$. Math. Probl. Eng. **2017**, Article ID 1624969 (2017). https://doi.org/10.1155/2017/1624969
13. Sheng, X., Sun, W.: The relaxed gradient based iterative algorithm for solving matrix equations. Comput. Math. Appl. **74**(3), 597–604 (2017). https://doi.org/10.1016/j.camwa.2017.05.008
14. Sheng, X.: A relaxed gradient based algorithm for solving generalized coupled Sylvester matrix equations. J. Franklin Inst. **355**(10), 4282–4297 (2018). https://doi.org/10.1016/j.jfranklin.2018.04.008

15. Li, M., Liu, X., Ding, F.: The gradient based iterative estimation algorithms for bilinear systems with autoregressive noise. Circuits Syst. Signal Process. **36**(11), 4541–4568 (2017). https://doi.org/10.1007/s00034-017-0527-4
16. Sun, M., Wang, Y., Liu, J.: Two modified least-squares iterative algorithms for the Lyapunov matrix equations. Adv. Differ. Equ. **2019**(1), Article ID 305 (2019). https://doi.org/10.1186/s13662-019-2253-7
17. Zhu, M.Z., Zhang, G.F., Qi, Y.E.: On single-step HSS iterative method with circulant preconditioner for fractional diffusion equations. Adv. Differ. Equ. **2019**(1), Article ID 422 (2019). https://doi.org/10.1186/s13662-019-2353-4
18. Zhang, H.M., Ding, F.: A property of the eigenvalues of the symmetric positive definite matrix and the iterative algorithm for coupled Sylvester matrix equations. J. Franklin Inst. **351**(1), 340–357 (2014). https://doi.org/10.1016/j.jfranklin.2013.08.023
19. Zhang, H.M., Ding, F.: Iterative algorithms for $X + A^T X^{-1} A = I$ by using the hierarchical identification principle. J. Franklin Inst. **353**(5), 1132–1146 (2016). https://doi.org/10.1016/j.jfranklin.2015.04.003
20. Ding, F., Zhang, H.: Brief paper – Gradient-based iterative algorithm for a class of the coupled matrix equations related to control systems. IET Control Theory Appl. **8**(15), 1588–1595 (2014). https://doi.org/10.1049/iet-cta.2013.1044
21. Xie, L., Ding, J., Ding, F.: Gradient based iterative solutions for general linear matrix equations. Comput. Math. Appl. **58**(7), 1441–1448 (2009). https://doi.org/10.1016/j.camwa.2009.06.047
22. Xie, L., Liu, Y.J., Yang, H.Z.: Gradient based and least squares based iterative algorithms for matrix equations $AXB + CX^T D = F$. Appl. Math. Comput. **217**(5), 2191–2199 (2010). https://doi.org/10.1016/j.amc.2010.07.019
23. Ding, F., Lv, L., Pan, J., et al.: Two-stage gradient-based iterative estimation methods for controlled autoregressive systems using the measurement data. Int. J. Control. Autom. Syst. **18**, 886–896 (2020). https://doi.org/10.1007/s12555-019-0140-3
24. Ding, F., Xu, L., Meng, D., et al.: Gradient estimation algorithms for the parameter identification of bilinear systems using the auxiliary model. J. Comput. Appl. Math. **369**, Article ID 112575 (2020). https://doi.org/10.1016/j.cam.2019.112575
25. Ding, F., Chen, T.: Iterative least-squares solutions of coupled Sylvester matrix equations. Syst. Control Lett. **54**(2), 95–107 (2005). https://doi.org/10.1016/j.sysconle.2004.06.008
26. Ding, F., Chen, T.: On iterative solutions of general coupled matrix equations. SIAM J. Control Optim. **44**(6), 2269–2284 (2006). https://doi.org/10.1137/S0363012904441350
27. Edwin, K.P.C., Stanislaw, H.Z.: An Introduction to Optimization, 2nd edn. Wiley-Interscience, New York (2001)
28. Barzilai, J., Borwein, J.: Two point step size gradient methods. IMA J. Numer. Anal. **8**(1), 141–148 (1988). https://doi.org/10.1093/imanum/8.1.141
29. Yuan, Y.X.: Step-sizes for the gradient method. AMS/IP Stud. Adv. Math. **42**, 785–797 (2008)
30. Dai, Y.H., Yuan, J.Y., Yuan, Y.: Modified two-point step-size gradient methods for unconstrained optimization. Comput. Optim. Appl. **22**, 103–109 (2002). https://doi.org/10.1023/A:1014838419611
31. Dai, Y.H., Fletcher, R.: On the asymptotic behaviour of some new gradient methods. Numerical analysis report NA/212, Department of Mathematics, University of Dundee, Scotland, UK (2003)
32. Dai, Y.H., Yuan, Y.: Analysis of monotone gradient methods. J. Ind. Manag. Optim. **1**(2), 181–192 (2005). https://doi.org/10.3934/jimo.2005.1.181
33. Fletcher, R.: On the Brazilar–Borwein method. Research report, University of Dundee, Scotland, UK (2001)
34. Yuan, Y.: A new stepsize for the steepest descent method. Research report, Institute of Computional Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences, China (2004)
35. Stephen, P.B., Lieven, V.: Convex Optimization. Cambridge University Press, Cambridge (2004)
36. Zhong, Z.B.: On Hermitian and skew-Hermitian spliting iteration methods for continuous Sylvester equations. J. Comput. Math. **29**(2), 185–198 (2011). https://doi.org/10.4208/jcm.1009-m3152