Perturbed Newton Method with Trust-region Time-stepping Schemes for Linear Programming with Uncertain Data

Xin-long Luo · Yi-yan Yao

Received: date / Accepted: date

Abstract In this article, we consider the path-following method based on the perturbed Newton flow with the new trust-region time-stepping scheme for the standard linear programming problem. For the problem with deficient rank matrix and the noise right-hand-side vector, we also give the pre-processing method based on the singular value decomposition. Then, we analyze the global convergence of the new method when the initial point is strictly primal-dual feasible. Finally, we test the new method for some problems with deficient rank matrices, and compare it with other popular interior-point methods such as the path-following method (the subroutine pathfollow.m coded by M. C. Ferris [14, 16]) and Mehrotra’s predictor-corrector algorithm (the built-in subroutine linprog.m of the MATLAB environment, which was implemented by S. Mehrotra and Y. Zhang [26, 27, 36]). Numerical results show that the new method is more robust than those methods for the large-scale deficient-rank problems without sacrificing its computational efficiency.

Keywords Continuation Newton method · trust-region technique · linear programming · deficient rank · path-following method · uncertain data

Mathematics Subject Classification (2010) 65H17 · 65J15 · 65K05 · 65L05
1 Introduction

In this article, we are mainly concerned with the linear programming problem with uncertain data as follows:

$$\min_{x \in \mathbb{R}^n} c^T x, \quad \text{subject to } Ax = b, \quad x \geq 0,$$

where $c$ and $x$ are vectors in $\mathbb{R}^n$, $b$ is a vector in $\mathbb{R}^m$, and $A$ is an $m \times n$ matrix. For the problem (1), there are many efficient methods to solve it such as the simplex methods and the interior-point methods \cite{14, 16, 28, 34, 36}. Those methods are all assumed to have the consistent system of constraints, i.e. $\text{rank}(A, b) = \text{rank}(A)$.

However, for the real-world problem, since it may exist the redundant constraints and the measurement errors, the rank of matrix $A$ is deficient and the right-hand-side vector $b$ has small noise, which may result in the inconsistent system of constraints \cite{8}. In order to handle this special case, we give a preprocessing method based on the singular value decomposition. Then, according to the first order KKT conditions of the linear programming, we convert the processed problems into the equivalent problem of nonlinear equations with nonnegative constraints. Based on the system of nonlinear equations with nonnegative constraints, we consider a special continuous Newton flow with nonnegative constraints, which has the nonnegative steady-state solution for any nonnegative initial point. In order to improve the robust and efficiency of tracing the trajectory, we construct a perturbed Newton method with the new time-stepping scheme based on the trust-region technique.

The rest of this article is organized as follows. In the next section, we give the continuation Newton method based on the perturbed Newton flow with the new trust-region time-stepping scheme to follow the trajectory and obtain its steady-state solution. Then, in section 3, we analyze the global convergence of the new method when the initial point is strictly feasible. In section 4, for the problems with deficient rank matrices and the noise right-hand-side vectors, some promising numerical results of the new method also reported, with comparison to other popular interior-point methods such as the path-following method (the subroutine pathfollow.m coded by M. C. Ferris \cite{14} \cite{16}) and Mehrotra’s predictor-corrector algorithm (the built-in subroutine linprog.m of the MATLAB environment, which was implemented by S. Mehrotra and Y. Zhang \cite{26} \cite{27} \cite{36}). Numerical results show that the new method is more robust than those methods for the large-scale deficient-rank problems without sacrificing its computational efficiency. Finally, some discussions are also given in section 5. \| \cdot \| denotes the Euclidean vector norm or its induced matrix norm through the paper.

2 Perturbed Newton method with trust-region time-stepping scheme

2.1 The perturbed Newton method

For the linear programming problem (1), it is well known that its optimal solution $x^*$ if and only if it satisfies the following Karush-Kuhn-Tucker conditions (see pp.
Perturbed Newton Method with Trust-region Time-stepping Schemes for Linear Programming

\[ \begin{align*}
Ax - b &= 0, \\
A^T y + s - c &= 0, \\
X s e &= 0, \\
(x, s) &\geq 0,
\end{align*} \]

where
\[ X = \text{diag}(x), \ S = \text{diag}(s), \ \text{and} \ e = (1, \ldots, 1)^T. \]

For the sake of convenience, we regard the optimality conditions (2)-(5) as a system of nonlinear equations with nonnegative constraints as follows:
\[ F(z) = \begin{bmatrix}
Ax - b \\
A^T y + s - c \\
X s e
\end{bmatrix} = 0, \ (x, s) \geq 0, \ \text{and} \ z = (x, y, s). \]

It is not difficult to know that the Jacobian matrix \( J(z) \) of \( F(z) \) defined by equation (7) has the following form:
\[ J(z) = \begin{bmatrix}
A & 0 & 0 \\
0 & A^T & 0 \\
S & 0 & X
\end{bmatrix}. \]

From the third block \( X s e = 0 \) of equation (7), we know that \( x^*_i = 0 \) or \( s^*_i = 0 \) at its solution \( z^* \), where \( x^*_i \) and \( s^*_i \) are the elements of \( x^* \) and \( s^* \), respectively. Thus, the Jacobian matrix \( J(z^*) \) defined by (8) may be singular, which results in numerical difficulties around the solution of the nonlinear system (7) for the Newton’s method or its variants. Therefore, we consider its small positive perturbed system, which is similar to the primal-dual central path \[ \text{[14, 28, 34]} \] as follows:
\[ F_\mu(z) = F(z) - \begin{bmatrix}
0 \\
0 \\
\mu e
\end{bmatrix} = 0, \ (x, s) > 0, \ \mu > 0 \ \text{and} \ z = (x, y, s). \]

We define the strictly feasible region \( \mathbb{P}^0 \) of the linear programming problem (1) as
\[ \mathbb{P}^0 = \{(x, y, s)|Ax = b, \ A^T y + s = c, \ (x, s) > 0\}. \]

Then, when it exist a strictly feasible interior point \( (\bar{x}, \bar{y}, \bar{s}) \in \mathbb{P}^0 \) and the rank of matrix \( A \) is full, the perturbed system (9) has a unique solution (see Theorem 2.8, p.
Its existence can be derived by the implicit theorem [13] and its uniqueness can be proved via considering the strict convexity of the following penalty problem [15]:

$$\min c^T x - \mu \sum_{i=1}^{n} \log(x_i) \text{ subject to } Ax = b,$$

where \(\mu\) is a positive parameter.

According to the duality theorem of the linear programming (Theorem 13.1, pp. 368-369 in [28]), we know

$$c^T x \geq b^T y^* = c^T x^* \geq b^T y^\mu,$$

for any primal-dual feasible solution \((x, y, s)\), where the triple \((x^*, y^*, s^*)\) is a primal-dual optimal solution of nonlinear system (7). Thus, when \(\mu > 0\) is small, the solution \(z^\mu\) is an approximation solution of nonlinear system (7). Consequently, \(x^\mu\) is an approximation of the optimal solution of the original linear programming problem (1). It can be illustrated as follows. Since \(z^\mu\) is the primal-dual feasible, from inequality (12), we have

$$c^T x^\mu \geq b^T y^\mu = c^T x^* \geq b^T y^\mu,$$

and

$$0 \leq (x^\mu)^T s^\mu = c^T x^\mu - b^T y^\mu.$$

From inequalities (13)-(14), it is not difficult to obtain

$$|c^T x^\mu - c^T x^*| \leq (x^\mu)^T s^\mu = n\mu.$$

Therefore, \(x^\mu\) is an approximation of the optimal solution of the original linear programming problem (1).

If we consider the Newton method with a line search strategy for the perturbed system (9) [12, 28], we have

$$z_{k+1} = z_k - \Delta t_k J(z_k)^{-1} F_\mu(z_k),$$

where \(J(z_k)\) is the Jacobian matrix of function \(F_\mu(\cdot)\) at \(z_k\). In equation (16), if we regard \(z_{k+1} = z(t_k + \Delta t_k)\), \(z_k = z(t_k)\) and let \(\Delta t_k \to 0\), we obtain the continuous Newton flow [4, 5, 9, 24, 32] of the perturbed system (9) with the constraints as follows:

$$\frac{dz(t)}{dt} = -J(z)^{-1} F_\mu(z), \quad z = (x, y, s) \text{ and } (x, s) > 0.$$

Actually, if we apply an iteration with the explicit Euler method [31, 35] for the continuous Newton flow (17), we obtain the damped Newton method (16).

Since the Jacobian matrix \(J(z) = F_\mu'(z)\) may be singular, we reformulate the continuous Newton flow (17) as the following more general formula [5, 32]:

$$-J(z) \frac{dz(t)}{dt} = F_\mu(z), \quad z = (x, y, s) \text{ and } (x, s) > 0.$$

The Newton flow (18) has some nice properties. We state them as the following property [4, 5, 24, 32].
Property 1 If \( z(t) \) is the solution of the continuous Newton flow (18), \( f(z(t)) = \|F_{\mu}(z)\|^2 \) converges to zero as \( t \) tends to infinity. Namely, for every limit point \( z^* \) of \( z(t) \), it is also a solution of nonlinear equations (9). Furthermore, every element \( F_{\mu,i}(z) \) of function \( F_{\mu}(z) \) has the linear convergence rate \( \exp(-t) \). If the Jacobian matrix function \( J(z) \) of function \( F_{\mu}(z) \) is nonsingular, \( x(t) \) can not converge to its equilibrium \( z^* \) on finite interval.

**Proof.** For the completeness, we restate its proof as follows. Assume that \( z(t) \) is the solution of the continuous Newton flow (18), then we have

\[
\frac{d}{dt} \left( \exp(t) F_{\mu}(z) \right) = \exp(t) J(z) \frac{dz(t)}{dt} + \exp(t) F_{\mu}(z) = 0,
\]

which gives

\[
F_{\mu}(z) = F_{\mu}(z_0) \exp(-t).
\]

From equation (19), it is easy to see that every element \( F_{\mu,i}(z) \) of function \( F_{\mu}(z) \) converges to zero with linear convergence rate \( \exp(-t) \) when \( t \) tends to infinity. Thus, if the solution \( z(t) \) of the continuous Newton flow (18) belongs to a compact set, it converges to a limit point \( z^* \) when \( t \) tends to infinity, and this limit point \( z^* \) is also a solution of nonlinear equations (9).

Furthermore, if the Jacobian matrix \( J(z) \) of function \( F_{\mu}(z) \) is nonsingular, equation (18) is equivalent to the continuous Newton flow (17). Thus, from the dynamical property of the ordinary differential equation (ODE), we know that the ODE (17) has a unique solution \( z(t) \) and it can not converge to its equilibrium \( z^* \) on finite interval (see pp. 79-82, [30]).

The solution \( x(t) \) of the continuous Newton flow (17) has a nice property. Namely, the inverse Jacobian matrix \( J(x)^{-1} \) can be regarded as the preconditioner of \( F_{\mu}(x) \) such that the solution elements \( x_i(t) (i = 1, 2, \ldots, n) \) have the roughly same convergence rates and it mitigates the stiff property of the ODE (ill-conditioned property) [24]. This property is very useful since it makes us adopt the explicit ODE method to trace the trajectory of the Newton flow.

2.2 The perturbed Newton method

From subsection 2.1, we know that the continuous Newton flow (18) has the nice global convergence property. On the other hand, when the Jacobian matrix \( J(x) \) is singular or nearly singular, the ODE (18) is the system of differential-algebraic equations \([3, 6, 18]\) and its trajectory can not be efficiently followed by the general ODE method such as the backward differentiation formulas (the built-in subroutine bdf15s.m of the MATLAB environment \([26, 31]\)). Thus, we need to construct the special method to handle this problem and we expect that the new method has the same global convergence as the homotopy continuation methods \([1, 25]\) and the fast convergence near the solution \( x^* \) as the traditional optimization methods. In order to attain these
two aims, we consider the continuation Newton method and construct a new time-stepping scheme based on the trust-region technique for problem (18).

If we apply the implicit Euler method to the continuous Newton flow (18) [3, 6], we obtain

\[
J(z_{k+1}) \frac{z_{k+1} - z_k}{\Delta t_k} = -F_{\mu_{k+1}}(z_{k+1}).
\] (20)

The scheme (20) is an implicit formula and it needs to solve a system of nonlinear equations at every iteration. To avoid solving the system of nonlinear equations, we replace \( J(z_{k+1}) \) with \( J(z_k) \) and replace \( F_{\mu_{k+1}}(z_{k+1}) + J(z_k)(z_{k+1} - z_k) \) in equation (20). Thus, we obtain the perturbed Newton method as follows:

\[
z_{k+1} = z_k - \frac{\Delta t_k}{1 + \Delta t_k} J(z_k)^{-1} F_{\mu_{k+1}}(z_k).
\] (21)

The method (21) is similar to the damped Newton method (16) if we regard \( \Delta t_k / (1 + \Delta t_k) \) in equation (21) as \( \Delta t_k \) in equation (16). However, from the view of the ODE method, they are very different. The damped Newton method (16) is obtained by the explicit Euler scheme applied to the continuous Newton flow (18), and its time-stepping \( \Delta t_k \) is restricted by the numerical stability [18, 31, 35]. Namely, the large time-stepping size \( \Delta t_k \) can not be adopted in the steady-state phase. The method (21) is obtained by the semi-implicit Euler scheme applied to the continuous Newton flow (18), and its time-stepping size \( \Delta t_k \) is not restricted by the numerical stability. Therefore, the large time-stepping size can be adopted in the steady-state phase, the perturbed Newton method (21) of which mimics the Newton method and is expected to have the fast convergence rate. The most of all, the continuation Newton method (21) is favourable to adopt the trust-region technique to accurately follow the trajectory of the continuous Newton flow in the transient-state phase and to maintain its fast convergence rate near the equilibrium point \( z^* \).

We select the perturbed factor \( \mu_{k+1} \) as the average of the residual sum [14, 28, 34]:

\[
\mu_{k+1} = \frac{(x_{k+1})^T s_{k+1}}{n}.
\] (22)

In equation (21), \( \mu_{k+1} \) is approximated by \( \sigma_k \mu_k \), where the penalty coefficient \( \sigma_k \) is simply selected as follows:

\[
\sigma_k = \begin{cases} 
0.1, & \text{when } u_k > 0.1, \\
u_k, & \text{when } u_k \leq 0.1.
\end{cases}
\] (23)

Thus, from equations (21), (22), and (23), we have the following iteration scheme:
and

\[(x_{k+1}, y_{k+1}, s_{k+1}) = (x_k, y_k, s_k) + \frac{\Delta t_k}{1 + \Delta t_k} (\Delta x_k, \Delta y_k, \Delta s_k), \quad (25)\]

where the perturbed function \(F_\mu(z)\) is defined by equation (2) and its Jacobian matrix \(J(z)\) is computed by equation (8).

When the rank of matrix \(A\) is full, the linear system (24) can be solved by the following three subsystems:

\[AX_kS_k^{-1}A^T\Delta y_k = -r^k_p + AS_k^{-1}\left(-X_kr^k_d + S_kX_k e - \sigma_k \mu e\right), \quad (26)\]

\[\Delta s_k = -r^k_d - A^T\Delta y_k, \quad (27)\]

\[\Delta x_k = -S_k^{-1}(X_kS_k e + X_k \Delta s_k - \sigma_k \mu e), \quad (28)\]

where the primal residual \(r^k_p\) and the dual residual \(r^k_d\) are defined by

\[r^k_p = Ax_k - b, \quad (29)\]

\[r^k_d = A^T y_k + s_k - c. \quad (30)\]

Since the matrix \(AX_kS_k^{-1}A^T\) is symmetric positive definite when the rank of matrix \(A\) is full and \((X_k, S_k) > 0\), the linear system (26) can be solved efficiently by the Cholesky factorization (see pp. 163-165, [17]).

2.3 The trust-region time-stepping scheme

Another issue is how to adaptively adjust the time-stepping size \(\Delta t_k\) at every iteration. A popular way to control the time-stepping size is based on the trust-region technique [7][10][19][21][22][24]. Its main idea is that the time-stepping size \(\Delta t_{k+1}\) will be enlarged when the linear model \(F_{\sigma_k}(z_k) + J(z_k)(z_{k+1} - z_k)\) approximates \(F_{\sigma_k}(z_{k+1})\) well, and \(\Delta t_{k+1}\) will be reduced when \(F_{\sigma_k}(z_k) + J(z_k)(z_{k+1} - z_k)\) approximates \(F_{\sigma_k}(z_{k+1})\) badly. We adopt the following ratio \(\rho_k\) as the measurement between the linear approximation model and the residual function:

\[\rho_k = \frac{\|F_{\sigma_k}(z_k)\| - \|F_{\sigma_k}(z_{k+1})\|}{\|F_{\sigma_k}(z_k)\| - \|F_{\sigma_k}(z_k) + J(z_k)(z_{k+1} - z_k)\|}, \quad (31)\]

Thus, based on the measurement model (31), we give the following trust-region time-stepping scheme:

\[\Delta t_{k+1} = \begin{cases} 2\Delta t_k, & \text{if } 0 \leq |1 - \rho_k| \leq \eta_1 \text{ and } (x_{k+1}, s_{k+1}) > 0, \\ \Delta t_k, & \text{else if } \eta_1 < |1 - \rho_k| < \eta_2 \text{ and } (x_{k+1}, s_{k+1}) > 0, \\ \frac{1}{2}\Delta t_k, & \text{others}, \end{cases} \quad (32)\]

where the constants are selected as \(\eta_1 = 0.25, \eta_2 = 0.75\), according to our numerical experiments.
2.4 The treatment of deficient rank

For a real-world problem, the rank of matrix $A$ in problem (1) may be deficient and the constraint system is even inconsistent when the right-hand-side vector $b$ has small noise [23]. For the consistent system of constraints with the deficient rank of matrix $A$, there are some pre-solving methods to eliminate the redundant constraints [2]. Here, in order to handle the inconsistent system of constraints, we consider the following best approximation problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|$$ (33)

and obtain the reduced system of constraints of problem (1).

Firstly, we factorize the matrix $A$ with its singular value decomposition (see pp. 76-80, [17]) as follows:

$$A = U \Sigma V^T, \quad \Sigma = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix}, \quad \Sigma_r = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r) \succ 0,$$ (34)

where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal matrices, and $r$ is the rank of matrix $A$. Then, from equation (34), we know that problem (33) equals the following linear least-squares problem

$$\min_{x \in \mathbb{R}^n} \|\Sigma V^T x - U^T b\|.$$ (35)

From equation (35), we obtain its solution

$$V_r^T x = b_r,$$ (36)

where $V_r = V(1 : n, 1 : r)$, $V_r^T V_r = I$, and $b_r = ((U^T b)(1 : r))./\text{diag}(\Sigma(1 : r, 1 : r))$.

Therefore, when the constraints of problem (1) is consistent, problem (1) equals the following linear programming problem:

$$\min_{x \in \mathbb{R}^n} c^T x \text{ subject to } V_r^T x = b_r, \quad x \geq 0.$$ (37)

When the constraints of problem (1) is inconsistent, problem (37) is the best relaxation approximation of the original problem (1). Consequently, in subsection 2.2 it only needs to replace matrix $A$ and vector $b$ with matrix $V_r^T$ and vector $b_r$ respectively, then the perturbed Newton method can handle the deficient rank problem.

From the reduced linear programming problem (37), we obtain its KKT conditions as follows:

$$V_r x - b_r = 0,$$ (38)

$$V_r^T y_r + s - c = 0,$$ (39)

$$XSe = 0, \quad i = 1, 2, \ldots, n,$$ (40)

$$(x, s) \geq 0,$$ (41)
where \( X = \text{diag}(x) \) and \( S = \text{diag}(s) \). Thus, from the singular decomposition (33) of matrix \( A \) and equation (36), if \((x_r, y_r, s_r)\) is a solution of nonlinear equations (38)-(41), we know that \((x_r, U(S^{-1}y_r; 0), s_r)\) is a solution of nonlinear equations (2)-(5). Consequently, we recover the solution of the KKT conditions of the original problem (1) from the solution of the KKT conditions of the reduced problem (37).

According to the above discussions, we give the detailed descriptions of the perturbed Newton method with the trust-region time-stepping scheme for linear programming problems (1) in Algorithm 1.

3 Algorithm analysis

In order to simplify the convergence analysis of Algorithm 1, we assume that (i) the initial point \((x_0, s_0)\) belongs to the strictly primal-dual feasible region \(\mathbb{R}^0\) defined by equation (10), and (ii) the time-stepping size \(\Delta t_k\) is selected such that \((x_k, y_k, s_k)\) satisfies the one-sided \(\infty\)-norm neighborhood \(\mathbb{N}_{\infty}(\gamma)\), which is defined by

\[
\mathbb{N}_{\infty}(\gamma) = \{(x, y, s) \in \mathbb{R}^0 | XSe \geq \gamma \mu e\},
\]

(42)

where \( X = \text{diag}(x), S = \text{diag}(s), e = [1, 1, \ldots, 1]^T, \mu = x^T s/n \) and \( \gamma \) is a small positive constant such as \( \gamma = 10^{-3} \). Without the loss of generality, we assume that the matrix \( A \in \mathbb{R}^{m \times n} \) has full rank.

**Lemma 1** Assume that the strictly primal-dual feasible point \((x_k, y_k, s_k)\) \(\in \mathbb{R}^0\) satisfies the proximity condition (42), then it exists a sufficiently small positive number \(\alpha_k > 0\) such that

\[
(x_k(\alpha_k), y_k(\alpha_k), s_k(\alpha_k) = (x_k, y_k, s_k) + \alpha_k(\Delta x_k, \Delta y_k, \Delta s_k)
\]

satisfies the proximity condition (42), where \((\Delta x_k, \Delta y_k, \Delta s_k)\) is solved by the linear system (24) and

\[
\alpha_k = \frac{\Delta t_k}{1 + \Delta t_k}.
\]

**Proof.** Since \((x_k, y_k, s_k)\) is a strictly primal-dual feasible point, from equation (24), we know that

\[
A\Delta x_k = 0, A^T \Delta y_k + \Delta s_k = 0, k = 0, 1, \ldots \quad (43)
\]

Thus, from equation (43), we have

\[
\Delta x_k^T \Delta s_k = -\Delta x_k^T A^T \Delta y_k = (A \Delta s_k)^T \Delta y_k = 0, k = 0, 1, \ldots \quad (44)
\]

and

\[
Ax_k(\alpha_k) = b, A^Ty_k(\alpha_k) + s_k(\alpha_k) = c, k = 0, 1, \ldots \quad (45)
\]
Algorithm 1 Perturbed Newton Method with the trust-region time-stepping scheme for linear programming (The PNMTRLP method)

Input:
matrix $A \in \mathbb{R}^{m \times n}$, vectors $b \in \mathbb{R}^{m}$ and $c \in \mathbb{R}^{n}$ for the linear programming problem:
$\text{min}_{x \in \mathbb{R}^{n}} c^T x$ subject to $Ax = b$, $x \geq 0$.

Output:
the primal-dual optimal solution: $(sols, soly, solz)$,
the maximum error of KKT conditions: $\text{KKT Error} = \max \{||A \ast solx - b||_{\infty}, ||A^T \ast soly + sols - c||_{\infty}, ||solx \ast soly||_{\infty}\}$.

1. Initialize the trust-region parameters: $\eta_{0} = 10^{-6}$, $\eta_{1} = 0.25$, $\eta_{2} = 0.75$, and set $\epsilon = 10^{-6}$, $\Delta_{0} = 0.9$.
2. Factorize matrix $A$ with the singular value decomposition as follows:
   $A = U \Sigma V^T$, $\Sigma = \text{diag}(\lambda_{1}, \lambda_{2}, \ldots, \lambda_{r}, 0, \ldots, 0)$, and denote $V_{r} = V(1 : n, 1 : r)$, $b_{r} = ((U^T b)(1 : r))$, ($\text{diag}(\Sigma(1 : r, 1 : r))$), where $r$ is the rank of matrix $A$.
3. Set $\text{bigM} = \max(\max(\text{abs}(V_{r})), \text{bigM})$, $\text{bigM} = \max(\{\text{norm}(b_{r}, \infty), \text{norm}(c, \infty), \text{bigM}\}$.
4. Initialize $x_{0} = \text{bigMf} \ast \text{bigM} \ast \text{ones}(n, 1), y_{0} = x_{0}, y_{0} = \text{zeros}(r, 1)$.
5. flag_success_trialstep = 1; % Indicates whether time-stepping size is accepted
6. while (itc < maxit) do
7.     if (flag_success_trialstep == 1) then
8.         Set $\text{itc} = \text{itc} + 1$.
9.         Compute $F_{2} = V_{r}^T x_{k} - b_{r}$, $F_{2} = V_{r} y_{k} + s_{k} - c$, $F_{2} = X_{k} y_{k}$; $\mu_{k} = x_{k}^T s_{k} / n$;
10.        Compute $\text{Resk}_{inf} = \text{norm}([F_{2}; F_{2}; F_{2}])$, inf.
11.       if ($\text{Resk}_{inf} < \epsilon$) then
12.           break;
13.       end if
14.       Set $\Delta c = \min(0.1, \mu_{k})$.
15.       Compute $F_{2} = F_{2} - \Delta c \mu_{k} \ast \text{ones}(n, 1)$, and set $F_{2} = [F_{2}; F_{2}; F_{2}]$.
16.       Use the Cholesky factorization to solve the system of linear equations:
17.         $V_{r}^T X_{k} S_{k}^{-1} V_{r} \Delta y_{k} = -F_{2} + V_{r}^T X_{k} S_{k}^{-1} (X_{k} F_{2} - F_{2})$.
18.         Compute $\Delta y_{k} = -F_{2} - V_{r} \Delta y_{k}$ and $\Delta x_{k} = -S_{k}^{-1} (F_{2} + X_{k} \Delta y_{k})$.
19.       end if
20.       Compute $(s_{k+1}, y_{k+1}, s_{k+1}) = (s_{k}, y_{k}, s_{k}) + \frac{\Delta x_{k}}{\mu_{k}}(\Delta x_{k}, \Delta y_{k}, \Delta s_{k})$.
21.       Compute $F_{2+1} = V_{r}^T X_{k+1} - b_{r}$, $F_{2+1} = V_{r} y_{k+1} + s_{k+1} - c$, $X_{k+1} = X_{k+1} + \Delta x_{k} - \Delta y_{k} \ast \text{ones}(n, 1)$,
22.         $F_{2+1} = [F_{2+1}; F_{2+1}; F_{2+1}]$.
23.       Compute $\text{LinApp}_{\text{tr}} = [F_{2+1} - F_{2+1}; (F_{2+1} + X_{k+1} - s_{k+1}) + S_{k}(s_{k+1} - s_{k})]$.
24.       Compute the ratio $\rho_{k} = (||F_{2+1}|| - ||F_{2+1}||) / (||F_{2}|| - ||\text{LinApp}_{\text{tr}}||)$.
25.       if ($(|\rho_{k} - 1| \leq \eta_{1})$&&&(s_{k+1}+s_{k+1} > 0)) then
26.           Set $\Delta s_{k+1} = 2 \Delta t_{k}$;
27.       else if ($|\rho_{k} - 1| \leq \eta_{2}$)&&&(s_{k+1}+s_{k+1} > 0)) then
28.           Set $\Delta s_{k+1} = \Delta t_{k}$;
29.       else
30.           Set $\Delta s_{k+1} = 0.5 \Delta t_{k}$;
31.       end if
32.       if ($|\rho_{k} \geq \eta_{3}$)&&&(s_{k+1}+s_{k+1} > 0)) then
33.           Accept the trial step (s_{k+1}+y_{k+1}, s_{k+1}); flag_success_trialstep = 1;
34.       else
35.           Set (s_{k+1}, y_{k+1}, s_{k+1}) = (s_{k}, y_{k}, s_{k}); flag_success_trialstep = 0;
36.       end if
37.       Set k = k + 1.
38. end while
39. Return $(sols, soly, solz) = (s_{k}, y_{k}, s_{k})$; $\text{KKT Error} = \max \{||A x_{k} - b||_{\infty}, ||A^T y_{k} + s_{k} - c||_{\infty}, ||s_{k} \ast s_{k}||_{\infty}\}$.
Since \( \alpha_k = \Delta t_k / (\Delta t_k + 1) \), from equation (25), we have
\[
X_k(\alpha_k)S_k(\alpha_k) = (X_k + \alpha_k \Delta X_k)(S_k + \alpha_k \Delta S_k) \\
= X_k S_k + \alpha_k (X_k \Delta S_k + S_k \Delta X_k) + \alpha_k^2 \Delta X_k \Delta S_k.
\]
(46)

Replace \( X_k \Delta S_k + S_k \Delta X_k \) with equation (28) into equation (46), then we have
\[
X_k(\alpha_k)S_k(\alpha_k) = (X_k \alpha_k)S_k(\alpha_k) = (X_k + \alpha_k \Delta X_k)e + \alpha_k (\sigma_k \mu_k - X_k S_k e) + \alpha_k^2 \Delta X_k \Delta S_k e \\
= (1 - \alpha_k)X_k S_k e + \alpha_k \sigma_k \mu_k + \alpha_k^2 \Delta X_k \Delta S_k e.
\]
(47)

From equation (44) and equation (47), we obtain
\[
\mu(\alpha_k) = \frac{1}{n} e^T X_k(\alpha_k)S_k(\alpha_k) e = (1 - \sigma_k) \alpha_k \mu_k.
\]
(48)

We denote
\[
\beta^k_{max} = \max_{1 \leq i \leq n} \{|\Delta x_i^k|, |\Delta s_i^k|\}.
\]
(49)

Then, from equation (47), we have
\[
X_k(\alpha_k)S_k(\alpha_k) e \geq (1 - \alpha_k) \gamma \mu_k + \alpha_k \sigma_k \mu_k - \alpha_k^2 \beta^k_{max}.
\]
(50)

From equation (48) and inequality (50), we know that the proximity condition
\[
X_k(\alpha_k)S_k(\alpha_k) e \geq \gamma \mu_k(\alpha_k)
\]
is satisfied, provided that
\[
(1 - \alpha_k) \gamma \mu_k + \alpha_k \sigma_k \mu_k - \alpha_k^2 \beta^k_{max} \geq \gamma (1 - \alpha_k + \alpha_k \sigma_k) \mu_k.
\]

Reformulate this expression, then we obtain
\[
\alpha_k (1 - \gamma) \sigma_k \mu_k \geq \alpha_k^2 \beta^k_{max}.
\]
(51)

Therefore, if we select
\[
\alpha_k \leq \frac{(1 - \gamma) \sigma_k \mu_k}{\beta^k_{max}},
\]
(52)

inequality (51) is true. Consequently, it exists the sufficiently small positive number \( \alpha_k \) such that \( (x_k(\alpha_k), y_k(\alpha_k), s_k(\alpha_k)) \) satisfies the proximity condition (42). \( \square \)

In the following Lemma 2, we give the lower bounded estimation of \( (x_k, s_k) \) when the initial point \( (x_0, y_0, s_0) \) is strictly primal-dual feasible and satisfies the proximity condition (42).
**Lemma 2** Assume that the initial point \((x_0, y_0, s_0) \in \mathbb{R}^n\) is strictly primal-dual feasible, and \((x_k, y_k, s_k) (k = 0, 1, \ldots)\) generated by Algorithm 1 satisfy the proximity condition (42). Furthermore, if it exists a constant \(C_\mu\) such that

\[
\mu_k \geq C_\mu > 0, \quad k = 0, 1, \ldots,
\]

we have the bounded estimations of \((x_k, s_k)\) as follows:

\[
0 < C_{\min} \leq \min_{1 \leq i \leq n} \{x'_i, s'_i\} \leq \max_{1 \leq i \leq n} \{x'_i, s'_i\} \leq C_{\max}, \quad k = 0, 1, \ldots
\]

where \(C_{\min}\) and \(C_{\max}\) are two positive constants.

**Proof.** Since \((x_k, y_k, s_k)\) is generated by Algorithm 1 from equation (48), we have

\[
\mu_{k+1} = \frac{x'_{k+1} + s'_{k+1}}{n} = (1 - (1 - \sigma_k)\alpha_k)\mu_k \leq \mu_k, \quad k = 0, 1, \ldots
\]

which gives

\[
\mu_{k+1} = \prod_{i=0}^{k} (1 - (1 - \sigma_i)\alpha_i)\mu_i \leq \mu_0, \quad k = 0, 1, \ldots \tag{55}
\]

Furthermore, since the initial point \((x_0, s_0, y_0)\) is strictly primal-dual feasible, from Lemma 1 and the mathematical induction, we know that \((x_k, y_k, s_k)\) is strictly primal-dual feasible. Thus, from equation (45), we have

\[
A(x_k - x_0) = 0, \quad A^T(y_k - y_0) + (s_k - s_0) = 0,
\]

which gives

\[
(x_k - x_0)^T(s_k - s_0) = 0.
\]

Rearrange this expression, from equation (55), then we obtain

\[
x_k' - x_0 = x_k' + x_0 s_0 \leq n(\mu_k + \mu_0) \leq 2n\mu_0,
\]

which gives

\[
x'_k \leq \frac{2n\mu_0}{\min_{1 \leq i \leq n} \{s'_i\}} \quad \text{and} \quad s'_k \leq \frac{2n\mu_0}{\min_{1 \leq i \leq n} \{x'_i\}}, \quad 1 \leq i \leq n, \quad k = 0, 1, \ldots
\]

Therefore, if we selected

\[
C_{\max} = \max \left\{ \frac{2n\mu_0}{\min_{1 \leq i \leq n} \{x'_i\}}, \frac{2n\mu_0}{\min_{1 \leq i \leq n} \{s'_i\}} \right\},
\]

we obtain

\[
\max_{1 \leq i \leq n} \{x'_i, s'_i\} \leq C_{\max}, \quad k = 0, 1, \ldots \tag{56}
\]
On the other hand, from the assumption \([53]\) and the proximity condition \([42]\), we have
\[
x_i^j s_k^j \geq \gamma \mu_k \geq \gamma C_{\mu}, \quad 1 \leq i \leq n, \quad k = 0, 1, \ldots
\]
Combining the upper estimation \([56]\) of \((x_k, s_k)\), we obtain
\[
x_k^j = \frac{\gamma C_{\mu}}{\max_{1 \leq j \leq n} \{s_k^j\}} \geq \frac{\gamma C_{\mu}}{C_{\max}}, \quad \text{and} \quad s_k^j \geq \frac{\gamma C_{\mu}}{\max_{1 \leq j \leq n} \{s_k^j\}} \geq \frac{\gamma C_{\mu}}{C_{\max}}, \quad k = 0, 1, \ldots,
\]
which concludes that
\[
\min_{1 \leq i \leq n} \{x_k^j, s_k^j\} \geq C_{\min}, \quad k = 0, 1, \ldots.
\]
if we select \(C_{\min} = \gamma C_{\mu}/C_{\max}\).

**Lemma 3** Assume that the initial point \((x_0, y_0, s_0) \in \mathbb{R}^n\) is strictly primal-dual feasible, and \((x_k, y_k, s_k) (k = 0, 1, \ldots)\) generated by Algorithm 1 satisfy the proximity condition \([42]\). Furthermore, if the assumption \([53]\) is true, it exists positive constants \(C_{\Delta x}\) and \(C_{\Delta s}\) such that
\[
\|\Delta s_k\| \leq C_{\Delta s}, \quad \text{and} \quad \|\Delta x_k\| \leq C_{\Delta x}, \quad k = 0, 1, \ldots, \quad (57)
\]

**Proof.** Factorize matrix \(A\) with the singular value decomposition \([54]\). Then, from the bounded estimation \([54]\) of \((x_k, s_k)\) in Lemma 2, we have
\[
vA X S^{-1} A^T v \geq \frac{C_{\min}}{C_{\max}} \|Av\|^2 \geq \frac{C_{\min} \lambda_{\min}^2}{C_{\max}} \|v\|^2, \quad \forall v \in \mathbb{R}^n, \quad (58)
\]
and
\[
vA X S^{-1} A^T v \leq \frac{C_{\max}}{C_{\min}} \|Av\|^2 \leq \frac{C_{\max} \lambda_{\max}^2}{C_{\min}} \|v\|^2, \quad \forall v \in \mathbb{R}^n, \quad (59)
\]
where \(\lambda_{\min}\) and \(\lambda_{\max}\) are the smallest and largest singular values of matrix \(A\), respectively. Since \((x_0, y_0, s_0)\) is strictly primal-dual feasible, from equations \([24, 25]\), Algorithm 1, and the mathematical induction, we know that \((x_k, y_k, s_k) (k = 0, 1, \ldots)\) are strictly primal-dual feasible. Consequently, from equations \([25, 54]\) and \([58, 59]\), we obtain
\[
\frac{C_{\min} \lambda_{\min}^2}{C_{\max}} \|\Delta y_k\|^2 \leq \Delta y_k^T (A X_k S^{-1} A^T) \Delta y_k = \Delta y_k^T A S_k^{-1} (X_k S_k e - \sigma_k \mu_k e)
\]
\[
\leq \|\Delta y_k\| \|A\| \|S_k^{-1}\| \|X_k S_k e - \sigma_k \mu_k e\| \leq \|\Delta y_k\| \frac{\lambda_{\max}}{C_{\min}} (\|X_k S_k e\| + n \sigma_k \mu_k)
\]
\[
\leq \|\Delta y_k\| \frac{\lambda_{\max}}{C_{\min}} (\|X_k S_k e\| + n \sigma_k \mu_k) = \|\Delta y_k\| \frac{\lambda_{\max}}{C_{\min}} (1 + \sigma_k) n \mu_k,
\]
which gives
\[
\|\Delta y_k\| \leq \frac{C_{\max} \lambda_{\max}}{C_{\min} \lambda_{\min}^2} 2n \mu_k \leq \frac{C_{\max} \lambda_{\max}}{C_{\min} \lambda_{\min}^2} 2n \mu_0 \quad (60)
\]
where we use the properties \( \sigma_k \leq 1 \) from equation (53) and \( \mu_k \leq \mu_0 \) from inequality (55). Therefore, from equation (27) and inequality (60), we have

\[
\|\Delta s_k\| = \| -A^T \Delta y_k \| \leq \|A^T\| \|\Delta y_k\| \leq \frac{C_{\max}^2}{C_{\min}^2} 2n\mu_0 = C_{\Delta t}.
\]

Thus, we prove the first part of inequality (57).

From equation (28), inequality (54) and the first part of inequality (57), we have

\[
\|\Delta s_k\| = \left\| -S_k^{-1} (X_k^T s_k e + X_k \Delta s_k - \sigma_k \mu_k e) \right\| \leq \left\| S_k^{-1} \right\| \|X_k^T s_k e + X_k \Delta s_k - \sigma_k \mu_k e\|
\]

\[
\leq \frac{1}{C_{\min}} (\|X_k^T s_k e\| + \|X_k \Delta s_k\| + \|\sigma_k \mu_k e\|)
\]

\[
\leq \frac{1}{C_{\min}} (\|X_k^T s_k e\| + \|X_k \|\Delta s_k\| + n \sigma_k \mu_k )
\]

\[
\leq \frac{1}{C_{\min}} (n \mu_k + C_{\max} C_{\Delta t} + n \sigma_k \mu_k ) \leq \frac{1}{C_{\min}} (2n \mu_0 + C_{\max} C_{\Delta t} ) = C_{\Delta t},
\]

where we use the properties \( \sigma_k \leq 1 \) from equation (23) and \( \mu_k \leq \mu_0 \) from inequality (55). Thus, we also prove the second part of inequality (57).

**Lemma 4** Assume that the initial point \( (x_0, y_0, s_0) \in \mathbb{R}^0 \) is strictly primal-dual feasible, and \( (x_k, y_k, s_k) \) \( (k = 0, 1, \ldots) \) generated by Algorithm 7 satisfy the proximity condition (42). Furthermore, if the assumption (53) is true, it exists a constant \( C_{\Delta t} \) such that

\[
\Delta t_k \geq C_{\Delta t} > 0, \quad k = 0, 1, \ldots
\]

**Proof.** From equation (31) and equations (24)-(25), we have

\[
|\rho_k - 1| = \frac{\|F_{\sigma_k \mu_k} (z_k)\| - \|F_{\sigma_k \mu_k} (z_{k+1})\|}{\left\| F_{\sigma_k \mu_k} (z_k) \right\| - \| F_{\sigma_k \mu_k} (z_k) + J(z_k) (z_{k+1} - z_k) \|} - 1
\]

\[
= \frac{\left\| F_{\sigma_k \mu_k} (z_k) \right\| - \| F_{\sigma_k \mu_k} (z_k) \| - \| F_{\sigma_k \mu_k} (z_k) + J(z_k) (z_{k+1} - z_k) \|}{\left\| F_{\sigma_k \mu_k} (z_k) \right\| - \| F_{\sigma_k \mu_k} (z_k) \| - \| F_{\sigma_k \mu_k} (z_k) + J(z_k) (z_{k+1} - z_k) \|} - 1
\]

\[
= \frac{\Delta t_k \|F_{\sigma_k \mu_k} (z_k)\|}{\| \Delta X_k \Delta S_k e \|}
\]

\[
= \frac{\Delta t_k \|\Delta X_k \Delta S_k e\|}{\|\Delta X_k \Delta S_k e\|}.
\]

In the last equality of equation (62), we use the property

\[
F_{\sigma_k \mu_k} (z_{k+1}) - F_{\sigma_k \mu_k} (z_k) = F_{\sigma_k \mu_k} \left( z_k + \frac{\Delta t_k}{1 + \Delta t_k} \Delta z_k \right) - F_{\sigma_k \mu_k} (z_k)
\]

\[
= \frac{\Delta t_k}{1 + \Delta t_k} J(z_k) \Delta z_k + \left( \frac{\Delta t_k}{1 + \Delta t_k} \right)^2 \Delta X_k \Delta S_k e
\]

\[
= \frac{\Delta t_k}{1 + \Delta t_k} F_{\sigma_k \mu_k} (z_k) + \left( \frac{\Delta t_k}{1 + \Delta t_k} \right)^2 \Delta X_k \Delta S_k e.
\]
On the other hand, using the property $\|a\| \geq a_i (i = 1, 2, \ldots, n)$, we have

$$\|X_k s_k - \sigma_k \mu_k e\| \geq x_k^T s_k - \sigma_k \mu_k, \ i = 1, 2, \ldots, n.$$  

By summing the $n$ components of the above two sides and the definition $\mu_k = x_k^T s_k / n$, we obtain

$$\|X_k s_k - \sigma_k \mu_k e\| \geq (1 - \sigma_k) \mu_k \geq (1 - 0.1) C_\mu = 0.9 C_\mu,$$  

where we use the properties $\sigma_k \leq 0.1$ from equation (23) and $\mu_k \geq C_\mu$ from the assumption (53).

Thus, from the bounded estimation (57) of $(\Delta x_k, \Delta s_k)$ and inequalities (62), (63), we obtain

$$|\rho_k - 1| \leq \frac{\Delta t_k}{1 + \Delta t_k} \frac{\|\Delta X_k \Delta S_k e\|}{0.9 C_\mu} \leq \frac{\Delta t_k}{1 + \Delta t_k} \frac{\|\Delta X_k\| \|\Delta S_k\|}{0.9 C_\mu} \leq \frac{\Delta t_k}{1 + \Delta t_k} \frac{C_{\Delta x} C_{\Delta s}}{0.9 C_\mu} \leq \eta_1,$$

provided that

$$\Delta t_k \leq \frac{0.9 C_\mu \eta_1}{C_{\Delta x} C_{\Delta s}}.$$  

Thus, if we assume that $K$ is the first index such that $\Delta t_K$ satisfies inequality (64), according to the trust-region time-stepping scheme (32), $\Delta t_{K+1}$ will be enlarged. Therefore, we prove the result (61) if we select $C_{\Delta t}$ as $\Delta t_K$. 

According to the above discussions, we give the global convergence analysis of Algorithm 1.

**Theorem 1** Assume that the initial point $(x_0, y_0, s_0) \in F^0$ is strictly primal-dual feasible, and $(x_k, y_k, s_k) (k = 0, 1, \ldots)$ generated by Algorithm 1 satisfy the proximity condition (42). Then, we have

$$\lim_{k \to \infty} \mu_k = 0,$$  

where $\mu_k = x_k^T s_k / n$.

**Proof.** Assume that it exists a positive constant $C_\mu$ such that

$$\mu_k \geq C_\mu, \ k = 0, 1, \ldots.$$  

Then, according to the result of Lemma 3 we know that it exists a positive constant $C_{\Delta t}$ such that

$$\Delta t_k \geq C_{\Delta t}, \ k = 0, 1, \ldots.$$
Therefore, from equation (48), we have
\[
\mu_{k+1} = \mu_k (\alpha_k) = (1 - (1 - \sigma_k) \alpha_k) \mu_k \leq \left( 1 - (1 - \sigma_k) \frac{C_{A^T}}{1 + C_{A^T}} \right) \mu_k
\]
\[
\leq \left( 1 - 0.9 \frac{C_{A^T}}{1 + C_{A^T}} \right) \mu_k \leq \left( 1 - 0.9 \frac{C_{A^T}}{1 + C_{A^T}} \right)^{k+1} \mu_0,
\]
where we use the property \( \sigma_k \leq 0.1 (k = 0, 1, \ldots) \) from the definition (23) of \( \sigma_k \). Thus, we have \( \mu_k \to 0 \), which contradicts the assumption (66). Consequently, we obtain \( \lim_{k \to \infty} \inf \mu_k = 0 \). Since \( \mu_k \) is monotonically decreasing, it is not difficult to know \( \lim_{k \to \infty} \|X_k s_k\| = 0 \). Furthermore, we obtain \( \lim_{k \to \infty} \|X_k s_k\| = 0 \) from \( \|X_k s_k\| \leq \|X_k s_k\| \leq n \mu_k \) and \( (x_k, s_k) > 0 \).

4 Numerical experiments

In this section, we test Algorithm 1 (the PNMTLP method) for some linear programming problems with full rank matrix \( A \) or deficient rank matrix \( A \), and compare it with the excellent path-following method (pathfollow.m) coded by M. C. Ferris in [14] and the state-of-the-art Mehrotra’s predictor-corrector algorithm (the built-in subroutine linprog.m of the MATLAB environment, which was implemented by S. Mehrotra and Y. Zhang [26, 27, 36]).

The tolerance of three methods is set to \( \varepsilon = 10^{-6} \) by default. We use the maximum absolute error (KKTError) of the KKT conditions (2)-(5) and the primal-dual gap \( x^T s \) to measure the error between the numerical optimal solution and the theoretical optimal solution.

4.1 The problem with the full rank matrix

For the standard linear programming problem with the full rank matrix \( A \), the sparse matrix \( A \) of given density 0.2 is randomly generated and we choose feasible \( x, y, s \) at random, with \( x \) and \( s \) each about half-full. The dimension of matrix \( A \) varies from \( 10 \times 100 \) to \( 300 \times 3000 \). One of its implementation is illustrated by Algorithm 2 (p. 210, [14]). According to Algorithm 2, we randomly generate 30 standard linear programming problems with full rank matrices.

For those 30 test problems, we compare Algorithm 1 (the PNMTLP method), Mehrotra’s predictor-corrector algorithm (the subroutine linprog.m of the MATLAB environment) and the path-following method (the subroutine pathfollow.m). The numerical results are put in Table 1 and Figure 1. The left sub-figure of Figure 1 represents the number of iterations and its right sub-figure represents the consumed CPU time. From Table 1, we find that the PNMTLP method and the subroutine linprog.m can solve all test problems, and their KKTError are small. However, the subroutine pathfollow.m cannot perform well in some higher-dimensional problems, such as examples 7, 14, 16, 21, 23, 24, 27, 28, 30, since their solutions do not satisfy the
Algorithm 2 The standard linear programming problem with the full rank matrix

**Input:**
- the number of equality constraints: $m$;
- the number of unknown variables: $n$.

**Output:**
- matrix $A$ and vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$.

1: density=0.2;
2: $A = \text{sprandn}(m,n,\text{density})$; % Generate a sparse matrix of give density.
3: $xfeas = [\text{rand}(n/2,1); \text{zeros}(n-(n/2),1)]$;
4: $sfeas = [\text{zeros}(n/2,1); \text{rand}(n-(n/2),1)]$;
5: $xfeas = xfeas(\text{randperm}(n))$;
6: $sfeas = sfeas(\text{randperm}(n))$;
7: $yfeas = (\text{rand}(m,1)-0.5)*4$; % Choose $b$ and $c$ to make this $(x,y,s)$ feasible.
8: $b = A*xfeas$;
9: $c = A'*yfeas + sfeas$;

KKT conditions. From Figure 1, we also find that the subroutine `linprog.m` performs the best and the number of its iterations is less than 20, the number of iterations of the PNMTRLP method is around 20, and the number of iterations of the subroutine `pathfollow.m` often reaches the maximum number (i.e. 200 iterations). Therefore, the PNMTRLP method is also an efficient and robust path-following method for linear programming problems with full rank matrices.

![iterations and consumed time](image)

**Fig. 1:** The number of iterations and the consumed time.

4.2 The problem with the deficient rank matrix and uncertain data

For some practical problems, since it may exist redundant constraints and measurement errors, the rank of matrix $A$ may be deficient and the right-hand-side vec-
Table 1: Numerical results of problems with full rank matrices.

| Problem ($m \times n, r$) | PNMTRLP | linprog | pathfollow |
|---------------------------|---------|---------|------------|
|                           | KKTError | Gap     | KKTError | Gap     | KKTError | Gap     |
| Exam. 1 ($10 \times 100$, 10) | 6.40E-07 | 4.58E-06 | 2.23E-09 | 1.56E-08 | 2.47E-07 | 2.61E-06 |
| Exam. 2 ($20 \times 200$, 20) | 1.12E-06 | 1.04E-04 | 5.39E-10 | 8.56E-08 | 3.24E-08 | 4.84E-06 |
| Exam. 3 ($30 \times 300$, 30) | 9.9E-07  | 1.39E-04 | 2.99E-11 | 1.15E-09 | 1.83E-09 | 4.48E-07 |
| Exam. 4 ($40 \times 400$, 40) | 5.84E-06 | 1.04E-03 | 2.75E-10 | 6.88E-08 | 5.59E-07 | 1.66E-04 |
| Exam. 5 ($50 \times 500$, 50) | 2.48E-06 | 5.01E-04 | 1.24E-09 | 1.87E-07 | 1.01E-06 | 3.64E-04 |
| Exam. 6 ($60 \times 600$, 60) | 2.45E-06 | 6.69E-04 | 3.53E-08 | 7.73E-06 | 1.08E-07 | 4.42E-05 |
| Exam. 7 ($70 \times 700$, 70) | 4.22E-06 | 1.20E-03 | 7.00E-10 | 2.53E-07 | 49327.15 | 5.48E-04 |
| Exam. 8 ($80 \times 800$, 80) | 3.66E-06 | 1.41E-03 | 6.41E-12 | 2.03E-09 | 4.03E-07 | 2.64E-04 |
| Exam. 9 ($90 \times 900$, 90) | 1.70E-06 | 7.34E-04 | 5.36E-12 | 1.66E-09 | 2.62E-08 | 1.64E-05 |
| Exam. 10 ($100 \times 1000$, 100) | 4.10E-06 | 1.60E-03 | 1.69E-09 | 3.81E-07 | 4.6E-07  | 3.59E-04 |
| Exam. 11 ($110 \times 1100$, 110) | 1.96E-05 | 9.32E-03 | 2.63E-07 | 7.05E-05 | 5.85E-07 | 4.43E-04 |
| Exam. 12 ($120 \times 1200$, 120) | 5.7E-06 | 2.84E-03 | 1.06E-09 | 4.91E-07 | 7.67E-09 | 6.75E-06 |
| Exam. 13 ($130 \times 1300$, 130) | 3.65E-06 | 2.24E-03 | 6.49E-09 | 2.26E-06 | 8.37E-07 | 9.33E-04 |
| Exam. 14 ($140 \times 1400$, 140) | 7.49E-06 | 4.37E-03 | 7.64E-11 | 1.74E-08 | 153366.2 | 5.67E-03 |
| Exam. 15 ($150 \times 1500$, 150) | 2.19E-05 | 1.25E-02 | 1.59E-08 | 3.02E-06 | 3.8E-08  | 4.28E-05 |
| Exam. 16 ($160 \times 1600$, 160) | 6.53E-06 | 3.84E-03 | 2.21E-09 | 1.52E-06 | 305845.9 | 1.93E-02 |
| Exam. 17 ($170 \times 1700$, 170) | 1.53E-05 | 9.79E-03 | 3.04E-11 | 1.16E-08 | 6.08E-07 | 7.96E-04 |
| Exam. 18 ($180 \times 1800$, 180) | 3.44E-05 | 2.28E-02 | 2.63E-10 | 3.08E-07 | 8.00E-09 | 1.12E-05 |
| Exam. 19 ($190 \times 1900$, 190) | 6.99E-06 | 5.69E-03 | 2.11E-11 | 2.05E-08 | 8.10E-09 | 1.21E-05 |
| Exam. 20 ($200 \times 2000$, 200) | 1.44E-05 | 1.48E-02 | 8.59E-08 | 6.88E-06 | 1.13E-07 | 1.79E-04 |
| Exam. 21 ($210 \times 2100$, 210) | 2.93E-05 | 1.51E-02 | 3.28E-09 | 5.93E-07 | 541460.8 | 2.45E-02 |
| Exam. 22 ($220 \times 2200$, 220) | 1.18E-05 | 1.10E-02 | 2.02E-10 | 2.11E-07 | 1.13E-07 | 1.64E-04 |
| Exam. 23 ($230 \times 2300$, 230) | 9.34E-06 | 8.48E-03 | 1.08E-09 | 7.37E-07 | 274616.9 | 1.48E-02 |
| Exam. 24 ($240 \times 2400$, 240) | 1.27E-05 | 1.36E-02 | 5.64E-07 | 7.68E-05 | 345886.2 | 5.03E-03 |
| Exam. 25 ($250 \times 2500$, 250) | 1.38E-05 | 1.43E-02 | 5.1E-12 | 4.07E-09 | 8.04E-07 | 1.53E-03 |
| Exam. 26 ($260 \times 2600$, 260) | 1.2E-05 | 1.29E-02 | 7.05E-09 | 2.57E-06 | 2.37E-07 | 4.63E-04 |
| Exam. 27 ($270 \times 2700$, 270) | 1.29E-05 | 1.69E-02 | 7.59E-08 | 7.85E-05 | 490836.3 | 2.03E-02 |
| Exam. 28 ($280 \times 2800$, 280) | 1.44E-05 | 2.11E-02 | 5.71E-09 | 6.67E-07 | 241381.4 | 1.29E-02 |
| Exam. 29 ($290 \times 2900$, 290) | 2.69E-05 | 3.11E-02 | 9.80E-12 | 2.83E-09 | 6.47E-07 | 1.30E-03 |
| Exam. 30 ($300 \times 3000$, 300) | 1.18E-05 | 1.59E-02 | 2.56E-08 | 3.21E-05 | 418713  | 9.26E-03 |

In order to verify the effect of the PNMTRLP method handling those problems, we randomly generate the sparse matrix $A$ and let $A(m - i, :) = i * A(m, :) (i = 1, 2, \ldots, m - r)$, where $r$ is the rank of matrix $A$, then we compare it with the subroutine linprog.m for those problems. The dimension of matrix $A$ varies from 10 $\times$ 100 to 300 $\times$ 3000. One of its implementation is illustrated by Algorithm 5.

Since the subroutine pathfollow.m can not effectively handle the problem with the deficient rank of matrix $A$, we only report the numerical results of the PNMTRLP method and the subroutine linprog.m, and put them in Tables 5-8. From Tables 5-8, we can find that the PNMTRLP method can solve all those problems. However,
Algorithm 3 The standard linear programming problem with the deficient rank matrix and the noise right-hand-side vector

**Input:**
- the number of equality constraints: \( m \);
- the number of unknown variables: \( n \);
- the rank of the matrix \( A \): \( r \);
- the maximum noise of the right-hand-side vector \( b \): \( \text{eps} \).

**Output:**
- matrix \( A \) and vectors \( b \in \mathbb{R}^m \) and \( c \in \mathbb{R}^n \).

```matlab
1: density = 0.2;
2: A = sprandn(m,n,density); % Generate a sparse matrix of give density.
3: for \( i = 1, 2, \ldots, r \) do
4: A(m-i, :) = i*A(m, :);
5: end for
6: xfeas = [rand(n/2,1); zeros(n-(n/2),1)];
7: sfeas = [zeros(n/2,1); rand(n-(n/2),1)];
8: xfeas = xfeas(randperm(n));
9: sfeas = sfeas(randperm(n));
10: yfeas = (rand(m,1) - 0.5)*4;
% Choose \( b \) and \( c \) to make this \((x, y, s)\) feasible.
11: b = A*xfeas + 2*(rand(m,1)-0.5)*\text{eps};
12: c = A'*yfeas + sfeas;
```

Table 2: Statistical results of deficient rank problems with the noise data.

| number of failed problems | PNMTRLP | linprog |
|---------------------------|---------|--------|
| 0/180                     | 143/180 |

the subroutine linprog.m can not solve some problems, because those problems do not have feasible solution and the subroutine linprog.m outputs \( NaN \). Furthermore, although the subroutine linprog.m can obtain the solutions for some problems, such as examples A1, A2, A5, A6, etc, the KKT errors and the primal-dual gaps are large, so we conclude that the subroutine linprog.m also fails to solve those problems. Table 2 is the statistical results of failed problems solved by the PNMTRLP method and the subroutine linprog.m. From Table 2, we can find that the subroutine linprog.m can not effectively handle the linear programming problem with the deficient rank of matrix \( A \) and the noise right-hand-side vector \( b \), because the failed problems of the subroutine linprog.m are around 80%.

We also give the trends of the KKT errors and the primal-dual gaps for the different right-hand-side noise from \( 10^{-6} \) to \( 10^{-1} \) in the right sub-figure and the left sub-figure in Figure 2 respectively. From Figure 2, we find that the KKT errors and the primal-dual gaps of the PNMTRLP method are less than 0.12 when the right-hand-side noise comes up to 0.1. Therefore, the PNMTRLP method can be used in practical application scenarios, which will greatly reduce the cost of industrial hardware facilities.
5 Conclusion and future work

For the standard linear programming problem with the deficient rank of matrix $A$ and the noise right-hand-side vector $b$, we give a preprocessing method based on the singular value decomposition method. Then, according to the first order KKT conditions of the linear programming, we convert the treated problems into the equivalent problem of nonlinear equations with nonnegative constraints. Based on the system of nonlinear equations with nonnegative constraints, we consider a special continuous Newton flow with nonnegative constraints, which has the nonnegative steady-state solution for any nonnegative initial point. In order to improve the robust and efficiency of tracing the trajectory, we construct a perturbed Newton method with the new time-stepping scheme based on the trust-region technique (the PNMTRLP method, Algorithm 1). We also prove the global convergence of the PNMTRLP method when the initial point is strictly primal-dual feasible. Finally, we test some standard linear programming problems with full rank matrices or deficient rank matrices, and compare it with the path-following method (the subroutine pathfollow.m [14]) and the state-of-the-art Mehrotra’s predictor-corrector algorithm (the built-in subroutine linprog.m of the MATLAB environment [26, 27, 36]). The numerical results show that the PNMTRLP method performs well and the subroutines pathfollow.m and linprog.m can not effectively handle the problem with the deficient rank of matrix $A$ and the noise right-hand-side vector $b$. Therefore, the PNMTRLP method is worth investigating further as a new path-following method.

Acknowledgments

This work was supported in part by Grant 61876199 from National Natural Science Foundation of China, Grant YBWL2011085 from Huawei Technologies Co., Ltd.,
and Grant YJCB2011003HI from the Innovation Research Program of Huawei Technologies Co., Ltd..

References

1. E. L. Allgower and K. Georg, *Introduction to Numerical Continuation Methods*, SIAM, Philadelphia, PA, 2003.
2. E. D. Anderson and K. D. Anderson, *Presolving in linear programming*, Mathematical Programming, 71: 221-245, 1995.
3. U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia, PA, 1998.
4. O. Axelsson and S. Sysala, *Continuation Newton methods*, Computer and Mathematics with Applications, 70: 2621-2637, 2015.
5. F. H. Branin, *Widely convergent method for finding multiple solutions of simultaneous nonlinear equations*, IBM Journal of Research and Development, 16: 504-521 (1972).
6. K. E. Brenan, S. L. Campbell and L. R. Petzold, *Numerical solution of initial-value problems in differential-algebraic equations*, SIAM, Philadelphia, PA, 1996.
7. A. R. Conn, N. Gould and Ph. L. Toint, *Trust-Region Methods*, SIAM, 2000.
8. B. D. Craven and S. M. N. Islam, *Linear programming with Uncertain data: extensions to robust optimization*, Journal of Optimization Theory and Applications, 155: 673-679, 2012.
9. D. F. Davidenko, *On a new method of numerical solution of systems of nonlinear equations* (in Russian), Dokl. Akad. Nauk SSSR, 88: 601-602, 1953.
10. P. Deuflhard, *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*, Springer-Verlag, 2004.
11. P. Deuflhard, H. J. Pesch and P. Rentrop, *A modified continuation method for the numerical solution of nonlinear two-point boundary value problems by shooting techniques*, Numerische Mathematik, 26: 327-343, 1975.
12. J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Science Press, 2009.
13. E. J. Doedel, *Lecture notes in numerical analysis of nonlinear equations*, in B. Krauskopf, H. M. Osinga and J. Galán-Vioque (Eds.), *Numerical Continuation Methods for Dynamical Systems*, 1-50, Springer, 2007.
14. M. C. Ferris, O. L. Mangasarian and S. J. Wright, *Linear Programming with MATLAB*, SIAM, Philadelphia, PA, http://pages.cs.wisc.edu/~ferris/ferris.books.html, 2007.
15. A. V. Fracco and G. P. McCormick, *Nonlinear programming: Sequential Unconstrained Minimization Techniques*, SIAM, 1990.
16. C. C. Gonzaga, *Path-following methods for linear programming*, SIAM Review, 34 (2): 167-224, 1992.
17. G. H. Golub and C. F. Van Loan, *Matrix Computation*, fourth edition, The John Hopkins University Press, 2013.
18. E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*, 2nd ed., Springer-Verlag, Berlin, 1996.
19. D. J. Higham, *Trust region algorithms and timestep selection*, SIAM Journal of Numerical Analysis, 37: 194-210 (1999).
20. N. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica, (4): 373-395 (1984).
21. X.-L. Luo, *A second-order pseudo-transient method for steady-state problems*, Applied Mathematics and Computation, 216: 1752-1762 (2010).
22. X.-L. Luo, *A dynamical method of DAEs for the smallest eigenvalue problem*, Journal of Computational Science, 3: 113-119 (2012).
23. X.-L. Luo, J.-H. Lv and G. Sun, *Continuation method with the trust-region time-stepping scheme for linearly constrained optimization with noise data*, http://arxiv.org/abs/2005.05965 May 2020.
24. X.-L. Luo, H. Xiao and J.-H. Lv, *Continuous Newton method with the trust-region time-stepping scheme*, https://arxiv.org/abs/2006.02634 June 2020.
25. J. M. Ortega and W. C. Rheinboldt, *Iteration Solution of Nonlinear Equations in Several Variables*, Classics in Applied Mathematics, SIAM, 2000.
26. MATLAB 9.6.0 (R2019a), The MathWorks Inc., [http://www.mathworks.com](http://www.mathworks.com), 2019.
27. S. Mehrotra, *On the implementation of a primal-dual interior point method*, SIAM Journal on Optimization, 2: 575-601 (1992).
28. J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, 1999.
29. M. J. D. Powell, *Convergence properties of a class of minimization algorithms*, in: O.L. Mangasarian, R. R. Meyer and S. M. Robinson, eds., *Nonlinear Programming 2*, Academic Press, New York, 1975, pp. 1-27.
30. L. S. Pontryagin, *The Ordinary differential equations*, Higher Education Press (in Chinese Translation from Russian), 2000.
31. L. F. Shampine, I. Gladwell and S. Thompson, *Solving ODEs with MATLAB*, Cambridge University Press, 2003.
32. K. Tanabe, *Continuous Newton-Raphson method for solving an undetermined system of nonlinear equations*, Nonlinear Analysis, 3: 495-503, 1979.
33. K. Tanabe, *Centered Newton method for mathematical programming*, in System Modeling and Optimization: Proceedings of the 13th IFIP conference, vol. 113 of Lecture Notes in Control and Information Systems, Berlin, Springer-Verlag, pp. 197-206, 1988.
34. S. J. Wright, *Primal-dual Interior Point Methods*, SIAM, Philadelphia, 1997.
35. Z.-D. Yuan, J.-G. Fei and D.-G. Liu, *The Numerical Solutions for Initial Value Problems of Stiff Ordinary Differential Equations* (in Chinese), Science Press, Beijing, 1987.
36. Y. Zhang, *Solving large-Scale linear programs by interior-point methods under the MATLAB environment*, Optimization Methods and Software, 10 (1): 1-31, 1998. [https://doi.org/10.1080/10556789808805699](https://doi.org/10.1080/10556789808805699)
## A Numerical results of test problems

| Problem $(m \times n, r)$ | PNMTLP CPU (s) | KKTError | Gap | linprog CPU (s) | KKTError | Gap |
|---------------------------|----------------|----------|-----|----------------|----------|-----|
| Exam. A1 $(10 \times 100, 4)$ | 0.0781 | 0.0793 | 1.14E-05 | 0.1406 | 15.0250 | 463.69 |
| Exam. A2 $(20 \times 200, 17)$ | 0.063 | 0.0820 | 4.78E-05 | 0.1562 | 168.7704 | 17754.71 |
| Exam. A3 $(30 \times 300, 23)$ | 0.072 | 0.0932 | 0.0006 | 0.1094 | Failed | Failed |
| Exam. A4 $(40 \times 400, 31)$ | 0.0625 | 0.0911 | 0.0002 | 0.0156 | Failed | Failed |
| Exam. A5 $(50 \times 500, 48)$ | 0.0625 | 0.0448 | 0.0005 | 0.1250 | 0.9009 | 159.74 |
| Exam. A6 $(60 \times 600, 51)$ | 0.2031 | 0.0926 | 0.0008 | 0.1094 | 105.0479 | 24875.35 |
| Exam. A7 $(70 \times 700, 68)$ | 0.1718 | 0.0879 | 0.0032 | 0.1406 | 0.1266 | 24.59 |
| Exam. A8 $(80 \times 800, 79)$ | 0.2343 | 0.0032 | 0.0011 | 0.1250 | 0.01242 | 1.11 |
| Exam. A9 $(90 \times 900, 89)$ | 0.2031 | 0.1067 | 0.0020 | 0.0781 | 125.0551 | 50816.85 |
| Exam. A10 $(100 \times 1000, 97)$ | 0.2812 | 0.0376 | 0.0020 | 0.1562 | 0.60036 | 278.61 |
| Exam. A11 $(110 \times 1100, 109)$ | 0.3125 | 0.1054 | 0.0039 | 0.1250 | 174.8302 | 86218.32 |
| Exam. A12 $(120 \times 1200, 118)$ | 0.4375 | 0.0758 | 0.0022 | 0.1250 | 146.0728 | 113072.09 |
| Exam. A13 $(130 \times 1300, 127)$ | 0.7187 | 0.0973 | 0.0028 | 0.1719 | 148.7301 | 80494.88 |
| Exam. A14 $(140 \times 1400, 134)$ | 0.5468 | 0.0830 | 0.0049 | 0.2344 | 146.0728 | 85987.87 |
| Exam. A15 $(150 \times 1500, 147)$ | 0.7187 | 0.0365 | 0.0051 | 0.1562 | Failed | Failed |
| Exam. A16 $(160 \times 1600, 159)$ | 0.6718 | 0.0884 | 0.0084 | 0.1719 | Failed | Failed |
| Exam. A17 $(170 \times 1700, 162)$ | 0.6250 | 0.0725 | 0.0116 | 0.2344 | 158.9470 | 152967.17 |
| Exam. A18 $(180 \times 1800, 179)$ | 0.5000 | 0.0881 | 0.0054 | 0.2969 | Failed | Failed |
| Exam. A19 $(190 \times 1900, 187)$ | 0.7343 | 0.0541 | 0.0140 | 0.2500 | Failed | Failed |
| Exam. A20 $(200 \times 2000, 197)$ | 0.7343 | 0.1011 | 0.0062 | 0.7500 | 9040.9040 | 1027.19 |
| Exam. A21 $(210 \times 2100, 209)$ | 1.0625 | 0.1123 | 0.0166 | 0.2656 | 163.2310 | 157681.89 |
| Exam. A22 $(220 \times 2200, 211)$ | 0.7968 | 0.0801 | 0.0113 | 1.2187 | Failed | Failed |
| Exam. A23 $(230 \times 2300, 227)$ | 0.9218 | 0.0938 | 0.0065 | 0.3125 | 218.9483 | 230272.22 |
| Exam. A24 $(240 \times 2400, 235)$ | 1.1250 | 0.0921 | 0.0114 | 0.4844 | 185.2789 | 187566.15 |
| Exam. A25 $(250 \times 2500, 247)$ | 1.2344 | 0.0575 | 0.0159 | 0.3906 | 177.0520 | 201144.94 |
| Exam. A26 $(260 \times 2600, 259)$ | 1.5781 | 0.0848 | 0.0118 | 1.6875 | Failed | Failed |
| Exam. A27 $(270 \times 2700, 262)$ | 1.7968 | 0.0928 | 0.0214 | 0.5000 | Failed | Failed |
| Exam. A28 $(280 \times 2800, 274)$ | 2.4062 | 0.0781 | 0.0191 | 0.7187 | 172.7984 | 201795.88 |
| Exam. A29 $(290 \times 2900, 284)$ | 1.6250 | 0.1012 | 0.0206 | 0.7812 | Failed | Failed |

Table 3: Numerical results of deficient rank problems with noise $\epsilon_b = 10^{-1}$. 
Table 4: Numerical results of deficient rank problems with noise $\epsilon_b = 10^{-2}$.

| Problem $(m \times n, r)$ | PNMTRLP | Improg |
|-------------------------|----------|--------|
|                         | CPU (s)  | KKTError | Gap | CPU (s)  | KKTError | Gap |
| Exam. B1 (10 × 100, 5)  | 0.0469   | 0.0078   | 0.0001 | 0.2344 | 48.9117 | 1771.6522 |
| Exam. B2 (20 × 200, 19) | 0.0521   | 0.0069   | 0.0002 | 0.0781 | 0.0138 | 0.5149 |
| Exam. B3 (30 × 300, 21) | 0.0625   | 0.0089   | 0.0003 | 0.1094 | 0.1094 | 0.3652 |
| Exam. B4 (40 × 400, 35) | 0.0625   | 0.0090   | 0.0003 | 0.1094 | 0.0322 | 74.6274 |
| Exam. B5 (50 × 500, 47) | 0.0781   | 0.0099   | 0.0006 | 0.0469 | 0.2956 | 0.0313 |
| Exam. B6 (60 × 600, 54) | 0.0156   | 0.0066   | 0.0005 | Failed | Failed |
| Exam. B7 (70 × 700, 66) | 0.1875   | 0.0099   | 0.0006 | Failed | Failed |
| Exam. B8 (80 × 800, 76) | 0.4219   | 0.0042   | 0.0012 | 0.1719 | 0.062 | 0.2883 |
| Exam. B9 (90 × 900, 89) | 0.2813   | 0.0048   | 0.0021 | 0.1094 | Failed | Failed |
| Exam. B10 (100 × 1000, 98) | 0.3594 | 0.0044 | 0.0020 | 0.3594 | 0.0157 | 0.5814 |
| Exam. B11 (110 × 1100, 107) | 0.4531 | 0.0093 | 0.0023 | 0.1719 | Failed | Failed |
| Exam. B12 (120 × 1200, 118) | 0.4063 | 0.0021 | 0.0017 | 0.4531 | 0.107 | 2.5241 |
| Exam. B13 (130 × 1300, 129) | 0.5156 | 0.0083 | 0.0033 | 0.1719 | Failed | Failed |
| Exam. B14 (140 × 1400, 137) | 0.5156 | 0.0093 | 0.0059 | 0.3594 | 0.0298 | 11.7993 |
| Exam. B15 (150 × 1500, 143) | 0.7188 | 0.0123 | 0.0091 | 0.2969 | Failed | Failed |
| Exam. B16 (160 × 1600, 154) | 0.5313 | 0.0038 | 0.0243 | 0.3125 | Failed | Failed |
| Exam. B17 (170 × 1700, 167) | 0.4844 | 0.0073 | 0.0086 | 0.2500 | Failed | Failed |
| Exam. B18 (180 × 1800, 175) | 0.7969 | 0.0100 | 0.0049 | 0.4375 | Failed | Failed |
| Exam. B19 (190 × 1900, 187) | 0.6719 | 0.0095 | 0.0074 | 0.5000 | Failed | Failed |
| Exam. B20 (200 × 2000, 195) | 0.6563 | 0.0088 | 0.0073 | 0.5625 | Failed | Failed |
| Exam. B21 (210 × 2100, 205) | 0.7656 | 0.0077 | 0.0204 | 0.3438 | Failed | Failed |
| Exam. B22 (220 × 2200, 216) | 0.7813 | 0.0065 | 0.0076 | 0.8594 | Failed | Failed |
| Exam. B23 (230 × 2300, 226) | 0.9063 | 0.0103 | 0.0124 | 0.3906 | Failed | Failed |
| Exam. B24 (240 × 2400, 235) | 1.1719 | 0.0098 | 0.0163 | 0.6719 | Failed | Failed |
| Exam. B25 (250 × 2500, 245) | 1.2344 | 0.0079 | 0.0101 | 1.5313 | Failed | Failed |
| Exam. B26 (260 × 2600, 258) | 2.2344 | 0.0083 | 0.0125 | 1.1563 | Failed | Failed |
| Exam. B27 (270 × 2700, 263) | 2.1250 | 0.0095 | 0.0178 | 1.1563 | Failed | Failed |
| Exam. B28 (280 × 2800, 271) | 1.5781 | 0.0083 | 0.0132 | 0.6719 | Failed | Failed |
| Exam. B29 (290 × 2900, 282) | 1.5938 | 0.0104 | 0.0324 | 0.9375 | Failed | Failed |
| Exam. B30 (300 × 3000, 291) | 1.5938 | 0.0104 | 0.0324 | 0.9375 | Failed | Failed |
Table 5: Numerical results of deficient rank problems with noise $\epsilon_b = 10^{-3}$.

| Problem ($m \times n, r$) | PNMTLP | limprog |
|---------------------------|---------|---------|
|                           | CPU (s) | KKTError | Gap | CPU (s) | KKTError | Gap |
| Exam. C1 (10 × 100, 9)    | 0.0938 | 0.0005   | 0.0001 | 0.2031 | 22.7266 | 887.6482 |
| Exam. C2 (20 × 200, 16)  | 0.0156 | 0.0008   | 0.0001 | 0.0156 | Failed  | Failed   |
| Exam. C3 (30 × 300, 25)  | 0.0938 | 0.0007   | 0.0001 | 0.1406 | Failed  | Failed   |
| Exam. C4 (40 × 400, 36)  | 0.0851 | 0.0010   | 0.0003 | 0.0781 | Failed  | Failed   |
| Exam. C5 (50 × 500, 49)  | 0.0781 | 0.0007   | 0.0007 | 0.1094 | 0.0013  | 0.0005   |
| Exam. C6 (60 × 600, 59)  | 0.1406 | 0.0001   | 0.0008 | 0.0938 | 0.0001  | 0.0028   |
| Exam. C7 (70 × 700, 66)  | 0.0469 | 0.0008   | 0.0004 | 0.1250 | Failed  | Failed   |
| Exam. C8 (80 × 800, 76)  | 0.1250 | 0.0006   | 0.0028 | 0.0469 | Failed  | Failed   |
| Exam. C9 (90 × 900, 84)  | 0.4375 | 0.0007   | 0.0018 | 0.0469 | Failed  | Failed   |
| Exam. C10 (100 × 1000, 93)| 0.2188 | 0.0008   | 0.0014 | 0.0938 | Failed  | Failed   |
| Exam. C11 (110 × 1100, 103)| 0.2813 | 0.0011   | 0.0042 | 0.0625 | Failed  | Failed   |
| Exam. C12 (120 × 1200, 111)| 0.2188 | 0.0007   | 0.0019 | 0.1094 | Failed  | Failed   |
| Exam. C13 (130 × 1300, 128)| 0.5625 | 0.0001   | 0.0035 | 0.2031 | Failed  | Failed   |
| Exam. C14 (140 × 1400, 131)| 0.3750 | 0.0007   | 0.0029 | 0.2188 | Failed  | Failed   |
| Exam. C15 (150 × 1500, 148)| 0.5469 | 0.0009   | 0.0033 | 0.5156 | Failed  | Failed   |
| Exam. C16 (160 × 1600, 157)| 0.7656 | 0.0008   | 0.0034 | 0.3594 | Failed  | Failed   |
| Exam. C17 (170 × 1700, 167)| 0.4375 | 0.0005   | 0.0024 | 0.3281 | Failed  | Failed   |
| Exam. C18 (180 × 1800, 176)| 0.4688 | 0.0011   | 0.0059 | 0.4688 | Failed  | Failed   |
| Exam. C19 (190 × 1900, 189)| 0.5938 | 0.0002   | 0.0030 | 0.2031 | 142.2067 | 132145  |
| Exam. C20 (200 × 2000, 195)| 0.7969 | 0.0007   | 0.0071 | 1.8750 | 0.0277  | 0.3313   |
| Exam. C21 (210 × 2100, 208)| 0.7813 | 0.0005   | 0.0022 | 0.5156 | Failed  | Failed   |
| Exam. C22 (220 × 2200, 218)| 1.1719 | 0.0004   | 0.0086 | 1.1094 | 0.0010  | 0.0000   |
| Exam. C23 (230 × 2300, 226)| 0.7500 | 0.0008   | 0.0121 | 0.6094 | Failed  | Failed   |
| Exam. C24 (240 × 2400, 239)| 0.9375 | 0.0006   | 0.0115 | 0.5469 | 0.0001  | 0.1087   |
| Exam. C25 (250 × 2500, 246)| 1.1094 | 0.0010   | 0.0154 | 2.0625 | Failed  | Failed   |
| Exam. C26 (260 × 2600, 258)| 1.5156 | 0.0010   | 0.0185 | 0.9531 | Failed  | Failed   |
| Exam. C27 (270 × 2700, 263)| 1.5469 | 0.0008   | 0.0149 | 1.9844 | 0.0089  | 0.1173   |
| Exam. C28 (280 × 2800, 277)| 1.8906 | 0.0007   | 0.0108 | 3.8281 | 0.0016  | 0.0045   |
| Exam. C29 (290 × 2900, 283)| 1.5469 | 0.0007   | 0.0148 | 1.2656 | Failed  | Failed   |
| Exam. C30 (300 × 3000, 291)| 1.7969 | 0.0004   | 0.0313 | 1.1719 | Failed  | Failed   |
| Problem \((m \times n, r)\) | PNMTRLP | linprog |
|------------------------|----------|---------|
|                       | CPU (s)  | KKTError | Gap | CPU (s)  | KKTError | Gap |
| Exam. D1 \((10 \times 100, 6)\) | 0.0469   | 1.03E-4  | 0.0001 | 0.2500   | 0.0003  | 1.01E-4 |
| Exam. D2 \((20 \times 200, 15)\) | 0.0625   | 3.6E-5   | 0.0001 | 0.1094   | Failed   | Failed |
| Exam. D3 \((30 \times 300, 27)\) | 0.1250   | 8.30E-5  | 0.0001 | 0.2656   | Failed   | Failed |
| Exam. D4 \((40 \times 400, 38)\) | 0.1563   | 9.35E-5  | 0.0004 | 0.1094   | 0.0001  | 0.0005 |
| Exam. D5 \((50 \times 500, 49)\) | 0.0313   | 5.01E-5  | 0.0004 | 0.0938   | 0.3173  | 53.3101 |
| Exam. D6 \((60 \times 600, 53)\) | 0.3281   | 8.96E-5  | 0.0008 | 0.0625   | Failed   | Failed |
| Exam. D7 \((70 \times 700, 69)\) | 0.2813   | 2.92E-5  | 0.0009 | 0.0625   | 128.9256 | 44395.4123 |
| Exam. D8 \((80 \times 800, 79)\) | 0.2813   | 6.33E-5  | 0.0020 | 0.1406   | 112.1089 | 40834.5967 |
| Exam. D9 \((90 \times 900, 82)\) | 0.0781   | 9.58E-5  | 0.0021 | 0.1094   | Failed   | Failed |
| Exam. D10 \((100 \times 1000, 95)\) | 0.1250   | 6.91E-5  | 0.0032 | 0.2656   | Failed   | Failed |
| Exam. D11 \((110 \times 1100, 108)\) | 0.2031   | 2.85E-5  | 0.0021 | 0.1250   | Failed   | Failed |
| Exam. D12 \((120 \times 1200, 118)\) | 0.3594   | 2.40E-5  | 0.0015 | 0.1563   | Failed   | Failed |
| Exam. D13 \((130 \times 1300, 129)\) | 0.3125   | 2.73E-5  | 0.0033 | 0.0938   | 156.8888 | 85946.2160 |
| Exam. D14 \((140 \times 1400, 131)\) | 0.7500   | 9.07E-5  | 0.0024 | 0.2500   | 0.0003  | 0.0001 |
| Exam. D15 \((150 \times 1500, 145)\) | 0.6406   | 8.05E-5  | 0.0085 | 0.4531   | Failed   | Failed |
| Exam. D16 \((160 \times 1600, 155)\) | 0.8594   | 8.11E-5  | 0.0063 | 0.1719   | Failed   | Failed |
| Exam. D17 \((170 \times 1700, 167)\) | 0.4688   | 6.79E-5  | 0.0054 | 0.5938   | Failed   | Failed |
| Exam. D18 \((180 \times 1800, 179)\) | 0.4844   | 1.09E-5  | 0.0040 | 0.1563   | 165.8385 | 132269.5382 |
| Exam. D19 \((190 \times 1900, 188)\) | 0.5469   | 2.20E-5  | 0.0050 | 0.9531   | 0.0001  | 0.0001 |
| Exam. D20 \((200 \times 2000, 197)\) | 0.7031   | 2.88E-5  | 0.0058 | 0.1406   | Failed   | Failed |
| Exam. D21 \((210 \times 2100, 202)\) | 0.7031   | 7.44E-5  | 0.0129 | 0.3906   | Failed   | Failed |
| Exam. D22 \((220 \times 2200, 217)\) | 0.6563   | 7.08E-5  | 0.0155 | 0.4688   | Failed   | Failed |
| Exam. D23 \((230 \times 2300, 222)\) | 0.8281   | 9.96E-5  | 0.0409 | 0.3125   | Failed   | Failed |
| Exam. D24 \((240 \times 2400, 239)\) | 0.9531   | 4.31E-5  | 0.0061 | 0.4375   | 188.7456 | 219609.81 |
| Exam. D25 \((250 \times 2500, 242)\) | 1.3125   | 8.70E-5  | 0.0153 | 1.2344   | Failed   | Failed |
| Exam. D26 \((260 \times 2600, 259)\) | 1.3906   | 1.46E-5  | 0.0145 | 0.4063   | 158.8031 | 20129.82 |
| Exam. D27 \((270 \times 2700, 268)\) | 1.5313   | 1.48E-5  | 0.0099 | 1.8281   | 0.0000  | 0.0012 |
| Exam. D28 \((280 \times 2800, 278)\) | 1.8594   | 7.47E-5  | 0.0127 | 0.5781   | Failed   | Failed |
| Exam. D29 \((290 \times 2900, 287)\) | 1.4219   | 7.96E-5  | 0.0144 | 2.5938   | Failed   | Failed |
| Exam. D30 \((300 \times 3000, 295)\) | 2.8125   | 7.56E-5  | 0.0269 | 2.0000   | Failed   | Failed |

Table 6: Numerical results of deficient rank problems with noise \(\varepsilon_b = 10^{-4}\).
Table 7: Numerical results of deficient rank problems with noise $\varepsilon_b = 10^{-5}$.

| Problem $(m \times n, r)$ | CPU (s) | KKTError | Gap | CPU (s) | KKTError | Gap |
|---------------------------|---------|----------|-----|---------|----------|-----|
| Exam. E1 $(10 \times 100, 5)$ | 0.1250  | 9.63E-06 | 2.27E-05 | 0.2188 | Failed | Failed |
| Exam. E2 $(20 \times 200, 19)$ | 0.0781  | 3.62E-06 | 3.26E-05 | 0.1719 | 62.0437 | 5499.26 |
| Exam. E3 $(30 \times 300, 28)$ | 0.1719  | 7.06E-06 | 0.000299 | 0.1250 | Failed | Failed |
| Exam. E4 $(40 \times 400, 35)$ | 0.0625  | 8.93E-06 | 0.000166 | 0.0625 | Failed | Failed |
| Exam. E5 $(50 \times 500, 42)$ | 0.1406  | 9.79E-06 | 0.000243 | 0.1094 | 0.0008 | 0.0020 |
| Exam. E6 $(60 \times 600, 58)$ | 0.2344  | 1.01E-05 | 0.000554 | 0.1250 | Failed | Failed |
| Exam. E7 $(70 \times 700, 69)$ | 0.1875  | 4.52E-06 | 0.000787 | 0.1406 | 5.8E-06 | 4.7E-08 |
| Exam. E8 $(80 \times 800, 78)$ | 0.2656  | 1.07E-05 | 0.001612 | 0.0469 | Failed | Failed |
| Exam. E9 $(90 \times 900, 82)$ | 0.2813  | 5.48E-06 | 0.00198 | 0.2188 | 9.97E-06 | 0.0002 |
| Exam. E10 $(100 \times 1000, 95)$ | 0.2188  | 9.59E-06 | 0.001049 | 0.1094 | Failed | Failed |
| Exam. E11 $(110 \times 1100, 105)$ | 0.3125  | 9.48E-06 | 0.001523 | 0.2969 | Failed | Failed |
| Exam. E12 $(120 \times 1200, 119)$ | 0.2188  | 1.49E-05 | 0.006811 | 0.9063 | 1.2E-05 | 1.2E-07 |
| Exam. E13 $(130 \times 1300, 127)$ | 0.3750  | 8.33E-06 | 0.000297 | 0.2188 | Failed | Failed |
| Exam. E14 $(140 \times 1400, 137)$ | 0.3281  | 8.33E-06 | 0.005197 | 0.4688 | Failed | Failed |
| Exam. E15 $(150 \times 1500, 147)$ | 0.5469  | 9.77E-06 | 0.006226 | 1.6094 | Failed | Failed |
| Exam. E16 $(160 \times 1600, 152)$ | 0.8438  | 1.18E-05 | 0.008784 | 0.5625 | 8.1E-06 | 6.4E-18 |
| Exam. E17 $(170 \times 1700, 167)$ | 0.5156  | 7.67E-06 | 0.003721 | 0.6875 | Failed | Failed |
| Exam. E18 $(180 \times 1800, 175)$ | 0.6094  | 4.82E-06 | 0.002951 | 0.8594 | Failed | Failed |
| Exam. E19 $(190 \times 1900, 181)$ | 0.4688  | 9.23E-06 | 0.004763 | 0.6406 | 2.2E-05 | 3.0E-05 |
| Exam. E20 $(200 \times 2000, 191)$ | 0.5625  | 9.53E-06 | 0.00634 | 1.0156 | 1.99E-05 | 8.9E-05 |
| Exam. E21 $(210 \times 2100, 201)$ | 0.7344  | 2.36E-05 | 0.022886 | 1.4844 | 3.42E-05 | 4.2E-18 |
| Exam. E22 $(220 \times 2200, 216)$ | 0.8281  | 1.34E-05 | 0.015473 | 1.2813 | Failed | Failed |
| Exam. E23 $(230 \times 2300, 222)$ | 0.2821  | 1.43E-05 | 0.012337 | 0.8594 | Failed | Failed |
| Exam. E24 $(240 \times 2400, 232)$ | 0.8281  | 1.67E-05 | 0.015467 | 1.4219 | Failed | Failed |
| Exam. E25 $(250 \times 2500, 247)$ | 1.0000  | 9.68E-06 | 0.010106 | 0.4688 | Failed | Failed |
| Exam. E26 $(260 \times 2600, 256)$ | 1.3281  | 8.56E-06 | 0.009204 | 0.5469 | Failed | Failed |
| Exam. E27 $(270 \times 2700, 264)$ | 1.4688  | 1.13E-05 | 0.011883 | 1.7969 | 1.13E-05 | 2.5E-19 |
| Exam. E28 $(280 \times 2800, 276)$ | 1.8125  | 1.56E-05 | 0.017586 | 6.0625 | 0.0212 | 3.2E-22 |
| Exam. E29 $(290 \times 2900, 285)$ | 1.4844  | 3.82E-05 | 0.044008 | 2.1250 | Failed | Failed |
| Exam. E30 $(300 \times 3000, 292)$ | 1.6094  | 1.07E-05 | 0.011986 | 4.8594 | 2.25E-05 | 4.1E-06 |
Table 8: Numerical results of deficient rank problems with noise $\epsilon_b = 10^{-6}$.

| Problem ($m \times n$, $r$) | CPU (s) | PNMTLP | KKTError | Gap | CPU (s) | linprog | KKTError | Gap |
|----------------------------|---------|---------|----------|-----|---------|---------|----------|-----|
| Exam. F1 (10 × 100, 2)    | 0.1094  | 9.1042E-07 | 1.95018E-05 | 0.1406 | 1.93E+21 | 701878381 |
| Exam. F2 (20 × 200, 16)   | 0.0469  | 2.4042E-06 | 0.000196712 | 0.2500 | Failed  | Failed  |
| Exam. F3 (30 × 300, 22)   | 0.0469  | 1.2484E-06 | 0.000182235 | 0.2969 | 7.96E-06 | 1.5039E-26 |
| Exam. F4 (40 × 400, 34)   | 0.1094  | 3.2483E-06 | 0.000508886 | 0.0781 | Failed  | Failed  |
| Exam. F5 (50 × 500, 48)   | 0.0625  | 1.1277E-06 | 0.000227713 | 0.0938 | Failed  | Failed  |
| Exam. F6 (60 × 600, 58)   | 0.2031  | 2.8283E-06 | 0.000773048 | 0.1563 | 8.89E-07 | 6.8731E-25 |
| Exam. F7 (70 × 700, 67)   | 0.2344  | 1.5834E-05 | 0.005080376 | 0.3594 | Failed  | Failed  |
| Exam. F8 (80 × 800, 79)   | 0.1875  | 3.4235E-06 | 0.001094181 | 0.1563 | 1.65E-06 | 5.9364E-08 |
| Exam. F9 (90 × 900, 82)   | 0.1250  | 3.9215E-06 | 0.001544717 | 0.1094 | 3.19E-06 | 6.4124E-24 |
| Exam. F10 (100 × 1000, 91)| 0.2188  | 1.0383E-05 | 0.004762294 | 0.1250 | Failed  | Failed  |
| Exam. F11 (110 × 1100, 109)| 0.3906  | 2.5373E-06 | 0.001204198 | 0.5938 | 2.74E-07 | 4.5308E-26 |
| Exam. F12 (120 × 1200, 116)| 0.4375  | 2.9454E-06 | 0.001782807 | 0.4375 | 2.76E-06 | 2.633E-23 |
| Exam. F13 (130 × 1300, 125)| 0.3750  | 1.1947E-05 | 0.005760924 | 1.4219 | Failed  | Failed  |
| Exam. F14 (140 × 1400, 138)| 0.3750  | 5.0445E-06 | 0.001733344 | 0.5156 | 5.11E-07 | 9.3129E-25 |
| Exam. F15 (150 × 1500, 144)| 0.4063  | 6.0814E-06 | 0.004527647 | 0.3281 | Failed  | Failed  |
| Exam. F16 (160 × 1600, 158)| 0.7969  | 8.3531E-06 | 0.005641379 | 0.1406 | Failed  | Failed  |
| Exam. F17 (170 × 1700, 168)| 0.4688  | 9.0894E-06 | 0.006639951 | 0.2344 | Failed  | Failed  |
| Exam. F18 (180 × 1800, 178)| 0.4531  | 6.8831E-06 | 0.005034438 | 0.7344 | Failed  | Failed  |
| Exam. F19 (190 × 1900, 189)| 0.4531  | 6.4803E-06 | 0.005940951 | 8.9844 | 416.7861 | 2.9417E-18 |
| Exam. F20 (200 × 2000, 195)| 0.5469  | 3.7399E-05 | 0.032659473 | 1.0938 | 9.84E-07 | 1.062E-22 |
| Exam. F21 (210 × 2100, 208)| 0.7500  | 7.5333E-06 | 0.00720211 | 1.0156 | 1.01E-06 | 1.2698E-05 |
| Exam. F22 (220 × 2200, 216)| 0.7969  | 7.8649E-06 | 0.007460031 | 1.1719 | 1.44E-06 | 3.2781E-05 |
| Exam. F23 (230 × 2300, 224)| 0.7344  | 7.4812E-06 | 0.007419235 | 0.5625 | Failed  | Failed  |
| Exam. F24 (240 × 2400, 232)| 0.8281  | 1.0365E-05 | 0.008959213 | 1.6094 | 2.38E-06 | 1.8293E-21 |
| Exam. F25 (250 × 2500, 243)| 1.1250  | 3.3161E-05 | 0.036442403 | 2.8906 | Failed  | Failed  |
| Exam. F26 (260 × 2600, 254)| 1.6563  | 9.8343E-06 | 0.012084121 | 3.0938 | 1.34E-06 | 1.001E-24 |
| Exam. F27 (270 × 2700, 266)| 1.4844  | 6.7967E-06 | 0.008387496 | 2.4219 | 1.1E-06 | 9.5878E-23 |
| Exam. F28 (280 × 2800, 278)| 2.0625  | 1.2999E-05 | 0.01323755 | 1.9844 | 2.55E-06 | 1.7473E-21 |
| Exam. F29 (290 × 2900, 286)| 1.4063  | 1.1847E-05 | 0.014796346 | 1.5156 | Failed  | Failed  |
| Exam. F30 (300 × 3000, 297)| 1.4688  | 1.5102E-05 | 0.017236736 | 1.1563 | Failed  | Failed  |