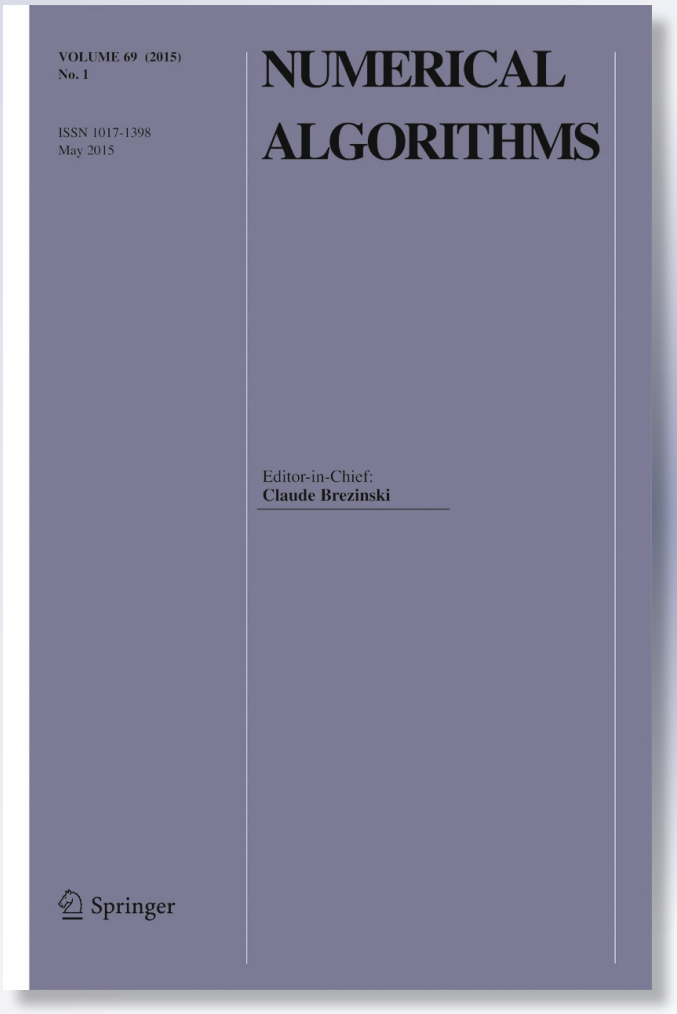
The inexact fixed matrix iteration for solving large linear inequalities in a least squares sense

Yuan Lei

Numerical Algorithms

ISSN 1017-1398
Volume 69
Number 1

Numer Algor (2015) 69:227-251
DOI 10.1007/s11075-014-9892-2



Springer

Numer Algor (2015) 69:227–251
DOI 10.1007/s11075-014-9892-2

ORIGINAL PAPER

The inexact fixed matrix iteration for solving large linear inequalities in a least squares sense

Yuan Lei

Received: 13 August 2013 / Accepted: 1 July 2014 / Published online: 16 July 2014

Abstract A fixed matrix iteration algorithm was proposed by A. Dax (Numer. Algor. 50, 97–114 2009) for solving linear inequalities in a least squares sense. However, a great deal of computation for this algorithm is required, especially for large-scale problems, because a least squares subproblem should be solved accurately at each iteration. We present a modified method, the inexact fixed iteration method, which is a generalization of the fixed matrix iteration method. In this inexact iteration process, the classical LSQR method is implemented to determine an approximate solution of each least squares subproblem with less computational effort. The convergence of this algorithm is analyzed and several numerical examples are presented to illustrate the efficiency of the inexact fixed matrix iteration algorithm for solving large linear inequalities.

Keywords Linear inequalities · Inconsistent systems · Fixed matrix iteration · Inexact fixed matrix iteration · Krylov subspace · LSQR method

1 Introduction

Given real matrix $A \in \mathbb{R}^{m\times n}$ and real vector $b \in \mathbb{R}^m$, the need for finding a real vector $x \in \mathbb{R}^n$ that satisfies the system of linear inequalities

$$Ax \geq b \tag{1.1}$$

The work was supported by National Natural Science Foundations of China (No. 11201136) and Fundamental Research Funds for the Central Universities (No. 531107040014).

Y. Lei (✉)
College of Mathematics and Econometrics Hunan University, Changsha 410082, People's Republic of China
e-mail: yleimath@yahoo.com

arises in many applications, such as medical image reconstruction from projections [17, 18], inverse problems in radiation therapy [4, 5], calculating a separating hyperplane between two sets of points in $\mathbb{R}^n$ [2, 6], and so on. However, the system of linear inequalities (1.1) may be inconsistent, i.e., the feasible region $\{x \mid Ax \geq b\}$ may be empty because of data measurement errors in these applications. Hence, it is necessary to study the solutions of the system of linear inequalities (1.1) in a least squares sense.

There are two ways to deal with the inconsistent linear inequalities (1.1). The first, which is similar to the traditional least squares (LS) problem [1, 13], is to solve the LS problem

$$\min_{x\in\mathbb{R}^n} f(x) = \|(b - Ax)_+\|^2, \tag{1.2}$$

where the ith component of the vector $v_+ \in \mathbb{R}^m$ is $(v_+)_i = \max\{0, v_i\}$. Similarly, we denote $(v_-)_i = \min\{0, v_i\}$ as the ith component of the vector $v_- \in \mathbb{R}^m$, and then any $v \in \mathbb{R}^m$ can be split as $v = v_+ + v_-$. Here and later, $\|\cdot\|$ refers to the 2-norm of a vector induced from the inner product $\langle\cdot,\cdot\rangle$. In particularly, if there exists $x^* \in \mathbb{R}^n$ such that $f(x^*) = 0$, i.e., $(b-Ax^*)_+ = 0$, then we have $b-Ax^* = (b-Ax^*)_- \leq 0$, which means (1.1) is consistent.

Obviously, the function $f(x)$ is convex and continuously differentiable in $\mathbb{R}^n$, and it follows that x is a solution of the least squares problem (1.2) if and only if x satisfies the equation

$$A^T(b - Ax)_+ = 0. \tag{1.3}$$

Several Newton-type methods were presented in [3, 14, 22] to solve the minimization problem (1.2). The main effort in these Newton-type methods lies in computing the search direction u_k in every iteration step by solving the LS subproblem

$$\min \|A_k u - h_k\|^2, \tag{1.4}$$

where the matrix A_k and vector h_k consist of the rows of A and b associated with the active indices $I_k = \{i \mid a_i^T x^k < b_i\}$, respectively. The main advantage of Newton-type methods is their "finite termination" property, which occurs after the current approximate solution reaches the vicinity of a solution point. However, the main drawback of Newton-type methods is that the matrix A_k in (1.4) may become rank-deficient or ill-conditioned.

An equivalent way to deal with the inconsistent system (1.1) proposed by A. Dax [9] is to find a correction vector y with minimum norm such that the modified system $Ax \geq b - y$ is consistent. That is, to solve the Euclidean least deviation problem

$$\begin{aligned} \min\ & P(x, y) = \|y\|^2 \\ \text{s.t.}\ & Ax + y \geq b, \end{aligned} \tag{1.5}$$

which is equivalent to the quadratic programming problem

$$\begin{aligned} \min\ & F(x, z) = \|Ax - z - b\|^2 \\ \text{s.t.}\ & z \geq 0. \end{aligned} \tag{1.6}$$

See Dax [9, 10] for details. In [10], Dax presents a "fixed matrix (FM) iteration" algorithm to solve problems (1.5) and (1.6). The algorithm can be described as follows.

Algorithm FM (Dax [10, Section 3])
Given an arbitrary initial vector $x_1 \in \mathbb{R}^n$. Let

$$r_1 = b - Ax_1, \quad y_1 = (r_1)_+, \quad z_1 = (-r_1)_+.$$

For $k = 1, 2, \ldots$, the basic steps of the kth iteration are as follows.

Step 1: Compute a descent step u_k by solving the following LS subproblem

$$u_k = \arg\min_{u \in \mathbb{R}^n} \|Au - y_k\|^2. \tag{1.7}$$

Step 2: Update the solution estimate

$$x_{k+1} = x_k + u_k. \tag{1.8}$$

Step 3: Update the residual vector and its components

$$r_{k+1} = b - Ax_{k+1}, \quad y_{k+1} = (r_{k+1})_+, \quad z_{k+1} = (-r_{k+1})_+ \tag{1.9}$$

Step 4: Test for convergence according to the optimality condition (1.3).

Based on the optimality condition (1.3), Dax uses

$$\|A^T y_{k+1}\| \le \delta \|y_{k+1}\| \tag{1.10}$$

as the termination criterion, where δ is a preassigned small positive constant. In [10], Dax takes δ as

$$\delta = \alpha mn 10\varepsilon,$$

where ε denotes the roundoff unit in floating point arithmetic, and $\alpha = \max_{i,j}\{|a_{ij}|\}$. If $Ax \ge b$ is feasible, $\|y_{k+1}\|$ is tending to zero, so the algorithm is terminated as soon as it satisfies

$$\|y_{k+1}\| \le \delta.$$

As in the Newton-type methods, the main computational effort of Algorithm FM is solving the LS subproblem (1.7). But it is only necessary to update the vector y_k each iteration, and the fixed matrix A is independent of k, which is why Algorithm FM is named the "fixed matrix iteration" in [10].

The convergence of Algorithm FM has been proved, but the theory indicates that the convergence depends on accurate solution u_k of the LS subproblem (1.7) every iteration; see Theorem 1 in [10] for details. To avoid repetitive computations, the QR decomposition [24] of A, a direct method, is computed before starting the iteration process. Although the computation effort is rather small each iteration, the method is not suitable for large problems because of the high storage.

In order to reduce the amount of storage, on the one hand, we may adopt an explicit iterative method with no subproblem. Because

$$b - Ax = (b - Ax)_+ - (Ax - b)_+, \tag{1.11}$$

the optimality condition (1.3) can be rewritten as

$$A^T(Ax - z - b) = 0 \quad \text{and} \quad z = (Ax - b)_+. \tag{1.12}$$

For simplicity, we define

$$\mathbb{H} = \mathbb{R}^n \times \mathbb{R}^m_+ = \left\{(x, z) \mid x \in \mathbb{R}^n, z \in \mathbb{R}^m_+\right\} \tag{1.13}$$

as the feasible set, where $\mathbb{R}^m_+$ denotes the set consisting of all nonnegative vectors in $\mathbb{R}^m$, i.e., $\mathbb{R}^m_+ = \{z \in \mathbb{R}^m \mid z \geq 0\}$. Consequently an equivalent formulation of (1.12) is the linear projection equation

$$u = P_H[u - (Mu + q)], \tag{1.14}$$

where $P_H[v]$ denotes the projection of $v \in \mathbb{R}^{n+m}$ onto the feasible set $\mathbb{H}$, and

$$u = \begin{pmatrix} x \\ z \end{pmatrix}, \quad M = \begin{pmatrix} A^TA & -A^T \\ -A & I \end{pmatrix} = \begin{pmatrix} A^T \\ -I \end{pmatrix} \begin{pmatrix} A & -I \end{pmatrix}, \quad q = \begin{pmatrix} -A^Tb \\ b \end{pmatrix}.$$

If x^* solves (1.3), then x^* and $z^* = (Ax^* - b)_+$ solve (1.14). The projection and contraction (PC) method, which is presented in [15, 16] to solve the linear projection equation (1.14), can be modified to solve the LS problem (1.2). The resulting algorithm can be described as follows.

Algorithm PC

Step 1: Given an arbitrary initial vector pair $(x_1, z_1) \in \mathbb{H}$. For $k = 1, 2, \ldots$, compute

$$e_1(x_k, z_k) = A^T(Ax_k - z_k - b), \quad e_2(x_k, z_k) = z_k - (Ax_k - b)_+.$$

Step 2: Calculate the step size

$$\rho_k = \frac{\|e_1(x_k, z_k)\|^2 + \|e_2(x_k, z_k)\|^2}{\|e_1(x_k, z_k)\|^2 + \|e_2(x_k, z_k)\|^2 + \|Ae_1(x_k, z_k) - e_2(x_k, z_k)\|^2}.$$

Step 3: Update the solution estimate

$$x_{k+1} = x_k - \rho_k e_1(x_k, z_k), \quad z_{k+1} = z_k - \rho_k e_2(x_k, z_k).$$

Step 4: Test for convergence (as in Algorithm FM).

The sequence $\{z^k\}$ generated in Algorithm PC may not be contained in $\mathbb{R}^m_+$, but it asymptotically converges to the vector z^* as $k \to \infty$; see the proofs in [15, 16] for details. The main advantages of Algorithm PC are its simplicity and ability to handle large problems with any start point. However, the convergent rate of this algorithm may be very slow.

On the other hand, we may use iterative methods such as the classical LSQR method [1, 21] to solve large LS subproblem (1.7). It may be more efficient than direct methods, especially for problems with sparse structures. Although iterative methods possess the advantage of simplicity and small storage requirements, they may need more computational effort for solving the LS subproblem (1.7) accurately each iteration.

In this paper, based on the fixed matrix iteration method, we present an inexact fixed matrix iteration method, in which a constrained LS subproblem over a special

Krylov subspace [1, 23] of low dimension is considered at each iteration, and the corresponding solution is an approximate solution of (1.7). We adopt the LSQR method to solve the constrained LS subproblem with less computational effort, and expect to gain faster convergence.

The rest of the paper is organized as follows. In Section 2, we briefly review the LSQR method of solving the LS subproblem (1.7). In Section 3 we propose an inexact fixed matrix iteration algorithm based on Algorithm FM, and give the convergence analysis of this algorithm. We present several examples in Section 4 to show the effectiveness of the inexact fixed matrix iteration method for solving large problems, and give some concluding remarks in Section 5.

2 The LSQR method

We briefly discuss LSQR by Paige and Saunders [21] for solving the linear LS problem

$$\min_{u\in\mathbb{R}^n} \|Au - y\|^2, \tag{2.1}$$

where $A \in \mathbb{R}^{m\times n}$ and $y \in \mathbb{R}^m$. The LSQR algorithm uses the bidiagonalization procedure of Golub and Kahan [12] to reduce the matrix A to the lower bidiagonal form:

$$\left.\begin{aligned} \beta_1 q_1 = y, \quad \alpha_1 v_1 = A^T q_1 \\ \beta_{i+1} q_{i+1} = A v_i - \alpha_i q_i \\ \alpha_{i+1} v_{i+1} = A^T q_{i+1} - \beta_{i+1} v_i \end{aligned}\right\} \; i = 1, 2, \ldots. \tag{2.2}$$

The scalars $\alpha_i \geq 0$ and $\beta_i \geq 0$ are chosen so that $\|q_i\| = \|v_i\| = 1$. It is easy to show by an inductive proof that the vectors $\{q_i\}$ and $\{v_i\}$ generated in (2.2) are orthonormal, i.e.,

$$\langle q_i, q_j\rangle = 0, \quad \langle v_i, v_j\rangle = 0, \quad i \neq j.$$

Furthermore, the recurrence relations (2.2) can be rewritten in matrix form as

$$AV_s = Q_{s+1}\widetilde{L}_s, \quad A^T Q_s = V_s L_s^T, \tag{2.3}$$

where $V_s = [v_1, \cdots, v_s]$ and $Q_{s+1} = [q_1, \cdots, q_s, q_{s+1}]$, and $\widetilde{L}_s$ denotes the $(s+1)\times s$ lower bidiagonal matrix whose nonzero entries are defined by (2.2), and L_s is the $s \times s$ matrix obtained from $\widetilde{L}_s$ by deleting its last row:

$$\widetilde{L}_s = \begin{pmatrix} L_s \\ \beta_{s+1} e_s^{(s)T} \end{pmatrix} \quad \text{and} \quad L_s = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_s & \alpha_s \end{pmatrix},$$

where $e_s^{(s)}$ is the last column of the identity matrix I_s.

In exact arithmetic, the bidiagonalization (2.2) will terminate with $\alpha_{i+1} = 0$ or $\beta_{i+1} = 0$ for some $i \leq \min(m, n)$ because there are at most n orthogonal vectors

in $\mathbb{R}^n$. Suppose that (2.2) does not stop before the sth step ($s \leq \min(m,n)$), it is easy to verify that the vectors $v_1, v_2, \cdots, v_s$ are the orthonormal basis of the Krylov subspace

$$\mathcal{K}_s \equiv \mathcal{K}_s(A^TA, A^Ty) = \text{span}\{A^Ty, (A^TA)A^Ty, \cdots, (A^TA)^{s-1}A^Ty\}. \tag{2.4}$$

We now seek an approximate solution $u^{(s)} \in \mathcal{K}_s$ such that

$$u^{(s)} = \arg\min_{u\in\mathcal{K}_s} \|Au - y\|. \tag{2.5}$$

Since $\mathcal{K}_s = \text{span}\{v_1, v_2, \cdots, v_s\}$, we have $u = V_s w$ for any $u \in \mathcal{K}_s$, where $w \in \mathbb{R}^s$. It follows from (2.3) that

$$\|Au - y\| = \|AV_s w - y\| = \|Q_{s+1}(\widetilde{L}_s w - \beta_1 e_1)\| = \|\widetilde{L}_s w - \beta_1 e_1\|,$$

where e_1 is the first column of the identity matrix I_{s+1}. Therefore, (2.5) is equivalent to the unconstrained LS problem

$$w^{(s)} = \arg\min_{w\in\mathbb{R}^s} \|\widetilde{L}_s w - \beta_1 e_1\|. \tag{2.6}$$

LSQR solves (2.6) and constructs $u^{(s)} = V_s w^{(s)}$ iteratively by using the QR decomposition of the lower bidiagonal matrix $\widetilde{L}_s$ and simultaneously transforming $\beta_1 e_1$. The QR decomposition and the transformation can be easily implemented by Givens rotations; see [1, 21] for details. The main steps of the LSQR algorithm can be summarized as follows.

Algorithm LSQR

Initialize: $u^{(0)} = 0$;

$$\beta_1 = \|y\|, \quad q_1 = y/\beta_1, \quad v_1 = A^Tq_1, \quad \alpha_1 = \|v_1\|, \quad v_1 = v_1/\alpha_1.$$

Set

$$\tilde{\rho}_1 = \alpha_1, \quad \tilde{\zeta}_1 = \beta_1, \quad g_1 = v_1.$$

For $i = 1, 2, \ldots,$ the basic iteration is composed of the following steps.

Step 1: Compute q_{i+1}

$$q_{i+1} = Av_i - \alpha_i q_i, \quad \beta_{i+1} = \|q_{i+1}\|, \quad q_{i+1} = q_{i+1}/\beta_{i+1};$$

Step 2: Construct Givens rotation

$$\rho_i = \sqrt{\tilde{\rho}_i^2 + \beta_{i+1}^2}, \quad c_i = \tilde{\rho}_i/\rho_i, \quad s_i = \beta_{i+1}/\rho_i;$$

Step 3: Compute v_{i+1}

$$v_{i+1} = A^Tq_{i+1} - \beta_{i+1}v_i, \quad \alpha_{i+1} = \|v_{i+1}\|, \quad v_{i+1} = v_{i+1}/\alpha_{i+1};$$

Step 4: Update $\tilde{\rho}_{i+1}$ and $\tilde{\zeta}_{i+1}$

$$\theta_{i+1} = s_i\alpha_{i+1}, \quad \tilde{\rho}_{i+1} = c_i\alpha_{i+1}, \quad \zeta_i = c_i\tilde{\zeta}_i, \quad \tilde{\zeta}_{i+1} = -s_i\tilde{\zeta}_i;$$

Step 5: Compute $u^{(i)}$ and update g_{i+1}

$$u^{(i)} = u^{(i-1)} + \frac{\zeta_i}{\rho_i}g_i, \quad g_{i+1} = v_{i+1} - \frac{\theta_{i+1}}{\rho_i}g_i;$$

Step 6: Test for convergence.

LSQR is unusual in not having the residual vector $r^{(i)} = y - Au^{(i)}$ explicitly present, and its norm can be computed by

$$\|r^{(i)}\| = \|y - Au^{(i)}\| = |\tilde{\zeta}_{i+1}|. \tag{2.7}$$

For the inconsistent linear system $Au = y$, it is important to estimate the quantity $\|A^T r^{(i)}\|$, which would converge to zero if exact arithmetic were performed. Moreover, it can be easily determined by the product of three scalars

$$\|A^T r^{(i)}\| = \|A^T(y - Au^{(i)})\| = |c_i \alpha_{i+1} \tilde{\zeta}_{i+1}|, \tag{2.8}$$

instead of computing the residual directly. See [21] for details. Based on allowable perturbations in the data, Paige and Saunder suggested two stopping rules

$$\|r^{(i)}\| = |\tilde{\zeta}_{i+1}| \leq btol\|y\| + atol\|A\|\|u^{(i)}\| \tag{2.9}$$

and

$$\frac{\|A^T r^{(i)}\|}{\|A\|\|r^{(i)}\|} = \frac{|c_i \alpha_{i+1}|}{\|A\|} \leq atol \tag{2.10}$$

for the consistent and inconsistent systems respectively, where $\|A\|$ denotes the Frobenius norm of the matrix A, and the small tolerances $atol > 0$ and $btol > 0$ can be set according to the accuracy of the data.

Remark 1 Mathematically, LSQR generates the same sequence of approximate solutions $\{u^{(i)}\}$ as CGLS algorithm [1, 25]. However, LSQR is shown in [21] to be numerically more reliable when A is ill-conditioned.

Remark 2 From the discussion above, we know that the solution $u^{(s)}$ of (2.5) is an approximate solution of (2.1), and can be obtained after performing s iterations of LSQR. The following conclusion indicates that $u^{(s)}$ is an accurate solution of (2.1) if s is large enough.

Theorem 1 *Let μ be the number of the distinct eigenvalues of $A^T A$. Then the solution $u^{(s)}$ of (2.5) is also the solution of the LS subproblem (2.1) if $s \geq \mu$.*

Proof Since the Krylov subspace $\mathcal{K}_s(A^T A, A^T y)$ is a subset of $\mathbb{R}^n$, we only need to show that there exists a least squares solution of (2.1) contained in the Krylov subspace when $s \geq \mu$.

It is well know that u is a solution of the LS problem (2.1) if and only if u satisfies the normal equation

$$A^T Au = A^T y. \tag{2.11}$$

We list all distinct eigenvalues of $A^T A$ and further order them such that

$$\lambda_1 > \lambda_2 > \cdots > \lambda_\mu \geq 0.$$

Then

$$p(t) = (t - \lambda_1)(t - \lambda_2)\cdots(t - \lambda_\mu) = t^\mu + a_{\mu-1}t^{\mu-1} + \cdots + a_1 t + a_0$$

is the monic polynomial of A^TA with degree μ, and

$$p(A^TA)(A^Ty) = (A^TA)^{\mu}(A^Ty) + a_{\mu-1}(A^TA)^{\mu-1}(A^Ty) + \cdots + a_1(A^TA)(A^Ty) + a_0(A^Ty) = 0. \quad (2.12)$$

If A has full rank, then $\lambda_\mu > 0$, which implies that $a_0 \neq 0$. Denote

$$u^{(s)} = -\frac{1}{a_0}\{(A^TA)^{\mu-1}(A^Ty) + a_{\mu-1}(A^TA)^{\mu-2}(A^Ty) + \cdots + a_1(A^Ty)\}.$$

Then $u^{(s)} \in \mathcal{K}_s(A^TA, A^Ty)$ because of $s \geq \mu$, and in this case, $u^{(s)}$ is the unique least squares solution of (2.1) because $u^{(s)}$ satisfies (2.11).

If A is rank-deficient, then $\lambda_\mu = 0$, which implies that $a_0 = 0$ and $a_1 \neq 0$. Denote

$$v^{(s)} = (A^TA)^{\mu-1}(A^Ty) + a_{\mu-1}(A^TA)^{\mu-2}(A^Ty) + \cdots + a_1(A^Ty). \quad (2.13)$$

Then we have $(A^TA)v^{(s)} = 0$, which means $Av^{(s)} = 0$, i.e., $v^{(s)} \in \text{Null}(A)$. On the other hand, it follows from (2.13) that $v^{(s)} \in \text{Range}(A^T) = \text{Null}(A)^{\perp}$, so we obtain $v^{(s)} = 0$. Denote

$$u^{(s)} = -\frac{1}{a_1}\{(A^TA)^{\mu-2}(A^Ty) + a_{\mu-1}(A^TA)^{\mu-3}(A^Ty) + \cdots + a_2(A^Ty)\}.$$

Then $u^{(s)} \in \mathcal{K}_s(A^TA, A^Ty)$, and $u^{(s)}$ also satisfies (2.11), which means $u^{(s)}$ is a solution of LS problem (2.1). □

Remark 3 From Theorem 1, we know that LSQR converges to the accurate solution of (2.1) after at most μ ($\mu \leq n$) iterations with exact arithmetic. In practice, the number of iterations needed could be fewer than μ or many more than μ depending on the right-hand side y because of loss of orthogonality in $\{q_i\}$ and $\{v_i\}$.

3 The inexact fixed matrix (IFM) iteration

In this section, we present an inexact algorithm for solving problem (1.2).

Algorithm IFM

Given a nonnegative integer s and an arbitrary initial vector $x_1 \in \mathbb{R}^n$. Let

$$r_1 = b - Ax_1, \quad y_1 = (r_1)_+, \quad z_1 = (-r_1)_+.$$

For $k = 1, 2, \ldots,$ the basic steps of the kth iteration are as follows.

Step 1: Compute a descent step u_k by solving the constrained LS subproblem

$$u_k = \arg \min_{u \in \mathcal{K}_s(A^TA, A^Ty_k)} \|Au - y_k\|, \quad (3.1)$$

where

$$\mathcal{K}_s(A^TA, A^Ty_k) = \text{span}\{A^Ty_k, (A^TA)A^Ty_k, \cdots, (A^TA)^{s-1}A^Ty_k\}. \quad (3.2)$$

is a Krylov subspace with the dimension not exceeding s.

Step 2: Update the solution estimate

$$x_{k+1} = x_k + u_k. \tag{3.3}$$

Step 3: Update the residual vector and its components

$$r_{k+1} = b - Ax_{k+1}, \quad y_{k+1} = (r_{k+1})_+, \quad z_{k+1} = (-r_{k+1})_+. \tag{3.4}$$

Step 4: Test for convergence according to the optimality condition (1.3).

Remark 4 The solution u_k in (3.1) is an approximate solution of (1.7). Theorem 1 indicates that u_k is the least squares solution over $\mathbb{R}^n$ if s is large enough, in which case, Algorithm IFM reduces to Algorithm FM, i.e., the "fixed matrix iteration". Hence, we refer to Algorithm IFM as the inexact fixed matrix iteration.

In the following, we show the convergence property of Algorithm IFM. Before that, let us briefly recall the classical projection theorem on closed convex set and subspace [26, 29].

Lemma 1 *Let $\mathbb{V}$ be an inner product space and $\mathbb{M}$ be a closed convex set of $\mathbb{V}$. For a given $v \in \mathbb{V}$, there exists a unique projection vector $m_0 \in \mathbb{M}$ such that $\|v - m_0\| \leq \|v - m\|$ for any $m \in \mathbb{M}$, and m_0 is a projection of v onto $\mathbb{M}$ if and only if m_0 satisfies*

$$\langle v - m_0, m - m_0 \rangle \leq 0, \quad \forall m \in \mathbb{M}. \tag{3.5}$$

Furthermore, if $\mathbb{M}$ is a subspace of $\mathbb{V}$, then $m_0 \in \mathbb{M}$ is the unique projection vector of v onto $\mathbb{M}$ if and only if m_0 satisfies

$$\langle v - m_0, m \rangle = 0, \quad \forall m \in \mathbb{M}. \tag{3.6}$$

It is clear that the solution u_k of the constrained LS subproblem (3.1) satisfies

$$\|Au_k - y_k\| \leq \|y_k\|, \tag{3.7}$$

and it follows from Lemma 1 that $\langle Au_k - y_k, Au_k \rangle = 0$, which is equivalent to

$$\|Au_k\|^2 = \langle y_k, Au_k \rangle = \langle b + z_k - Ax_k, Au_k \rangle. \tag{3.8}$$

Moreover, equality holds in (3.7) if and only if $A^T y_k = 0$.

Theorem 2 *For any given positive integer s, assuming (3.1) is solved exactly, then the sequence $\{x_k\}$ generated by Algorithm IFM satisfies*

$$\lim_{k\to\infty} A^T (b - Ax_k)_+ = \lim_{k\to\infty} A^T y_k = 0.$$

Proof For the sequence $\{y_{k+1}\}$ in (3.4), it follows from (3.7) and (3.8) that

$$\begin{aligned} \|y_{k+1}\|^2 &= \|(b - Ax_{k+1})_+\|^2 \\ &= \|(b + z_k - z_k - Ax_{k+1})_+\|^2 \\ &\le \|(b + z_k - Ax_{k+1})_+\|^2 \\ &\le \|b + z_k - Ax_{k+1}\|^2 \\ &= \|y_k - Au_k\|^2 \\ &= \|y_k\|^2 - 2\langle y_k, Au_k\rangle + \|Au_k\|^2 \\ &= \|y_k\|^2 - \|Au_k\|^2 \\ &\le \|y_k\|^2, \end{aligned} \tag{3.9}$$

which implies that the sequence $\{\|y_k\|\}$ is monotonically decreasing and is bounded below. Consequently, we have

$$\lim_{k\to\infty} (\|y_k\|^2 - \|y_{k+1}\|^2) = 0 \quad \text{and} \quad \lim_{k\to\infty} Au_k = 0. \tag{3.10}$$

Since u_k is the solution of constrained LS subproblem (3.1) over the Krylov subspace (2.4), we have from Lemma 1 that

$$\langle y_k - Au_k, Au\rangle = 0, \quad \forall u \in \mathcal{K}_s(A^TA, A^Ty_k). \tag{3.11}$$

By taking $u = A^Ty_k$ in (3.11), we have

$$\langle y_k - Au_k, AA^Ty_k\rangle = \langle A^T(y_k - Au_k), A^Ty_k\rangle = \|A^Ty_k\|^2 - \langle A^TAu_k, A^Ty_k\rangle = 0, \tag{3.12}$$

which yields

$$\|A^Ty_k\|^2 = \langle A^TAu_k, A^Ty_k\rangle \ge 0$$

and

$$\langle A^TAu_k, A^Ty_k\rangle \le \|A^TAu_k\|\|A^Ty_k\| \le \|A\|_2^2\|Au_k\|\|y_k\| \le \|A\|_2^2\|Au_k\|\|y_1\| \to 0$$

when $k \to \infty$, where $\|A\|_2$ denotes the spectral norm of the matrix A. Hence,

$$\lim_{k\to\infty} A^Ty_k = \lim_{k\to\infty} A^T(b + z_k - Ax_k) = \lim_{k\to\infty} A^T(b - Ax_k)_+ = 0.$$

□

Since the sequence $\{y_k\}$ is bounded, there exists a subsequence $\{y_{k_j}\}$ such that

$$\lim_{j\to\infty} y_{k_j} = y^*.$$

It is clear that the cluster point $y^* \in \mathbb{R}^m_+$, and from Theorem 2 we have

$$\lim_{k\to\infty} A^Ty_k = \lim_{j\to\infty} A^Ty_{k_j} = A^Ty^* = 0.$$

That is, the cluster point y^* of the sequence $\{y_k\}$ satisfies $y^* \in \mathbb{Y}$. As shown in [10], the cluster point y^* is unique by means of the uniqueness of "polar decomposition" theorem [19, 27]. Hence, we can obtain the same convergence result that is established in [10].

Corollary 1 *Let $b = w^* + y^*$ be the unique polar decomposition of b, where $w^* \in \mathbb{W}$ and $y^* \in \mathbb{W}^*$. Then*

$$\lim_{k\to\infty} y_k = y^* \quad \text{and} \quad \lim_{k\to\infty} (Ax_k - z_k) = w^*.$$

4 Numerical experiments

In this section, we present the results of some numerical experiments aimed at examining the performance of the inexact fixed matrix iterative method (Algorithm IFM), and also compare Algorithm IFM with the exact fixed matrix iterative method (Algorithm FM) as well as with the projection and contraction method (Algorithm PC) presented in [15, 16], in terms of the number of iteration steps (denoted by "IT") and the total elapsed CPU time in seconds (denoted by "CPU"). All the tests are performed using MATLAB 7.10, which has a machine precision of around 10^{-16}.

In all cases, we used $x_1 = 0 \in \mathbb{R}^n$ as the initial guess, and the stopping criterion for each algorithm is

$$\frac{\|A^T(b - Ax_k)_+\|}{\|A\|\|(b - Ax_k)_+\|} = \frac{\|A^T y_k\|}{\|A\|\|y_k\|} \le 10^{-12}. \tag{4.1}$$

If (1.1) is feasible, $\|y_k\|$ is tending to zero, so we accept x_k as the feasible solution if it satisfies

$$\|(b - Ax_k)_+\| = \|y_k\| \le 10^{-12}(\|A\|\|x_k\| + \|b\|). \tag{4.2}$$

For Algorithm FM, we use the direct method (Sparse QR) [11] and the iterative method (LSQR) [21] to solve the large sparse LS subproblem (1.7), respectively. Furthermore, a delicate issue concerning the stopping criterion for the inner iterative solver of (1.7) should be considered. For the infeasible system (1.1), it is readily seen from (3.9) that the linear systems $Au = y_k$ is inconsistent for all y_k generated exactly by Algorithms FM and IFM, but the result may be invalid for the feasible case. For this reason, we accept the approximate solution u_k of (1.7) if the corresponding residual satisfies

$$\frac{\|A^T(y_k - Au_k)\|}{\|A\|\|y_k - Au_k\|} \le \epsilon \quad \text{or} \quad \frac{\|y_k - Au_k\|}{\|A\|} \le \epsilon \tag{4.3}$$

for a small tolerance $\epsilon > 0$. For convenience, we test the performance of Algorithm FM with an invariable tolerance $\epsilon = 10^{-12}$ in our experiments.

For Algorithm IFM, we also use the LSQR algorithm to solve the LS subproblem (3.1), where s is a preassigned small positive integer. As shown in Section 2, the orthonormal basis of the Krylov subspace (3.2) can be obtained after s iteration steps at most by using the bidiagonalization procedure (2.2), and the solution u_k of (3.1) will be determined by the LSQR algorithm within s iterations in exact arithmetic. In the practical implementation of the inner iteration, the solution u_k which satisfies

(4.3) could be obtained by using less than s iterations once y_k becomes sufficiently close to y^*. In other words, the LS subproblem (3.1) has reduced to (1.7) in this case. To save the amount of computation, in our algorithm we adopt the following alternative inner stopping strategy

$$\textbf{if} \quad u_k \text{ satisfies(4.3)} \quad or \quad \mathrm{IT}_{inner} > s \quad \textbf{then} \quad stop, \tag{4.4}$$

where IT_{inner} denotes the number of inner iterations.

Example 1

We first test the performance of Algorithms PC, FM and IFM by solving the consistent systems of linear inequalities (1.1), where the test matrices WELL1033, ILLC1033, WELL1850, ILLC1850 come from the Matrix Market repository [30], and the vector $b \in \mathbb{R}^m$ is given by

$$b = (1, 1, \cdots, 1)^T \quad \text{and} \quad b = (-1, 1, \cdots, (-1)^i, \cdots, (-1)^m)^T. \tag{4.5}$$

For Algorithm IFM, we take $s = 1, 5, 10, 20$ and use different inner tolerance values $\epsilon = 10^{-6}, 10^{-9}, 10^{-12}$ to test the behaviour of this algorithm. The numerical results are listed in Tables 1, 2, 3, 4.

The second columns of Tables 1, 2, 3, 4 give the least residual norms of the systems of linear equations $Ax = b$ with these four matrices, and the results show that these systems are consistent for the first vector b in (4.5) and inconsistent for the second one. On the other hand, the last two columns of Tables 1, 2, 3, 4 indicate that these four systems of linear inequalities for each vector b in (4.5) are consistent and can be solved effectively by the proposed algorithms. For Algorithms FM and IFM, the fourth columns of Tables 1, 2, 3, 4 give the number of outer iterations (outside the brackets), and the average number of inner iterations (inside the brackets) required for solving the LS subproblems (1.7) and (3.1) by LSQR. For Algorithm FM, the figure inside the brackets denotes the CPU time for the sparse QR decomposition, and the total running time is given outside the brackets in the fifth columns of Tables 1, 2, 3, 4.

For the consistent systems of linear equations $Ax = b$, the results in Tables 1 and 2 indicate that the superiority of Algorithm FM is pronounced compared to Algorithms PC and IFM, and the number of outer iterations of Algorithm FM is far less. The reason is that the solution of $Ax = b$ is one of the solutions of the system of linear inequalities (1.1), and the solution can be obtained after one iteration step in the absence of roundoff errors. However, the least squares solution of the inconsistent system $Ax = b$ may not be the solution of (1.1). Hence, more CPU time for Algorithm FM is consumed to implement the QR factorization and solve the least squares subproblem (1.7) accurately. This is confirmed by the results in Table 3 and 4.

Similar to the LSQR method, the sparse QR factorization can also be used to solve large sparse LS supproblem (1.7), especially for the ill-conditioned problems (ILLC1033 and ILLC1850). As mentioned in [10], this factorization is done only once before starting the iterative process, so the computational effort per iteration is rather small. However, the required numbers of outer iterations by using these

Table 1 Results for well-conditioned problems when $b = (1, 1, \cdots, 1)^T$

A $(m \times n)$	$\min \|Ax - b\|$	Algorithm			IT	CPU	$\|y_k\|$	$\|A^T y_k\|$
		PC			4489	0.79	4.9e-10	4.6e-11
		FM	QR		1	0.21 (0.19)	1.5e-14	1.7e-14
			LSQR		2(100)	0.03	5.5e-13	4.6e-13
			$s = 1$		1416(1)	0.27	5.0e-10	5.7e-11
WELL1033				ϵ = 1e-6	316(3)	0.14	4.7e-10	6.0e-11
			$s = 5$	ϵ = 1e-9	253(4)	0.14	4.7e-10	6.0e-11
	1.3e-8			ϵ = 1e-12	217(5)	0.14	4.7e-10	1.2e-10
				ϵ = 1e-6	326(3)	0.17	4.9e-10	1.1e-10
(1033 × 320)		IFM	$s = 10$	ϵ = 1e-9	325(7)	0.27	5.0e-10	1.1e-10
				ϵ = 1e-12	324(10)	0.34	5.0e-10	1.1e-10
				ϵ = 1e-6	341(6)	0.24	5.3e-10	1.2e-10
			$s = 20$	ϵ = 1e-9	334(15)	0.47	5.4e-10	1.2e-10
				ϵ = 1e-12	331(20)	0.62	5.3e-10	1.2e-10
		PC			4701	1.81	9.2e-10	1.1e-10
		FM	QR		1	0.21 (0.19)	1.5e-14	1.7e-14
	2.7e-8							

Table 1 (continued)

A ($m \times n$)	$\min \|Ax - b\|$	Algorithm			IT	CPU	$\|y_k\|$	$\|A^T y_k\|$
			LSQR		2(246)	0.11	1.7e-11	1.6e-11
			$s = 1$		1369(1)	0.60	9.3e-10	1.1e-10
WELL1850				ϵ = 1e-6	496(2)	0.36	9.2e-10	1.2e-10
			$s = 5$	ϵ = 1e-9	289(3)	0.25	9.1e-10	1.2e-10
				ϵ = 1e-12	218(5)	0.25	8.9e-10	2.1e-10
				ϵ = 1e-6	353(3)	0.29	9.2e-10	1.5e-10
(1850 × 712)		IFM	$s = 10$	ϵ = 1e-9	334(8)	0.45	9.0e-10	1.9e-10
				ϵ = 1e-12	331(10)	0.58	9.1e-10	1.9e-10
				ϵ = 1e-6	320(4)	0.33	9.5e-10	2.1e-10
			$s = 20$	ϵ = 1e-9	314(15)	0.71	9.6e-10	2.1e-10
				ϵ = 1e-12	311(20)	0.92	9.7e-10	2.1e-10

Table 2 Results for ill-conditioned problems when $b = (1, 1, \cdots, 1)^T$

A ($m \times n$)	$\min \|Ax - b\|$	Algorithm			IT	CPU	$\|y_k\|$	$\|A^T y_k\|$
ILLC1033 (1033 × 320)	1.7e-7	PC			9753	1.70	3.5e-10	4.4e-11
		FM	QR		1	0.21 (0.19)	1.5e-14	1.7e-14
			LSQR		2(1840)	0.33	1.2e-11	1.5e-11
		IFM	$s = 1$		3543(1)	0.92	3.7e-10	4.5e-11
			$s = 5$	ϵ = 1e-6	1143(2)	0.46	3.7e-10	4.8e-11
				ϵ = 1e-9	695(3)	0.33	3.7e-10	4.8e-11
				ϵ = 1e-12	445(5)	0.29	3.6e-10	7.0e-11
			$s = 10$	ϵ = 1e-6	458(3)	0.23	4.0e-10	8.7e-11
				ϵ = 1e-9	359(7)	0.32	4.0e-10	1.0e-10
				ϵ = 1e-12	351(10)	0.36	4.0e-10	1.1e-10
			$s = 20$	ϵ = 1e-6	414(5)	0.26	4.4e-10	9.5e-11
				ϵ = 1e-9	304(14)	0.39	4.2e-10	1.2e-10
				ϵ = 1e-12	301(20)	0.57	4.3e-10	1.2e-10

Table 2 (continued)

A ($m \times n$)	$\min \|Ax - b\|$	Algorithm			IT	CPU	$\|y_k\|$	$\|A^T y_k\|$
		PC			9342	2.83	7.3e-10	9.0e-11
		FM	QR		1	0.21 (0.19)	1.5e-14	1.7e-14
			LSQR		2(1191)	0.34	1.8e-11	2.1e-11
			$s = 1$		3422(1)	1.48	7.6e-10	9.3e-11
ILLC1850				ϵ = 1e-6	1120(2)	0.75	7.6e-10	1.0e-10
			$s = 5$	ϵ = 1e-9	726(3)	0.60	7.6e-10	1.0e-10
	3.9e-8			ϵ = 1e-12	522(5)	0.54	7.5e-10	1.4e-10
				ϵ = 1e-6	462(4)	0.39	8.2e-10	1.9e-10
(1850 × 712)		IFM	$s = 10$	ϵ = 1e-9	320(7)	0.40	8.2e-10	1.9e-10
				ϵ = 1e-12	250(10)	0.42	8.2e-10	1.9e-10
				ϵ = 1e-6	417(5)	0.42	8.7e-10	1.9e-10
			$s = 20$	ϵ = 1e-9	240(12)	0.48	8.9e-10	2.0e-10
				ϵ = 1e-12	182(20)	0.54	8.8e-10	2.7e-10

two methods are almost identical. Recently, Tim Davis proposed a new sparse QR factorization: SuiteSparseQR in [7] and the available software package [31]. It stores R as an explicit sparse matrix and can store Q compactly as a product of sparse Householder transformations. This factorization is sometimes extremely efficient for solving large sparse LS problem. But it is unhelpful for reducing the number of outer iterations.

Compared to Algorithms PC and FM, Algorithm IFM is quite efficient in this example, even for the ill-conditioned problems. For different choices of s, Algorithm IFM has different numerical behaviors. When $s = 1$, the Algorithm IFM is equivalent to an explicit method, so it requires more outer iterations and more CPU time to update the residual vector and its components in (3.4). When $s > 1$, the Algorithm IFM shows faster convergence because it needs the fewer inner iterations.

This test also shows that the performance of Algorithm IFM is influenced by the inner tolerance ϵ while the current estimate x_k reaches the vicinity of the solution point. By the inner stopping strategy (4.4), the higher precision the less numbers of outer iterations are required. On the other hand, the performance of the inexact method should be evaluated by taking into account the overall cost and not only the number of outer iterations. When close to convergence, the inner iterations may terminate within s iterations by choosing a small tolerance value. As a consequence, the average numbers of inner iterations are less than s for $s > 1$ and the total computational cost may decrease.

Moreover, it is worth noting that the required outer iterations of Algorithm FM in Tables 3 and 4 may be a little more than for Algorithm IFM in some instances, even though Algorithm FM is solving the related linear LS problem accurately. The reason is that the accurate solution u_k of (1.7) may not be the optimal descent direction for these instances.

Example 2

This example is used to test the numerical performance of Algorithm IFM by solving several inconsistent systems of linear inequalities (1.1), which are constructed artificially based on Example 1. Let $N = \{1, 2, \ldots, m\}$ be an index set and I be a subset of N. For given matrix $A \in \mathbb{R}^{m\times n}$, $A(i,:)$ denotes the i^{th} row of A, and we define a new matrix $A_I \in \mathbb{R}^{m\times n}$, whose i^{th} row satisfies

$$A_I(i,:) = \begin{cases} A(i,:) & \text{if } i \notin I, \\ 0 & \text{if } i \in I. \end{cases}$$

For arbitrary $b \in \mathbb{R}^m$, if its i^{th} element b_i satisfies $b_i > 0$ for $i \in I$, then it is obvious that the system of linear inequalities $A_I x \geq b$ is inconsistent.

In this example, we construct four test matrices A_I based on the matrices given in Example 1, where $I = \{20j,\ j = 1, 2, \ldots, 50\}$, and take b as the second vector in (4.5). For convenience, we only use the inner tolerance value $\epsilon = 10^{-9}$ to test the

Table 3 Results for well-conditioned problems when $b = (-1, \cdots, (-1)^i, \cdots, (-1)^m)^T$

A ($m \times n$)	$\min \|Ax - b\|$	Algorithm			IT	CPU	$\|y_k\|$	$\|A^T y_k\|$
		PC			3300	0.54	4.8e-10	6.8e-11
		FM	QR		370	3.06 (0.18)	1.5e-9	3.3e-10
			LSQR		371(161)	5.12	1.5e-9	3.4e-10
			$s = 1$		964(1)	0.26	4.7e-10	6.5e-11
WELL1033				ϵ = 1e-6	427(3)	0.21	4.8e-10	7.7e-11
			$s = 5$	ϵ = 1e-9	380(4)	0.22	4.7e-10	1.0e-10
	27.84			ϵ = 1e-12	378(5)	0.23	4.6e-10	1.1e-10
				ϵ = 1e-6	361(4)	0.21	5.4e-10	8.6e-11
(1033 × 320)		IFM	$s = 10$	ϵ = 1e-9	366(8)	0.31	5.3e-10	1.2e-10
				ϵ = 1e-12	364(10)	0.39	5.4e-10	1.2e-10
				ϵ = 1e-6	401(8)	0.36	7.8e-10	1.8e-10
			$s = 20$	ϵ = 1e-9	393(16)	0.59	7.6e-10	1.7e-10
				ϵ = 1e-12	389(20)	0.74	7.6e-10	1.8e-10

Table 3 (continued)

A ($m \times n$)	$\min \|Ax - b\|$	Algorithm			IT	CPU	$\|y_k\|$	$\|A^T y_k\|$
		PC			4738	1.33	9.8e-10	1.1e-10
		FM	QR		279	6.2 (0.40)	2.1e-9	5.6e-10
			LSQR		280(430)	16.81	2.1e-9	5.6e-10
			$s = 1$		1381(1)	0.61	9.1e-10	1.0e-10
WELL1850				ϵ = 1e-6	480(2)	0.32	1.1e-9	1.3e-10
			$s = 5$	ϵ = 1e-9	308(4)	0.28	1.1e-9	1.4e-10
	32.82			ϵ = 1e-12	275(5)	0.29	1.1e-9	2.8e-10
				ϵ = 1e-6	369(4)	0.34	1.2e-9	2.7e-10
(1850 × 712)		IFM	$s = 10$	ϵ = 1e-9	364(8)	0.54	1.2e-9	2.8e-10
				ϵ = 1e-12	362(10)	0.66	1.2e-9	2.7e-10
				ϵ = 1e-6	278(10)	0.40	1.5e-9	4.0e-10
			$s = 20$	ϵ = 1e-9	280(16)	0.69	1.5e-9	4.0e-10
				ϵ = 1e-12	279(20)	0.89	1.5e-9	4.0e-10

Table 4 Results for ill-conditioned problems when $b = (-1, \cdots, (-1)^i, \cdots, (-1)^m)^T$

A $(m \times n)$	$\min \|Ax - b\|$	Algorithm			IT	CPU	$\|y_k\|$	$\|A^T y_k\|$
		PC			6154	1.59	3.6e-10	5.3e-11
		FM	QR		328	2.67 (0.18)	8.9e-9	2.6e-10
			LSQR		329(3240)	85.55	8.9e-9	2.6e-9
			$s = 1$		1980(1)	0.53	3.5e-10	5.0e-11
ILLC1033				ϵ = 1e-6	969(2)	0.39	3.6e-10	5.9e-11
			$s = 5$	ϵ = 1e-9	674(3)	0.34	3.6e-10	5.9e-11
	27.84			ϵ = 1e-12	510(5)	0.31	3.7e-10	8.2e-11
				ϵ = 1e-6	409(4)	0.23	4.5e-10	1.2e-10
(1033 × 320)		IFM	$s = 10$	ϵ = 1e-9	386(8)	0.33	4.5e-10	1.2e-10
				ϵ = 1e-12	381(10)	0.39	4.5e-10	1.2e-10
				ϵ = 1e-6	394(6)	0.29	5.8e-10	1.4e-10
			$s = 20$	ϵ = 1e-9	275(14)	0.37	5.8e-10	1.4e-10
				ϵ = 1e-12	224(20)	0.41	5.8e-10	2.2e-10

Table 4 (continued)

A $(m \times n)$	$\min \|Ax - b\|$	Algorithm			IT	CPU	$\|y_k\|$	$\|A^T y_k\|$
		PC			9269	2.66	7.9e-10	9.8e-11
		FM	QR		280	5.7 (0.40)	2.0e-9	6.9e-10
			LSQR		281(430)	76.01	2.0e-9	6.9e-10
			$s = 1$		3474(1)	1.48	7.1e-10	8.7e-11
ILLC1850				ϵ = 1e-6	1121(2)	0.74	9.3e-10	1.2e-10
			$s = 5$	ϵ = 1e-9	715(3)	0.59	9.4e-10	1.2e-10
	32.82			ϵ = 1e-12	520(5)	0.51	9.2e-10	1.7e-10
				ϵ = 1e-6	496(5)	0.34	1.2e-9	2.9e-10
(1850 × 712)		IFM	$s = 10$	ϵ = 1e-9	355(7)	0.47	1.2e-9	2.8e-10
				ϵ = 1e-12	293(10)	0.50	1.2e-9	3.2e-10
				ϵ = 1e-6	396(8)	0.46	1.6e-9	3.6e-10
			$s = 20$	ϵ = 1e-9	259(16)	0.67	1.7e-9	4.5e-10
				ϵ = 1e-12	251(20)	0.75	1.7e-9	5.2e-10

Table 5 Results for solving inconsistent systems of linear inequalities

A_I ($m \times n$)	$\min \|Ax-b\|$	Algorithm		IT	CPU	$\|y_k\|$	$\|A^T y_k\|$
WELL1033$_I$ (1033 × 320)	27.82	PC		2974	0.53	7.07	1.2e-10
		FM	QR	377	3.31(0.38)		1.2e-10
			LSQR	386(112)	3.60		1.2e-10
		IFM	$s=1$	903(1)	0.26		1.2e-10
			$s=5$	396(3)	0.19		1.2e-10
			$s=10$	342(5)	0.23		1.2e-10
			$s=20$	374(8)	0.33		1.2e-10
ILLC1033$_I$ (1033 × 320)	27.82	PC		5494	0.67	7.07	1.2e-10
		FM	QR	383	3.45(0.35)		1.2e-10
			LSQR	409(1293)	42.82		1.2e-10
		IFM	$s=1$	1779(1)	0.50		1.2e-10
			$s=5$	812(3)	0.41		1.2e-10
			$s=10$	380(6)	0.23		1.2e-10
			$s=20$	361(8)	0.28		1.2e-10
WELL1850$_I$ (1850 × 712)	33.00	PC		4375	1.35	7.07	1.9e-10
		FM	QR	293	7.57(1.68)		1.8e-10
			LSQR	300(165)	6.77		1.8e-10
		IFM	$s=1$	1334(1)	0.65		1.8e-10
			$s=5$	412(3)	0.31		1.8e-10
			$s=10$	384(6)	0.37		1.8e-10
			$s=20$	356(8)	0.44		1.8e-10
ILLC1850$_I$ (1850 × 712)	33.00	PC		8834	2.78	7.07	1.9e-10
		FM	QR	298	7.59(1.65)		1.8e-10
			LSQR	309(770)	32.25		1.8e-10
		IFM	$s=1$	3323(1)	1.56		1.9e-10
			$s=5$	918(3)	0.67		1.9e-10
			$s=10$	463(5)	0.39		1.8e-10
			$s=20$	395(9)	0.50		1.8e-10

performance of Algorithm IFM. The numerical results are listed in Table 5, and the symbols are the same as that in Tables 1,2,3,4.

Example 3

In this example, we compare the numerical behaviour of these algorithms for solving large and dense linear inequality systems (1.1), where the entries of A and b generated by Matlab are random numbers from the uniform distribution on the interval $[-1, 1]$. Therefore, the test problems are not necessarily consistent. For simplicity, we only use LSQR method to solve (1.7) in Algorithm FM, and take $s = 5$

Table 6 CPU times and iteration counts of Algorithms when solving random test problems

m	Algorithm	The value of n				
		$n = 0.2m$	$n = 0.4m$	$n = 0.6m$	$n = 0.8m$	$n = m$
$m = 500$	PC	6.62(†)	105.18(†)	132.98(†)	18.59(†)	7.02(†)
	FM	0.29(197)	4.18(1383)	21.95(1417)	5.68(136)	0.29(2)
	IFM	0.11(194)	1.19(1356)	1.92(1589)	0.26(165)	0.11(54)
$m = 1000$	PC	41.17(†)	767.36(†)	*	222.99(†)	119.09(†)
	FM	0.79(194)	18.96(1042)	85.09(731)	34.22(104)	5.14(2)
	IFM	0.32(190)	5.29(1087)	6.65(761)	1.63(116)	0.80(53)
$m = 2000$	PC	595.40(†)	*	*	*	*
	FM	6.26(192)	154.74(1790)	569.06(1154)	140.49(93)	38.34(2)
	IFM	2.53(189)	45.13(1782)	44.34(1219)	5.93(125)	2.58(45)
$m = 3000$	PC	1749.58(†)	*	*	*	*
	FM	11.68(174)	269.88(1356)	1249.27(1134)	309.96(94)	131.73(2)
	IFM	4.69(171)	72.53(1333)	94.06(1184)	12.69(125)	6.50(50)
$m = 4000$	PC	*	*	*	*	*
	FM	22.71(189)	588.05(1787)	1900.55(1026)	541.83(95)	299.59(2)
	IFM	9.02(186)	167.63(1770)	147.85(1076)	21.60(122)	11.24(52)

and $\epsilon = 10^{-9}$ for Algorithm IFM. The number of variables is defined with $n = \sigma * m$ in all test problems, where m is the number of inequalities, and $\sigma = 0.2, 0.4, 0.8, 1$. Each cell in Table 6 gives the CPU time (outside the bracket) and the number of iterations (inside the bracket) taken by Algorithms PC, FM and IFM for solving the problems with various sizes. For Algorithm PC, the symbol $*$ denotes that it does not converge within 2000 seconds and † in brackets indicates that the required number of iterations exceeds 10000 steps.

Although the amount of computation per iteration for Algorithm PC is less than for Algorithms FM and IFM, the results in Table 6 show that Algorithm PC is not suitable for solving large dense linear inequality systems. The main reason of its slow convergence is that the iteration sequence $\{(x^k, z^k)\}$ generated by Algorithm PC may not be contained in the feasible set $\mathbb{H}$ in (1.13), so it is called an infeasible PC method in [28]. As shown in Tables 3,4,5, almost identical numbers of iteration are required by Algorithms FM and IFM in most cases, but less CPU time is consumed by Algorithm IFM because of its lower computation amount per iteration, and the facts are also confirmed by the figures in Table 6.

5 Concluding remarks

Based on the fixed matrix (FM) iterative algorithm, we have attempted to present an inexact fixed matrix (IFM) iterative algorithm for solving large consistent or inconsistent systems of linear inequalities (1.1) in a least squares sense. In addition, the

projection and contraction (PC) method can also be modified to solve the LS problem (1.2). Compared to the FM and IFM iteration methods, PC method is an explicit method and does not need to solve a subproblem at each iteration. However, it is not suitable for solving large dense problems because of its slow convergence.

The main difference of Algorithms FM and IFM lies in the LS subproblems (1.7) and (3.1). (3.1) is a constrained LS subproblem over a special Krylov subspace with lower dimension, which is solved by iteration s of LSQR in exact arithmetic. If the dimension of this Krylov subspace is large enough, the solution of (3.1) is an accurate solution of (1.7). The idea of Algorithm IFM is to use an approximate solution of (1.7) as a new descent direction instead of an accurate one, and we can obtain different descent steps by choosing different s in (3.2). Numerical results showed that Algorithm IFM can approach a solution of (1.2) quickly with less computational effort in each iteration step by setting a small positive integer s. To further save the amount of computation, a practical stopping strategy (4.4) was presented. Although the number of outer iterations is increased by (4.4) as the value of ϵ reduces, the total computational cost may decrease because the inner iteration may terminate within s iterations for lower precision when the estimate x_k reaches the vicinity of the solution point. Moreover, the accurate solution u_k of (1.7) may not be the optimal descent direction for some problems, and the required number of outer iterations for Algorithm IFM may be less than that for Algorithm FM if we take the appropriate values of s and ϵ.

In our experiments, we mainly use LSQR to solving the large LS subproblem (1.7) because of its favorable numerical properties, such as low storage and numerical stability. Many other iterative methods are suitable, especially for large sparse problems; for example, the "column relaxation" GS iteration method mentioned in [8, 10]. The "column relaxation" GS iteration method is to apply the Gauss-Seidel iteration to the normal equation of the LS problem. It can be implemented via a "column relaxation" scheme and does not need to form the matrix $A^T A$ explicitly. Its iterative scheme is rather simple, but it converges more slowly than LSQR, especially for the ill-conditioned problems. Although it is inefficient for solving large LS problems directly, the GS iteration and other stationary iteration can serve as a powerful preconditioner for Krylov subspace methods for large LS problems [20].

Acknowledgments We would like to thank two anonymous referees for their valuable remarks and comments, which helped to greatly improve this paper. We would also like to thank Professors An-ping Liao and Tao-wen Liu for discussions and valuable advice.

References

1. Björck, Å.: Numerical Methods for Least Squares Problems. SIAM, Philadelphia (1996)
2. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)
3. Bramley, R., Winnicka, B.: Solving linear inequalities in a least squares sense. SIAM J. Sci. Comput **17**, 275–286 (1996)
4. Censor, Y., Altschuler, M.D., Powlis, W.D.: A computational solution of the inverse problem in radiation-therapy treatment planning. Appl. Math. Comput **25**, 57–87 (1988)

5. Censor, Y., Ben-Israel, A., Xiao, Y., Galvin, J.M.: On linear infeasibility arising in intensity modulated radiation therapy inverse planning. Linear Algebra Appl **428**, 1406–1420 (2008)
6. Chinneck, J.W.: Feasibility and infeasibility in optimization: algorithms and computational methods. In: International Series in Operations Research and Management Sciences, vol. 118. Springer-Verlag (2007)
7. Davis, T.A.: Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization. ACM Trans. Math. Softw. **38**, 1–22 (2011)
8. Dax, A.: The convergence of linear stationary iterative processes for solving singular unstructured systems of linear equations. SIAM Rev. **32**, 611–635 (1990)
9. Dax, A.: The smallest correction of an inconsistent system of linear inequalities. Optimiz. Eng. **2**, 349–359 (2001)
10. Dax, A.: A hybrid algorithm for solving linear inequalities in a least squares sense. Numer. Algor. **50**, 97–114 (2009)
11. Gilbert, J.R., Moler, C., Schreiber, R.: Sparse matrices in MATLAB: design and implementation. SIAM J. Matrix Anal. Appl **13**, 333–356 (1992)
12. Golub, G.H., Kahan, W.: Calculating the singular values and pseudoinverse of a matrix. SIAM J. Numer. Anal **2**, 205–224 (1965)
13. Golub, G.H., Van Loan, C.F.: Matrix Computations. Johns Hopkins University Press (1983)
14. Han, S.P.: Least squares solution of linear inequalities. Technical Report 2141, Math. Res. Center, University of Wisconsin-Madison (1980)
15. He, B.S.: A projection and contraction method for a class of linear complementarity problem and its application in convex quadratic programming. Appl. Math. Optim. **25**, 247–262 (1992)
16. He, B.S.: Solving a class of linear projection equations. Numer. Math. **68**, 71–80 (1994)
17. Herman, G.T.: A relaxation method for reconstructing object from noisy X-rays. Math. Progr. **8**, 1–19 (1975)
18. Herman, G.T.: Image Reconstruction from Projections: The Fundamentals of Computerized Tomography. Academic Press, New York (1982)
19. Moreau, J.J.: Decomposition orthogonale d'un espace Hilbertien selon deux cânes mutuellement polaires. C.R. Acad. Sci. Paris **225**, 238–240 (1962)
20. Morikuni, K., Hayami, K.: Inner-iteration Krylov subspace methods for least squares problems. SIAM J. Matrix Anal. Appl **34**, 1–22 (2013)
21. Paige, C.C., Saunders, M.A.: LSQR, An algorithm for sparse linear equations and sparse least squares. ACM Trans. Math. Software **8**, 43–71 (1982)
22. Pinar, M.C.: Newton's method for linear inequality systems. Eur. J. Oper. Res **107**, 710–719 (1998)
23. Saad, Y.: Iterative Methods for Sparse Linear Systems. SIAM, Philadelphia (2003)
24. Stewart, G.W.: Matrix Algorithms, Volume 1: Basic Decompositions. SIAM, Philadelphia (1998)
25. Stiefel, E.: Ausgleichung ohne Aufstellung der Gausschen Normalgleichungen. Wiss. Z. Tech. Hochsch. Dresden **2**, 441–442 (1952–1953)
26. Wang, R.S.: Functional Analysis and Optimization Theory. Beijing University of Aeronautics and Astronautics Press (2003). In Chinese
27. Wierzbicki, A.P., Kurcyusz, S.: Projection on a cone, penalty functionals and duality theory for problems with inequality constraints in Hilbert space. SIAM J. Control Optim **15**, 25–56 (1977)
28. Xiu, N., Wang, C., Zhang, J.: Convergence properties of projection and contraction methods for variational inequality problems. Appl. Math. Optim **43**, 147–168 (2001)
29. Zarantonello, E.H.: Projections on Convex Sets in Hilbert Space and Spectral Theory. Academic Press (1971)
30. National Institute of Standards and Technology, MatrixMarket, http://math.nist.gov/MatrixMarket (2002)
31. Sparse QR factorization package, http://www.cise.ufl.edu/research/sparse/SPQR/ (2013)