Journal of Applied Analysis and Computation
Volume *, Number *, * *, 1–15
Website:http://www.jaac-online.com
DOI:10.11948/***

ITERATIVE METHOD FOR RANK-DEFICIENT (2,1)-BLOCK KKT SYSTEMS

Gül Karaduman[1,2,†]

Abstract Karush-Kuhn-Tucker (KKT) systems are widely used in scientific and engineering problems. They are characterized by a 2-by-2 block structure coefficient matrix that exhibits various features, such as sparsity, symmetry, non-symmetry, singularity, and nonsingularity. Solving singular (2,1)-block KKT systems presents significant computational challenges due to their inherent complexity. To address this challenge, an iterative method has been introduced explicitly for addressing singular (2,1)-block KKT systems. The proposed method reduces the system size by using only maximum linearly independent rows of (2,1) block matrices and effectively handles the singularity by transforming the row deficiency problem into a reduced form. The effectiveness and efficiency of the proposed iterative method are verified using numerical experiments with various matrices. It has demonstrated the capacity to provide definitive solutions for singular KKT systems and improve computing performance across multiple applications.

Keywords Rank deficient, Iterative solver, KKT systems, Numerical optimization, Krylov subspace solvers

MSC(2010) 65F10

1. Introduction

The Karush-Kuhn-Tucker (KKT) system is a linear system that appears in many scientific and engineering applications [14]. This system is characterized by a large and sparse coefficient matrix with a 2 by 2 block structure. The system can be written as $\mathcal{K}z = b$. In this system, $\mathcal{K} \in \mathbb{R}^{(n+m)\times(n+m)}$ refers to a large and sparse matrix, the vector $b \in \mathbb{R}^{n+m}$ represents the right-hand vector containing nonzero values, and $z \in \mathbb{R}^{n+m}$ represents the solution vector. KKT can be expressed as,

$$\mathcal{K}z \equiv \begin{bmatrix} A & {B_1}^T \\ B_2 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \equiv b. \tag{1.1}$$

In the system, $A \in \mathbb{R}^{n\times n}$, $B_1 \in \mathbb{R}^{m\times n}$, and $B_2 \in \mathbb{R}^{m\times n}$ are large and sparse matrices. B_2 is a low-rank matrix, and the condition $n \geq m$ holds. The vectors

†The corresponding author.
[1]Karamanoglu Mehmetbey University, Department of Mathematics, 70100, Karaman, Turkey
[2]University of Texas at Arlington, Department of Mathematics, Arlington, TX 76019-0408, USA
Email: gulk@bu.edu(G. Karaduman)

on the right are represented by $f \in \mathbb{R}^n$ and $g \in \mathbb{R}^m$, while the solution vectors are denoted by $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$.

The KKT system with rank deficiency poses some difficulties for numerical methods that require special techniques for optimal solutions [1]. Various techniques have been developed over the years to solve such systems, each with its own limitations. One of these is the regularization method, which involves adding a regularization term to ensure that the system [20] is non-singular. Tikhonov regularization is a leading example of this technique [13]. However, this approach can introduce bias into the solution and requires careful parameter tuning.

Another approach is constraint preconditioning [4]. This method uses a preconditioner to constrain the solution space and effectively reduces the system's rank deficiency. Constraint preconditioning methods aim to improve convergence but may be computationally expensive, particularly for large-scale problems. The null space method is also an important technique [8]. It uses the null space of the KKT system to reduce its dimensionality by leveraging the null space of B_2, but its effectiveness diminishes when B_2 has a high rank or is ill-conditioned. The static condensation method is an example of this approach. In addition to all these studies, we can list articles that develop various techniques on KKT systems [3] [5] [15] [16] [22]. Additionally, methods such as the Uzawa algorithm and the augmented Lagrangian method [6, 7, 10] have been widely employed, yet they often struggle with slow convergence rates when dealing with highly rank-deficient systems. Iterative methods offer an approach to overcome the rank deficient problems [17] [23] [19]. They are particularly effective in solving large-scale problems by recursively solving the KKT system.

To address these limitations, this work introduces an iterative approach tailored explicitly for rank-deficient KKT systems, where $B_2 \in \mathbb{R}^{m\times n}$ has limited rank, i.e., $l \leq m$ and $m \ll n$. Our method constructs a projection matrix that transforms the original problem into a least squares problem, which can then be efficiently solved using iterative solvers such as LSMR [9]. Unlike standard regularization methods, our approach does not introduce additional regularization terms, thus avoiding artificial bias. Moreover, by explicitly identifying the linearly independent rows of B_2, our method ensures numerical stability and enhances convergence. While our technique focuses on rank-deficient real matrices, it can be readily adapted to handle full-rank real and complex coefficient matrices.

The main structure of this study is as follows. Section 2 introduces how the projection matrix is constructed and provides a theoretical analysis of the projection approach. Section 3 identifies the linearly independent rows within the matrix B_2. Section 4 outlines the solution to the problem and presents the algorithmic framework for our method. Section 5 presents the numerical results and accompanying figures. Finally, the concluding section draws insights based on the findings.

2. Formulation of Projection Matrix Construction

This section outlines the process of constructing the projection matrix, assuming that B_2 is not a full-rank matrix and can have its rows rearranged into a specific partition format, as shown below.

$$PB_2 = \begin{bmatrix} B_{21} \\ B_{22} \end{bmatrix} \in \mathbb{R}^{m\times n}, \tag{2.1}$$

where $P \in \mathbb{R}^{m\times m}$ is a permutation matrix, $B_{21} \in \mathbb{R}^{l\times n}$ is a full rank matrix and $\text{rank}(B_2) = l \leq m$. In particular, the rows of B_{21} are linearly independent. Similarly a vector $g \in \mathbb{R}^m$ can be permuted as $Pg = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$.

Theorem 2.1. *Assume that B_2 is not a full rank matrix. Permute B_2 using the formula* (2.1)*, where $P \in \mathbb{R}^{m\times m}$ is a permutation matrix, $B_{21} \in \mathbb{R}^{l\times n}$ has full row rank, and both* $\text{rank}(B_2)$ *and* $\text{rank}(B_{21})$ *equal l, which is less than m. Additionally, $B_{22} \in \mathbb{R}^{(m-l)\times n}$. Consequently, each row in B_{22} can be expressed as a linear combination of rows in B_{21}. Therefore, there exists a matrix $C \in \mathbb{R}^{(m-l)\times l}$ such that $B_{22} = CB_{21}$.*

Proof. : Let

$$\begin{bmatrix} B_{21} \\ B_{22} \end{bmatrix} = \left[\begin{array}{c} b_1 \\ b_2 \\ \vdots \\ b_l \\ \hline b_{l+1} \\ \vdots \\ b_m \end{array}\right]$$

where $\{b_1, b_2, \ldots, b_l\}$ represent the row vectors of B_{21}, and $\{b_{l+1}, b_{l+2}, \ldots, b_m\}$ represent the row vectors of B_{22}. As $\{b_{l+1}, b_{l+2}, \ldots, b_m\}$ are linearly dependent on the rows of B_{21}, it is possible to express every row vector of B_{22} as a linear combination of the vectors $\{b_1, b_2, \ldots, b_l\}$. This implies the existence of scalar coefficients $c_{l+1,1}, c_{l+1,2}, \ldots, c_{l+1,l}$, $c_{l+2,1}, c_{l+2,2}, \ldots, c_{l+2,l}$, $\ldots$, $c_{m,1}, c_{m,2}, \ldots, c_{m,l}$ such that,

$$\begin{aligned} b_{l+1} &= c_{l+1,1}b_1 + c_{l+1,2}b_2 + \ldots + c_{l+1,l}b_l, \\ b_{l+2} &= c_{l+2,1}b_1 + c_{l+2,2}b_2 + \ldots + c_{l+2,l}b_l, \\ &\vdots \\ b_m &= c_{m,1}b_1 + c_{m,2}b_2 + \ldots + c_{m,l}b_l. \end{aligned}$$

Equivalently,

$$B_{22} = \begin{bmatrix} c_{l+1,1} & c_{l+1,2} & \cdots & c_{l+1,l} \\ \vdots & \vdots & & \vdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,l} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_l \end{bmatrix}$$
$$= CB_{21}.$$

□

Theorem 2.2. *(Rank-Nullity Theorem [2]). Let B_2 be an $m \times n$ matrix. Then*

$$\text{rank}(B_2) + \text{null}(B_2) = n. \tag{2.2}$$

Proof. : [2] □

Theorem 2.3. *If $B_{21} \in \mathbb{R}^{l\times n}$ is a full row rank matrix (i.e., rank(B_{21}) = l), then $B_{21}{B_{21}}^T \in \mathbb{R}^{l\times l}$ is invertible.*

Proof. : Assuming that B_{21} is a matrix with full row rank, the aim is to demonstrate that $B_{21}{B_{21}}^T$ is an invertible $l \times l$ square matrix. To prove this statement, it needs to be established that if $B_{21}{B_{21}}^T w = 0$ for some vector w, then w must be the zero vector. Since B_{21} has a rank of l, it is known that ${B_{21}}^T$ has the same rank of l. From the rank-nullity theorem, it can be concluded that the null space of ${B_{21}}^T$ is trivial, which implies that if ${B_{21}}^T w = 0$, then w must be the zero vector.

$$\begin{aligned} \text{null}({B_{21}}^T) &= l - \text{rank}({B_{21}}^T) \\ &= l - l \\ &= 0. \end{aligned} \tag{2.3}$$

If $B_{21}{B_{21}}^T w = 0$ then

$$\begin{aligned} 0 = w^T B_{21}{B_{21}}^T w &= ({B_{21}}^T w)^T({B_{21}}^T w) \\ &= \langle {B_{21}}^T w, {B_{21}}^T w\rangle \\ &= \left\|{B_{21}}^T w\right\|. \end{aligned}$$

If $\left\|{B_{21}}^T w\right\| = 0$ then ${B_{21}}^T w = 0$. Additionally, since null$({B_{21}}^T) = 0$ as shown in Equation (2.3), the vector w must be the zero vector. Therefore, $B_{21}{B_{21}}^T \in \mathbb{R}^{l\times l}$ is an invertible matrix. □

Definition 2.1. (Right Inverse Matrix) Let $B_{21} \in \mathbb{R}^{l\times n}$ with rank$({B_{21}}^T) = l$. Then $B_{21}{B_{21}}^T$ is invertible matrix and the right inverse of B_{21} is

$${B_{21}}^+ = {B_{21}}^T(B_{21}{B_{21}}^T)^{-1}, \tag{2.4}$$

with $B_{21}{B_{21}}^+ = I_l$.

Theorem 2.4. *Let $B_{21} \in \mathbb{R}^{l\times n}$ and $g_1 \in \mathbb{R}^l$. Let $x \in \mathbb{R}^n$ be the minimum-norm solution of*

$$\|g_1 - B_{21}x\|_2 = \min_{w\in\mathbb{R}^n} \|g_1 - B_{21}w\|_2 . \tag{2.5}$$

Then $x = {B_{21}}^+ g_1$.

Proof. : By [21]. □

Theorem 2.5. *A vector $g \in \mathbb{R}^m$ belongs to the range of B_2 if and only if there exists a vector $x \in \mathbb{R}^n$ that satisfies $B_2x = g$. Given that $B_{21} \in \mathbb{R}^{l\times n}$ with rank(B_{21})=l, x can be expressed as:*

$$x = {B_{21}}^+ g_1, \tag{2.6}$$

where $Pg = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$, $P \in \mathbb{R}^{m\times n}$ is a permutation matrix.

Proof. : ($\Rightarrow$) Suppose that $B_{21} \in \mathbb{R}^{l\times n}$ with $\text{rank}(B_{21}) = l$ and $g \in \mathbb{R}^m$ is in the range of $B_2 \in \mathbb{R}^{m\times n}$. Then

$$B_2x = g, \tag{2.7}$$

for some $x \in \mathbb{R}^n$. Apply the permutation matrix $P \in \mathbb{R}^{m\times m}$ to both sides of equation (2.7).

$$PB_2x = Pg \tag{2.8}$$

$$\begin{bmatrix} B_{21} \\ B_{22} \end{bmatrix} x = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \tag{2.9}$$

$$B_{21}x = g_1. \tag{2.10}$$

$$\tag{2.11}$$

By Theorem (2.4),

$$x = {B_{21}}^+ g_1,$$

where ${B_{21}}^+ = {B_{21}}^T({B_{21}B_{21}}^T)^{-1} \in \mathbb{R}^{n\times l}$.

($\Leftarrow$) Suppose that $x \in \mathbb{R}^n$ takes the form

$$x = {B_{21}}^+ g_1,$$

${B_{21}}^+ = {B_{21}}^T({B_{21}B_{21}}^T)^{-1} \in \mathbb{R}^{n\times l}$ and $g_1 \in \mathbb{R}^l$. It is also known that $B_{22} = CB_{21}$ for some $C \in \mathbb{R}^{(m-l)\times l}$ by Theorem 2.1. Then

$$PB_2{B_{21}}^+ g_1 = \begin{bmatrix} B_{21} \\ B_{22} \end{bmatrix} {B_{21}}^+ g_1 \tag{2.12}$$

$$= \begin{bmatrix} B_{21} \\ CB_{21} \end{bmatrix} \left[{B_{21}}^T({B_{21}B_{21}}^T)^{-1}\right] g_1 \tag{2.13}$$

$$= \begin{bmatrix} ({B_{21}B_{21}}^T)({B_{21}B_{21}}^T)^{-1} \\ C({B_{21}B_{21}}^T)({B_{21}B_{21}}^T)^{-1} \end{bmatrix} g_1 \tag{2.14}$$

$$= \begin{bmatrix} I \\ CI \end{bmatrix} g_1 \tag{2.15}$$

$$= \begin{bmatrix} g_1 \\ Cg_1 \end{bmatrix} \tag{2.16}$$

$$= \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \tag{2.17}$$

$$= Pg. \tag{2.18}$$

$$\tag{2.19}$$

Hence, g is in the range of B_2. This completes the proof of the theorem. □

3. Finding the Full Row Rank Matrix

This section discusses a method for identifying the linearly independent rows of the (2,1)-block matrix B_2. In certain cases, the initial l rows of B_2, where l corresponds to the rank of B_2, may lack linear independence. To determine which rows of B_2 are linearly independent, QR factorization with column pivoting is employed on the transpose of B_2. This approach is outlined in [12]. This factorization can be expressed as follows,

$${B_2}^T P_\pi = Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} \tag{3.1}$$

$$= \left[\hat{b}_1 \dots \hat{b}_m\right], \tag{3.2}$$

where, Q is an orthogonal matrix of size $n \times n$, R_{11} is an upper triangular and nonsingular matrix of size $l \times l$, and P_π is a permutation matrix of size $m \times m$.

Suppose Householder matrices $H_1, \dots, H_{k-1}$ and permutation matrices $P_{\pi_1}, \dots, P_{\pi_{k-1}}$ have been computed for some integer k. The transpose of matrix B_2 can be transformed into an upper triangular matrix $R^{(k-1)}$ using these matrices. $R^{(k-1)}$ can then be partitioned into ${R_{11}}^{(k-1)}$ and ${R_{22}}^{(k-1)}$, where ${R_{11}}^{(k-1)}$ is a nonsingular and upper triangular matrix such that,

$$(H_{k-1} \dots H_1){B_2}^T(P_{\pi_1} \dots P_{\pi_{k-1}}) = R^{(k-1)} \tag{3.3}$$

$$= \begin{bmatrix} {R_{11}}^{(k-1)} & {R_{12}}^{(k-1)} \\ 0 & {R_{22}}^{(k-1)} \end{bmatrix}. \tag{3.4}$$

Suppose that the remaining rows and columns of ${R_{22}}^{(k-1)}$ are denoted by ${u_k}^{(k-1)}, \dots, {u_m}^{(k-1)}$ such that

$${R_{22}}^{(k-1)} = \left[{u_k}^{(k-1)}, \dots, {u_m}^{(k-1)}\right] \tag{3.5}$$

is divided into columns, and let $i \geq k$ be the smallest index that satisfies

$$\left\| {u_i}^{(k-1)} \right\|_2 = \max \left\{ \left\| {u_k}^{(k-1)} \right\|_2, \ldots, \left\| {u_m}^{(k-1)} \right\|_2 \right\}. \tag{3.6}$$

If $\left\| {u_i}^{(k-1)} \right\|_2 = 0$, the calculation must be stopped. Otherwise, if $\left\| {u_i}^{(k-1)} \right\|_2 > 0$, the matrix $P_\pi \in \mathbb{R}^{m \times m}$ needs to be found by interchanging the p-th and k-th columns. Then, the Householder matrix H_k needs to be computed such that $R^{(k)} = H_k R^{(k-1)} P_{\pi_k}$ and ensuring that $R^{(k)}(k+1:n, k) = 0$. After completing the k-th step, it needs to be verified if $\frac{|u_{kk}|}{|u_{11}|}$ is less than a tolerance value 10^{-12}, denoted as "tol". If $\frac{|u_{kk}|}{|u_{11}|} >$tol, $\hat{b}_k$ is a column of ${B_{21}}^T$. Then the matrix ${B_{21}}^T$ will be

$${B_{21}}^T = \left[\hat{b}_1 \ \hat{b}_2 \ldots \hat{b}_{k-1} \ \hat{b}_k \right], \tag{3.7}$$

where $\hat{b}_1$, $\hat{b}_2$, ... ,$\hat{b}_{k-1}$, $\hat{b}_k$ are linearly independent columns of ${B_{21}}^T$. Then the matrix B_{21} will be

$$B_{21} = \begin{bmatrix} \hat{b}_1^T \\ \hat{b}_2^T \\ \vdots \\ \hat{b}_{k-1}^T \\ \hat{b}_k^T \end{bmatrix}. \tag{3.8}$$

4. Solving the Rank-Deficient KKT Problem

The KKT system (1.1) can be expressed in the following

$$\begin{aligned} Ax + {B_1}^T y &= f, \\ B_2 x &= g. \end{aligned}$$

According to Theorem (2.5), the solution vector $x \in \mathbb{R}^n$ can be represented as,

$$x = {B_{21}}^+ g_1. \tag{4.1}$$

Substituting $x = {B_{21}}^+ g_1$ into $Ax + {B_1}^T y = f$, the following equation is obtained,

$$A{B_{21}}^+ g_1 + {B_1}^T y = f,$$

then

$${B_1}^T y = f - A{B_{21}}^+ g_1, \tag{4.2}$$

where $f - A{B_{21}}^+ g_1$ belongs to the vector space $\mathbb{R}^n$.

When dealing with an overdetermined linear system in equation (4.2) where $m \ll n$, it is possible to use advanced numerical methods to solve the system efficiently and accurately, this allows us to reformulate the problem (4.2) as a least squares problem.

$$\min_y \left\| {B_1}^T y - (f - A{B_{21}}^+ g_1) \right\|_2, \tag{4.3}$$

the coefficient matrix of this least squares problem belongs to $\mathbb{R}^{n\times m}$. By solving this problem, the solution vector can be obtained.

The LSMR algorithm, based on the Golub-Kahan bidiagonalization process [11], employs recursive transformations to convert the matrix $\left[\hat{b}\ {B_1}^T\right]$ into a bidiagonal form. This process forms the essential foundation for the LSMR method.

To elaborate, the transformation procedure can be demonstrated as follows,

1. Given initial guess $y_0 = 0$. The residual $r_0 = \hat{b}$;
2. Set $\beta_1 = \|r_0\|_2$, $u_1 = r_0/\beta_1$, $\hat{v}_1 = B_1 u_1$, $\alpha_1 = \|\hat{v}_1\|_2$, $v_1 = \hat{v}_1/\alpha_1$;
3. For $i = 1, 2, \cdots$, do

$$\hat{u}_{i+1} = {B_1}^T v_i - \alpha_i u_i,\ \ \beta_{i+1} = \|\hat{u}_{i+1}\|_2,\ \ u_{i+1} = \hat{u}_{i+1}/\beta_{i+1},$$
$$\hat{v}_{i+1} = B_1 u_{i+1} - \beta_{i+1} v_i,\ \ \alpha_{i+1} = \|\hat{v}_{i+1}\|_2,\ \ v_{i+1} = \hat{v}_{i+1}/\alpha_{i+1}.$$

Upon completing the k-th step, assuming no breakdown (i.e., $\beta_{i+1} = 0$ or $\alpha_{i+1} = 0$), the following relationship holds

$${B_1}^T V_k = U_{k+1} F_k,\ \ B_1 U_{k+1} = V_k F_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T, \tag{4.4}$$

where $V_k = \left[v_1\ v_2\ \cdots\ v_k\right]$, $U_k = \left[u_1\ u_2\ \cdots\ u_k\right]$, $U_k^{\mathrm{T}} U_k = I$, $V_k^T V_k = I$, and

$$F_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix}.$$

The k-th approximate solution y_k is sought over span(V_k), and V_k is the orthogonal basis of the Krylov subspace.

$$\mathcal{K}_k\left(B_1 {B_1}^T, B_1 r_0\right) = \text{span}\left(B_1 r_0, B_1 {B_1}^T (B_1 r_0), \cdots, (B_1 {B_1}^T)^{k-1} (B_1 r_0)\right).$$

Denote by $y_k = V_k t_k$. In LSMR, the goal is to minimize $\|B_1 r_k\|_2$ over span(V_k), where $r_k = \hat{b} - {B_1}^T y_k$ is the k-th residual. According to (4.4), the following holds,

$$B_1 r_k = V_{k+1} \left(\beta_1 \alpha_1 e_1 - \begin{bmatrix} F_k^T F_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{bmatrix} t_k \right).$$

Since ${V_{k+1}}^T V_{k+1} = I_{k+1}$, the reduced problem is the following,

$$\min_t \|B_1 r\|_2 = \min_t \left\| \bar{\beta}_1 e_1 - \begin{bmatrix} F_k^T F_k \\ \bar{\beta}_{k+1} e_k^T \end{bmatrix} t \right\|_2,$$

where $\bar{\beta}_k = \alpha_k \beta_k$ and $\bar{\beta}_1 = \alpha_1 \beta_1$. The LSMR method employs the double QR decomposition on $F_k^T F_k$ to minimize $\|B_1 r\|_2$during the k-th iteration.

5. Numerical Results

This section presents a comprehensive set of numerical tests to demonstrate the performance of the proposed method. The numerical examples address a variety of problems, including combinatorial issues, linear programming problems, and biochemical networks. The aim is to illustrate the superiority of the RDKKT iteration method over the well-established generalized minimum residual method (GMRES) [18] and least squares minimum residual methods (LSMR) [9]. This will be accomplished by thoroughly comparing the numerical results of these iteration methods, focusing on the number of iterations and their elapsed time in seconds (CPU). The results offer valuable insight into how the relative residual changes across iterations of the RDKKT algorithm.

Table 1. Rank-Deficient Testing Matrices B_2: Characteristics of the rank-deficient matrices used in the experiments, including their dimensions, sparsity, and application domains

Matrix	m	n	nnz	application
`GL6_D_10`	163	341	2053	Combinatorial Problem
`lp_scorpion`	388	466	1534	Linear Programming Problem
`GL6_D_9`	340	545	4349	Combinatorial Problem
`GL6_D_8`	544	637	6153	Combinatorial Problem
`GL7d26`	305	2798	7412	Combinatorial Problem
`N_biocarta`	1922	1996	4335	Biochemical Network
`lp_ship12s`	1151	2869	8284	Linear Programming Problem
`N_pid`	3625	3923	8054	Biochemical Network

Table 1 presents a collection of example matrices representing the (2,1)-block matrix B_2. The matrices are not only rank-deficient but also originate from diverse real-world problems, highlighting the relevance of the proposed method across multiple domains. Matrices such as `GL6_D_10`, `GL6_D_9`, `GL6_D_8`, and `GL7d26` arise in combinatorial optimization and graph theory applications, where solving large-scale constrained optimization problems is crucial. These problems frequently appear in network design, scheduling, and integer programming. Matrices like `lp_scorpion` and `lp_ship12s` are derived from linear programming models, which are commonly used in logistics, resource allocation, and operational research. The rank deficiency in these cases often results from constraints that introduce dependencies, making the solution process more challenging. Additionally, matrices such as `N_biocarta` and `N_pid` originate from network-based models in systems biology, capturing interactions between biomolecules in pathways such as metabolic or signaling networks. In these cases, rank deficiency occurs due to conservation laws and redundancies in biochemical reactions, making these systems particularly suitable for testing the effectiveness of iterative solvers.

Each entry in Table 1 provides information about the number of columns in B_2, denoted by n, and the number of rows in B_2, denoted by m. To construct the

coefficient $\mathcal{K}$, sparse matrices are randomly assigned to $A \in \mathbb{R}^{n\times n}$ and $B_1 \in \mathbb{R}^{n\times m}$, resulting in the following structure.

$$\mathcal{K} = \begin{bmatrix} A & {B_1}^T \\ B_2 & 0 \end{bmatrix}.$$

This formulation allows for examining the method's performance across various problem instances of different sizes and characteristics, facilitating a comprehensive analysis of its effectiveness and efficiency. Given the rank deficiency of the system, a thorough review is conducted to determine whether a solution exists. The consistency of the system $\mathcal{K}z = b$ is assessed by comparing the ranks of the coefficient matrix $\mathcal{K}$ and the augmented matrix $[\mathcal{K}\ b]$. Extensive investigations consistently demonstrate that the rank of $\mathcal{K}$ matches the rank of $[\mathcal{K}\ b]$in every case. To evaluate the proposed algorithm 1, the relative residual is defined as

$$\frac{\left\|b - \mathcal{K}z\right\|_2}{\left\|b\right\|_2}. \tag{5.1}$$

Two conditions define the stopping criterion: reaching 10000 iterations or achieving a relative residual (5.1) below 10^{-8}. Calculations in this research were performed using MATLAB version R2023b on a computer equipped with an Apple M2 chip and 16GB of RAM. This information gives insights into the computational environment used for conducting our analyses, ensuring transparency and reproducibility of the results.

The framework for solving the rank-deficient KKT problems (RDKKT) is presented below.

Algorithm 1 Solving KKT system with rank-deficient B_2 and $g \neq 0$ (RDKKT)

Input: $\mathcal{K}$ and b as described in equation (1.1), initial guess $y_0 = 0$, maximum iterations $k_{\max} = 10000$, tolerance $\epsilon = 10^{-8}$;

Output: An approximate solution, denoted as $(x_{\text{opt}}, y_{\text{opt}})$, to the KKT system (1.1).

1: Initialize $k = 0$;
2: **while** $k < k_{\max}$ **do**
3: Apply LSMR method to solve the least squares problem (4.3) and obtain an estimated solution y_k;
4: **if** relative residual (5.1) $< \epsilon$ **then**
5: Set $y_{\text{opt}} = y_k$ and break;
6: **end if**
7: $k \leftarrow k + 1$;
8: **end while**
9: Compute $x_{\text{opt}} = {B_{21}}^{+} g_1$;
10: Return $\begin{bmatrix} x_{\text{opt}} \\ y_{\text{opt}} \end{bmatrix}$.

Figure 1 illustrates the convergence based on the relative residual (5.1). These figures display the approximations obtained using Algorithm 1 on the testing matrices generated from random A, B_1, and $B_2 \in \mathbb{R}^{m\times n}$, as outlined in Table 1.

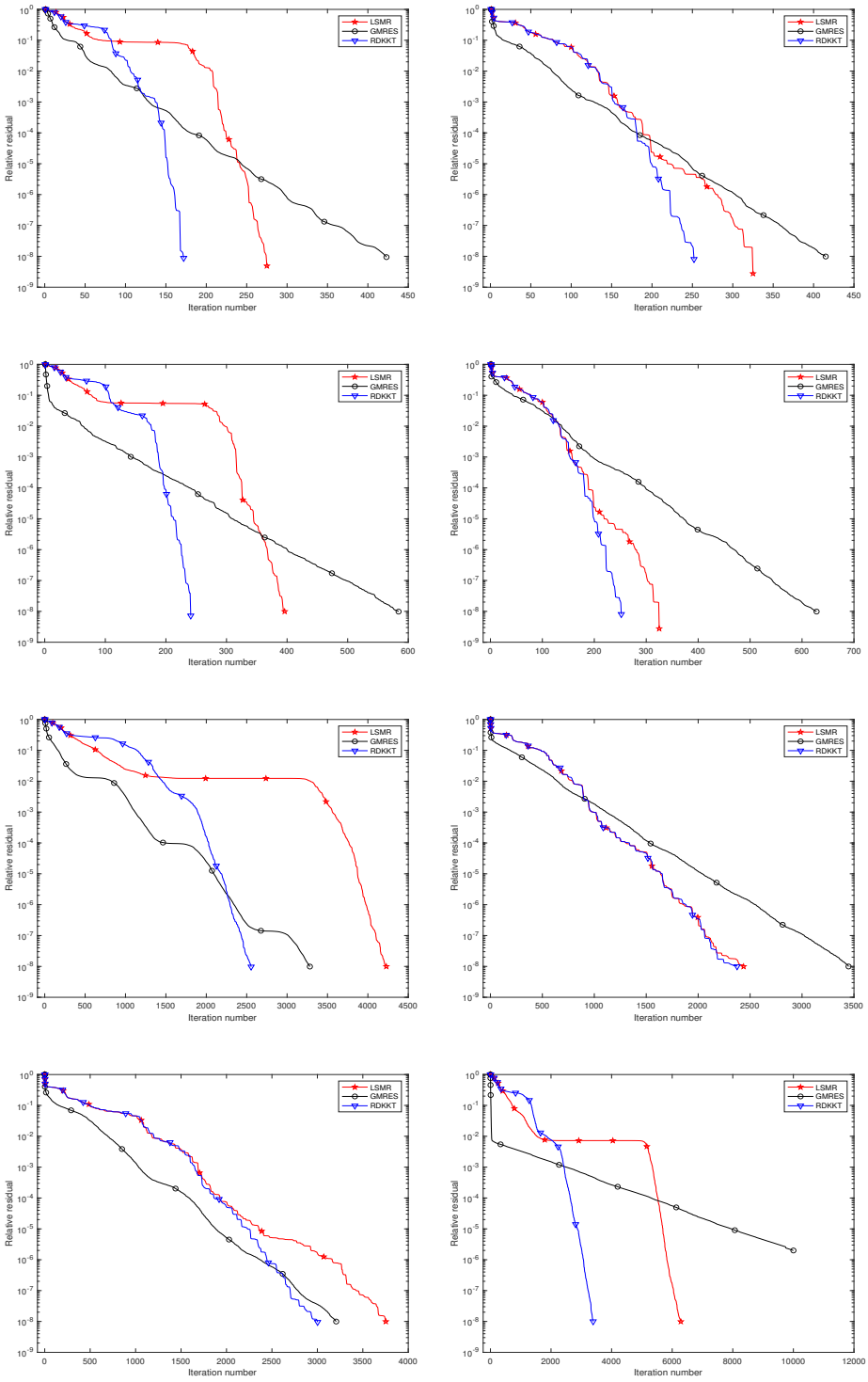

Figure 1. *Left*: Relative residual *vs*. iteration number for $\mathcal{K}$ formed by random A, B_1 and $B_2 \in \mathbb{R}^{m\times n}$ for `GL6_D_10`, `GL6_D_9`, `GL7d26`, and `lp_ship12s` ; *Right*: Relative residual *vs*. iteration number for $\mathcal{K}$ formed by random A, B_1 and $B_2 \in \mathbb{R}^{m\times n}$ for `lp_scorpion`, `GL6_D_8`, `N_biocarta`, and `N_pid`.

Table 2 presents the iteration numbers required by LSMR, GMRES, and RDKKT to achieve a relative residual (5.1) of less than or equal to 10^{-8} for various examples.

Table 2. Iteration Number for Different Solvers: Number of iterations required by LSMR, GMRES, and RDKKT to solve the rank-deficient KKT system

Matrix	LSMR	GMRES	RDKKT
GL6_D_10	275	423	172
lp_scorpion	325	415	252
GL6_D_9	396	584	241
GL6_D_8	325	628	252
GL7d26	3282	4228	2555
N_biocarta	2437	3448	2375
lp_ship12s	3753	3207	3002
N_pid	6284	Not available	3391

The results in table 2 show that RDKKT consistently outperforms both LSMR and GMRES regarding iteration counts for most matrices. For instance, for the matrix GL6_D_10, RDKKT requires only 172 iterations, significantly lower than LSMR's 275 and GMRES's 423 iterations. This trend continues with other matrices such as lp_scorpion, GL6_D_9, and GL6_D_8, where RDKKT maintains a lower iteration count compared to the other methods.

Notably, the difference in iteration counts becomes more pronounced with larger and more complex matrices. For example, with GL7d26, RDKKT requires 2555 iterations, while LSMR and GMRES require 3282 and 4228, respectively. Similarly, for the matrix N_biocarta, RDKKT again demonstrates its efficiency by requiring 2375 iterations compared to LSMR's 2437 and GMRES's 3448.

Interestingly, in the case of the matrix lp_ship12s, LSMR requires more iterations (3753) than GMRES (3207), which is an exception to the general trend observed. However, RDKKT still maintains a competitive edge with 3002 iterations. For the matrix N_pid, GMRES results were not available, but RDKKT (3391) still needed fewer iterations compared to LSMR (6284). The results indicate that RDKKT is generally more efficient, requiring fewer iterations to converge to the desired relative residual than LSMR and GMRES, particularly for larger and more complex matrices.

The CPU time is summarized in table 3. This table lists the CPU times required by each method to achieve convergence, highlighting the efficiency and computational cost associated with the RDKKT method compared to LSMR and GMRES. According to the reported CPU times RDKKT generally has the lowest CPU time across the various matrices, underscoring its efficiency. For example, the matrix GL6_D_10 requires only 0.588 seconds with RDKKT, compared to 0.942 seconds with LSMR and 1.448 seconds with GMRES. Similarly, for lp_scorpion, RDKKT takes 0.705 seconds, significantly less than LSMR's 0.968 seconds and GMRES's 1.236

Table 3. CPU Time Comparison for Different Solvers: Computational time in seconds for solving the rank-deficient KKT system using LSMR, GMRES, and RDKKT

Matrix	LSMR	GMRES	RDKKT
GL6_D_10	0.942	1.448	0.588
lp_scorpion	0.968	1.236	0.705
GL6_D_9	1.143	1.685	0.695
GL6_D_8	1.110	2.145	0.860
GL7d26	6.212	8.002	4.835
N_biocarta	4.448	6.293	4.334
lp_ship12s	6.445	5.508	5.155
N_pid	12.703	Not available	6.855

seconds. This trend continues with other matrices such as GL6_D_9 and GL6_D_8, where RDKKT shows marked reductions in CPU time compared to LSMR and GMRES. For larger and more complex matrices, such as GL7d26 and N_biocarta, RDKKT maintains its efficiency, requiring 4.835 seconds and 4.334 seconds, respectively, while LSMR and GMRES require significantly more time. An interesting observation is with the matrix lp_ship12s, where GMRES shows a lower CPU time (5.508 seconds) compared to LSMR (6.445 seconds), yet RDKKT still outperforms both with a time of 5.155 seconds. For the matrix N_pid, GMRES results were not available, but RDKKT (6.855 seconds) still significantly outperformed LSMR (12.703 seconds). The results show that RDKKT consistently demonstrates lower CPU times, making it a more cost-effective choice for solving large-scale linear systems than LSMR and GMRES.

The experimental results demonstrate the remarkable convergence achieved by the projection method when applied to rank-deficient B_2 matrices, indicating the approach's efficacy in solving such systems. The method extends its applicability to singular $\mathcal{K}$ and A matrices, illustrating that a full-rank system is unnecessary for favorable outcomes. This demonstrates the robustness and flexibility of the approach. Re-orthogonalization or preconditioning techniques were not used in the current implementation, but the potential benefits of these methods are acknowledged. In future research, the integration of preconditioning methods is planned to improve the efficiency and accuracy of solving KKT systems.

6. Conclusion

The presented research contributes to the field by offering an iterative method specifically designed for rank-deficient (2,1)-block KKT systems. The findings, supported by experimental evidence, highlight the effectiveness of the projection method in addressing the challenges posed by singular systems. The notable convergence achieved by the method underscores its potential to deliver successful outcomes

in solving these systems. The results from the experiments clearly demonstrate that a full-rank system is not necessary for obtaining satisfactory results with this approach. This illustrates the robustness and versatility of the method, allowing it to address scenarios where traditional methods may struggle. Beyond its theoretical significance, the proposed method has practical applications in a range of real-world optimization problems. Rank-deficient KKT systems frequently arise in large-scale constrained optimization tasks, such as linear and quadratic programming, network flow optimization, and control problems in engineering. Additionally, these systems play a crucial role in economic modeling, machine learning optimization frameworks, and structural mechanics, where constrained formulations often lead to singular systems. By extending the method's applicability to singular matrices, a valuable solution is provided that can tackle a wider array of practical problems, ensuring its relevance across multiple disciplines.

References

[1] Aslani, H., & Khojasteh Salkuyeh, D. *On the preconditioned APSS iterative method for singular coupled saddle point problems*, 2024, Mathematical Communications, 29(1), 21-37.

[2] Axler, S. *Linear Algebra Done Right*, 1997, 2nd edn. Springer, New York.

[3] Benzi, M., Golub, G., Liesen, J. *Numerical solution of saddle point problems*, Acta Numerica, 2005, vol. 14, pp. 1–137. DOI : 10.1007/978−3−031−41026−0.

[4] Bergamaschi, L., Gondzio, J., Zilli, G. *Preconditioning indefinite systems in interior point methods for optimization*, Computational Optimization and Applications, 2004, vol. 28, pp. 149–171. DOI : 10.1007/978 − 3 − 031 − 41026 − 0.

[5] Cao, Y., Yi, S. *A class of Uzawa-PSS iteration methods for nonsingular and singular non-Hermitian saddle point problems*, Applied Mathematics and Computation, 2016, vol. 275, pp. 41–49. DOI : 10.1016/$j.amc$.2015.11.049.

[6] Cheng, G., & Li, J. C. *Class of Uzawa-NPHSS iteration method for solving nonsingular and singular saddle point problems*, Linear and Multilinear Algebra, 2022, 70(14), 2706-2738. DOI : 10.1080/03081087.2020.1811628.

[7] Cui, M. *A sufficient condition for the convergence of the inexact Uzawa algorithm for saddle point problems*, Journal of Computational and Applied Mathematics, 2002, vol. 139(2), pp. 189–196. DOI : 10.1016/S0377−0427(01)00430−7.

[8] Fletcher, R., Johnson, T. *On the stability of null-space methods for KKT systems*, SIAM Journal on Matrix Analysis and Applications, 1997, vol. 18(4), pp. 938–958. DOI : 10.1137/S0895479896297732.

[9] Fong, D.C., Saunders, M. *LSMR: An iterative algorithm for sparse least-squares problems*, SIAM Journal on Scientific Computing, 2011, vol. 33(5), pp. 2950–2971. DOI : 10.1137/10079687X.

[10] Fortin, M., Glowinski, R. *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems*, 2000, 1st edn. North-Holland, Amsterdam.

[11] Golub, G.H., Kahan, W. *Calculating the singular values and pseudo-inverse of a matrix*, SIAM Journal on Numerical Analysis, 1965, vol. 2, pp. 205–224. *DOI* : 10.1137/0702016.

[12] Golub, G.H., Van Loan, C.F. *Matrix Computations*, 4th edn. Johns Hopkins University Press, 2013, Baltimore.

[13] Kress, R. *Tikhonov Regularization. In: Linear Integral Equations.* Applied Mathematical Sciences, 1989, Springer, Berlin, Heidelberg. *DOI* : $10.1007/978-3-642-97146-4_16$.

[14] Kuhn, H.W., Tucker, A.W. *Nonlinear programming.* In: Giorgi, G., Kjeldsen, T. (eds) Traces and Emergence of Nonlinear Programming. Birkhäuser, Basel, 2014. *DOI* : $10.1007/978-3-0348-0439-4_11$.

[15] Liao, L., Zhang, G., Wang, X. *Preconditioned iterative method for nonsymmetric saddle point linear systems*, 2021, Computers & Mathematics with Applications, vol. 98, pp. 69–80. *DOI* : 10.1016/*j.camwa*.2021.07.002.

[16] Regev, S., Chiang, N. Y., Darve, E., Petra, C. G., Saunders, M. A., Świrydowicz, K., & Peleš, S. *HyKKT: a hybrid direct-iterative method for solving KKT linear systems*, Optimization Methods and Software, 2023, 38(2), 332-355. *DOI* : 10.1080/10556788.2022.2124990

[17] Rosolia, U., Lian, Y., Maddalena, E. T., Ferrari-Trecate, G., & Jones, C. N. *On the optimality and convergence properties of the iterative learning model predictive controller*, IEEE Transactions on Automatic Control, 2022, 68(1), 556-563.

[18] Saad, Y., Schultz, M.H. *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 1986, vol. 7(3), pp. 856–869. *DOI* : 10.1137/0907058

[19] Sarin, V. *Efficient iterative methods for saddle point problems.* Ph.D. Thesis, University of Illinois at Urbana-Champaign. ProQuest Dissertations Publishing, 1997.

[20] Saunders, M.A. *Cholesky-based methods for sparse least squares: The benefits of regularization.* In: Linear and Nonlinear Conjugate Gradient-Related Methods, 1996, vol. 100, pp. 92–100.

[21] Watkins, D.S. *Fundamentals of Matrix Computations*, 2nd edn. Wiley-Interscience, New York, 2002. *DOI* : 10.1002/0471249718

[22] Xiao, M., Bo, S., & Wu, Z. *Multiple greedy quasi-newton methods for saddle point problems.* 6th International Conference on Data-driven Optimization of Complex Systems (DOCS), 2024, (pp. 749-754). IEEE. *DOI* : 10.1109/*DOCS*63458.2024.10704381

[23] Xiao, X. Y., & Wang, C. S. *A block upper triangular preconditioner with two parameters for saddle-point problems*, Computers & Mathematics with Applications, 2024, 160, 15-29. *DOI* : 10.1016/*j.camwa*.2024.02.015