Turkish Journal of Mathematics

http://journals.tubitak.gov.tr/math/

Research Article

Turk J Math
(2022) 46: 1330 – 1341
© TÜBİTAK
doi:10.3906/mat-2008-59

An alternative method for SPP with full rank (2,1)-block matrix and nonzero right-hand side vector

GÜl KARADUMAN[1,*], **Mei YANG**[2]
[1]Vocational School of Health Services, Karamanoğlu Mehmetbey University, Karaman, Turkey
[2]School of Mathematical Sciences, Shanghai Jiao Tong University, 800 Dongchuan RD, Shanghai, China

Received: 18.08.2020 • **Accepted/Published Online:** 18.03.2022 • **Final Version:** 05.05.2022

Abstract: We propose an alternative method to solve large linear saddle point problems arising from computational sciences and engineering such as finite element approximations to Stokes problems, image reconstructions, tomography, genetics, statistics, and model order reductions for dynamical systems. Such problems have large sparse 2-by-2 block structure coefficient matrices with zero (2,2)-block matrix. A new technique is presented to solve saddle point problems with full row rank (2,1)-block matrix and nonzero right-hand side vector. By constructing a projection matrix and transforming the original problem into a least squares problem, a new reduced least squares problem is solved via the well-known iterative method LSMR. Numerical experiments show that this new method works very well for the specified saddle point systems.

Key words: Karush-Kuhn-Tucker problem, nonhomogeneous system, linear systems, Krylov subspace method

1. Introduction

Large linear system of saddle point problems arise from statistics [11], electromagnetic [9], incompressible flow [15], and computational fluid dynamics [15]. As an important research topic in numerical linear algebra, researchers make a lot of effort to solve it efficiently. A typical saddle point problem is in the form of

$$\begin{bmatrix} A & {B_1}^T \\ B_2 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$ is large and sparse, $B_1, B_2 \in \mathbb{R}^{m\times n}$ with $n \geq m$, nonzero vectors $f \in \mathbb{R}^n$ and $g \in \mathbb{R}^m$. The vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ are unknown variables. (1.1) can also be written as a linear system with 2-by-2 block coefficient matrix like

$$\mathcal{A}z = b, \tag{1.2}$$

where $\mathcal{A} = \begin{bmatrix} A & {B_1}^T \\ B_2 & 0 \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$, $b = \begin{bmatrix} f \\ g \end{bmatrix}$, and $z = \begin{bmatrix} x \\ y \end{bmatrix}$. (1.1) is known as a Karush-Kuhn-Tucker(KKT) problem [7] as well.

There are many efficient and stable numerical methods for solving (1.1) based on the block structures, such as the HSS method [4], Uzawa methods [1], null space methods [10], etc. Some other approaches for

*Correspondence: gkaraduman@kmu.edu.tr
2010 *AMS Mathematics Subject Classification:* 65F10

solving system (1.2) instead of (1.1) are Krylov subspace methods such as CG [2], GMRES [13], FOM [12] and MINRES [8]. Even though Krylov subspace methods are easy to implement, the convergence of all these methods depends on the spectral properties of the coefficient matrix $\mathcal{A}$. For example, GMRES converges fast if $\mathcal{A}$ has clustered eigenvalues, which cannot be guaranteed for every problem. Moreover, some factorization techniques, such as LDL^T factorization [14], can be used, while it involves a large computational cost of the inverse of A. For these reasons, we try to find a computational saving, stable, and efficient numerical method for solving (1.1), which can also be implemented as easily as GMRES. In this paper, we focus on solving (1.1) when (2,1)-block matrix $B_2 \in \mathbb{R}^{m\times n}$ is full row rank with $m \ll n$, and $g \neq 0$. The key idea of our method is to construct a projection matrix and transform the original problem (1.1) to a least squares problem, which is solved by one of the iterative methods such as LSMR [5]. We work with real matrices but the idea can be easily extended for complex coefficient matrices. The remainder of this paper is organized as follows. In Section 2, we present the theoretical analysis of our projection method including construction of the projection matrix, transforming (1.1) to a least squares problem and solving the least squares system. In this section, we also show the algorithmic framework of the new method. In Section 3, numerical results are displayed to illustrate the performance of the new method. Finally, we make conclusions and give an outlook on our future work in Section 4.

Notation. $\mathbb{R}^{n\times m}$ is the set of all $n\times m$ real matrices, $\mathbb{R}^n = \mathbb{R}^{n\times 1}$, $\mathbb{R} = \mathbb{R}^1$. The superscript "$\cdot^+$" takes the right inverse of a matrix. I_n (or simply I if its dimension is clear from the context) is the $n\times n$ identity matrix, and e_j is its jth column. For a matrix $X \in \mathbb{R}^{n\times m}$, $\mathcal{R}(X)$ and $\mathcal{N}(X)$ denote the range (column space) and null space of X, respectively. Denote by $\|x\|_2$ the Euclidean norm of a vector x, and by $\|X\|_1$ the ℓ_1 operator norm of a matrix X.

2. Main contribution

In this part, we show the details about how to construct and implement the new method.

2.1. Constructing a projection matrix

A projection matrix is constructed to transform (1.1) to a least squares problem with the help of the following theorems. The first theorem is known as the Rank-Nullity theorem.

It was proven in [3] the following theorem is true.

Theorem 2.1 *(Rank-Nullity Theorem [3]). Let B_2 be an $m\times n$ matrix. Then*

$$\operatorname{rank}(B_2) + \operatorname{null}(B_2) = n. \tag{2.1}$$

Based on Theorem 2.1, we can show the presence of the inverse of $B_2{B_2}^T$.

Theorem 2.2 *If $B_2 \in \mathbb{R}^{m\times n}$ is full row rank, $B_2{B_2}^T$ is invertible.*

Proof Suppose that $B_2 \in \mathbb{R}^{m\times n}$ is a full row rank matrix, $\operatorname{rank}{B_2}^T = \operatorname{rank}(B_2) = m$. By Theorem 2.1, we have $\operatorname{null}({B_2}^T) = 0$. Let $B_2{B_2}^T x = 0$ for some vector x, then $\left\|{B_2}^T x\right\|_2^2 = x^T B_2 {B_2}^T x = 0$. This implies ${B_2}^T x = 0$. Since $\operatorname{null}({B_2}^T) = 0$, x can only be zero vector. Thus $B_2{B_2}^T \in \mathbb{R}^{m\times m}$ is invertible. □

Definition 2.3 *(Right Inverse Matrix) Let $B_2 \in \mathbb{R}^{m\times n}$ with rank$(B_2) = m$, the right inverse of B_2 is*

$$B_2{}^{+} = B_2{}^{T}(B_2 B_2{}^{T})^{-1}, \tag{2.2}$$

with $B_2 B_2{}^{+} = I_m$.

It was proven in [16] the following theorem is true.

Theorem 2.4 *Let $B_2 \in \mathbb{R}^{m\times n}$ with rank$(B_2) = m$ and $g \in \mathbb{R}^m$, and let $x \in \mathbb{R}^n$ be the minimum norm solution of*

$$\|g - B_2 x\|_2 = \min_{w\in\mathbb{R}^n} \|g - B_2 w\|_2 , \tag{2.3}$$

Then $x = B_2{}^{+} g$.

Theorem 2.5 *Let $B_2 \in \mathbb{R}^{m\times n}$ with rank$(B_2{}^{T}) = m$. A vector $g \in \mathbb{R}^m$ is in the range of B_2 such that*

$$B_2 x = g, \tag{2.4}$$

where $x \in \mathbb{R}^n$ if and only if x can be written as

$$x = B_2{}^{+} g, \tag{2.5}$$

where $B_2{}^{+} = B_2{}^{T}(B_2 B_2{}^{T})^{-1} \in \mathbb{R}^{n\times m}$.

Proof Let $B_2 \in \mathbb{R}^{m\times n}$ with rank$(B_2{}^{T}) = m$. A vector $g \in \mathbb{R}^m$ is in the range of B_2 such that $B_2 x = g$, where $x \in \mathbb{R}^n$, then x can be written as $x = B_2{}^{+} g$, where $B_2{}^{+} = B_2{}^{T}(B_2 B_2{}^{T})^{-1} \in \mathbb{R}^{n\times m}$. This follows immediately from Theorem 2.4.

To prove the other direction, given $g \in \mathbb{R}^m$, let $x = B_2{}^{+} g$, then

$$B_2 x = B_2 B_2{}^{+} g = (B_2 B_2{}^{T})(B_2 B_2{}^{T})^{-1} g = g.$$

This completes the proof of the theorem. □

Theorem 2.5 implies that we can use the projection B_2^{+} to decouple the saddle point problem (1.1) into a small size subproblem. This is shown as follows.

2.2. Transforming (1.1) to a least squares problem

In this part, we derive how to transform (1.1) into a small size least squares problem. First, we rewrite (1.1) as

$$\begin{aligned} Ax + B_1{}^{T} y &= f, \\ B_2 x &= g. \end{aligned} \tag{2.6}$$

By Theorem 2.5, we know that x takes the form

$$x = B_2{}^{+} g.$$

Then the first equation in (2.6) becomes

$$A{B_2}^{+}g + {B_1}^{T}y = f,$$

which can be rewritten as

$${B_1}^{T}y = f - A{B_2}^{+}g, \tag{2.7}$$

where $A{B_2}^{+}g \in \mathbb{R}^n$.

If we can solve for y in (2.7) via some efficient and accurate numerical method, we can obtain an approximate solution $z = \begin{bmatrix} x \\ y \end{bmatrix}$ to (1.1). Since $m \ll n$, (2.7) is an overdetermined linear system. This indicates that we need to solve an overdetermined least squares problem with the form

$$\min_{y} \left\| {B_1}^{T}y - (f - A{B_2}^{+}g) \right\|_2, \tag{2.8}$$

Once y is obtained numerically, x can also be calculated by $x = {B_2}^{+}g$. Provided y is accurate enough, x should be a good solution in the sense of least squares norm.

Algorithm 1 shows how the efficient matrix-vector product should be computed.

```
Algorithm 1 Efficient matrix-vector product.
Require: $B_2 \in \mathbb{R}^{m\times n}, g \in \mathbb{R}^m$;
Ensure: ${B_2}^{+}g$.
1: Solve $g = (B_2{B_2}^{T})\hat{g}$ for $\hat{g}$;
2: Compute $\hat{g} = {B_2}^{T}\hat{g}$;
```

2.3. Solving (2.8) by LSMR

There are many choices for solving (2.8). We prefer the LSMR method [5] because of its efficiency in computation and easy implementation. We transform (2.8) to

$$\min_{y} \left\| {B_1}^{T}y - \hat{b} \right\|_2, \tag{2.9}$$

where $\hat{b} = f - A{B_2}^{+}g$.

LSMR is based on the Golub-Kahan bidiagonalization process [6] that recursively transforms $\begin{bmatrix} \hat{b} & {B_1}^{T} \end{bmatrix}$ into a bidiagonal form.

Specifically, it can be shown 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

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

After the k-th step and provided no breakdown, (i.e. $\beta_{i+1} = 0$ or $\alpha_{i+1} = 0$), occurs, we have

$$B_1{}^T V_k = U_{k+1} F_k, \ \ B_1 U_{k+1} = V_k F_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T, \tag{2.10}$$

where $V_k = \begin{bmatrix} v_1 & v_2 & \cdots & v_k \end{bmatrix}$, $U_k = \begin{bmatrix} u_1 & u_2 & \cdots & u_k \end{bmatrix}$, $U_k^{\mathrm{T}} U_k = I$, $V_k^T V_k = I$, 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, we 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 (2.10), we have

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

Since $V_{k+1}{}^T V_{k+1} = I_{k+1}$, we have a reduced problem

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

where $\bar{\beta}_k = \alpha_k \beta_k$ and $\bar{\beta}_1 = \alpha_1 \beta_1$. LSMR uses the double QR decomposition on $F_k^T F_k$ to iteratively minimize $\|B_1 r\|_2$ at k-th iteration. The framework of our method (FSPPvLS) is shown below.

Algorithm 2 SPP via least squares with full rank B_2 and $g \neq 0$ (FSPPvLS).

```
Require: $\mathcal{A}$ and $b$ as in (1.2), an initial guess $y_0 = 0$;
Ensure: an approximate solution $\begin{bmatrix} x_{\text{opt}} \\ y_{\text{opt}} \end{bmatrix}$ to the SPP (1.1).
1: solve least squares problem (2.8) by LSMR to find an approximate solution $y_{\text{opt}}$;
2: compute $x_{\text{opt}} = B_2{}^+ g$ via Algorithm 1;
3: return $\begin{bmatrix} x_{\text{opt}} \\ y_{\text{opt}} \end{bmatrix}$.
```

3. Numerical results

In this section, we exhibit some numerical results to illustrate the performance of our method. The numerical experiments show that the comparison of the convergence for solving the problem (1.1) by LSMR and GMRES applied to the whole problem and LSMR applied to the least squares problem (2.8).

All numerical results shown in this work were run using MATLAB version R2017b (9.3.0) on a machine with 2.7 GHz Dual-Core Intel Core i5 and 8GB RAM. The testing matrices with their generic properties are shown in Table 1. Each matrix examples downloaded in the MATLAB format with the built-in right-hand side vector b. The example matrices have the form $\mathcal{A} = \begin{bmatrix} A & {B_1}^T \\ B_2 & 0 \end{bmatrix}$ in Table 1, n represents the number of columns in B_2 and m is the number of the rows in B_2. The size of $\mathcal{A}$ is $(n+m)\times(n+m)$. B_2 has full row rank and g is nonzero vector for each example.

Given guess $\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = 0$ for all problems, we report the relative residual

$$\frac{\|r\|}{\|\mathcal{A}\|\|z\| + \|b\|}, \tag{3.1}$$

where $r = b - \mathcal{A}z$. The stopping criterion is either when the number of iterations reaches 3000 or the relative residual (3.1) is no bigger than tol $= 10^{-8}$. We check the consistency of the system $\mathcal{A}z = b$ by calculating the rank of the coefficient matrix, $\mathcal{A}$ and the rank of the augmented matrix $[\mathcal{A}, b]$. In all our examples, rank$(\mathcal{A})$=rank$([\mathcal{A}, b])$. We make a note that no reorthogonalization or preconditioning is applied for all examples. The idea we used here is easy to be used among a wide selection of problems.

Table 1. Testing matrices.

Matrix	n	m	nonzero	application
`lshape1`	353	98	3807	statistics
`lshape2`	634	179	6926	statistics
`lshape3`	2004	604	23408	statistics
`maxwell1`	88	25	958	electromagnetic
`maxwell2`	368	113	4374	electromagnetic
`maxwell3`	1504	481	18598	electromagnetic
`navier_stokes_N2`	16	7	252	incompressible flow
`navier_stokes_N4`	80	31	1852	incompressible flow
`navier_stokes_N8`	352	127	9372	incompressible flow
`stokes_N2`	16	7	252	computational fluid dynamics
`stokes_N4`	80	31	1852	computational fluid dynamics
`stokes_N8`	352	127	9372	computational fluid dynamics

Figures 1 – 4 show the sparsity pattern of $\mathcal{A}$ and the convergence for solving (1.1) by Algorithm 2, LSMR, and GMRES on the testing matrices.

Table 2 lists the number of cycles needed by algorithms to achieve relative residual less than or equal to 10^{-8}. According to the results, we have the following observations:

- For all examples, GMRES can not make the relative residual as small as 10^{-8} within 3000 iterations.
- Both LSMR and FSPPvLS can make relative residuals reach 10^{-8} within 3000 iterations. Though LSMR is better than GMRES, it is not as good as our method FSPPvLS for most of the examples. We point out that LSMR and FSPPvLS have comparable performance for `navier_ stokes_N2` and `stokes_N2`.
- FSPPvLS obtains the smallest iteration numbers and the fastest convergence rate for each example.

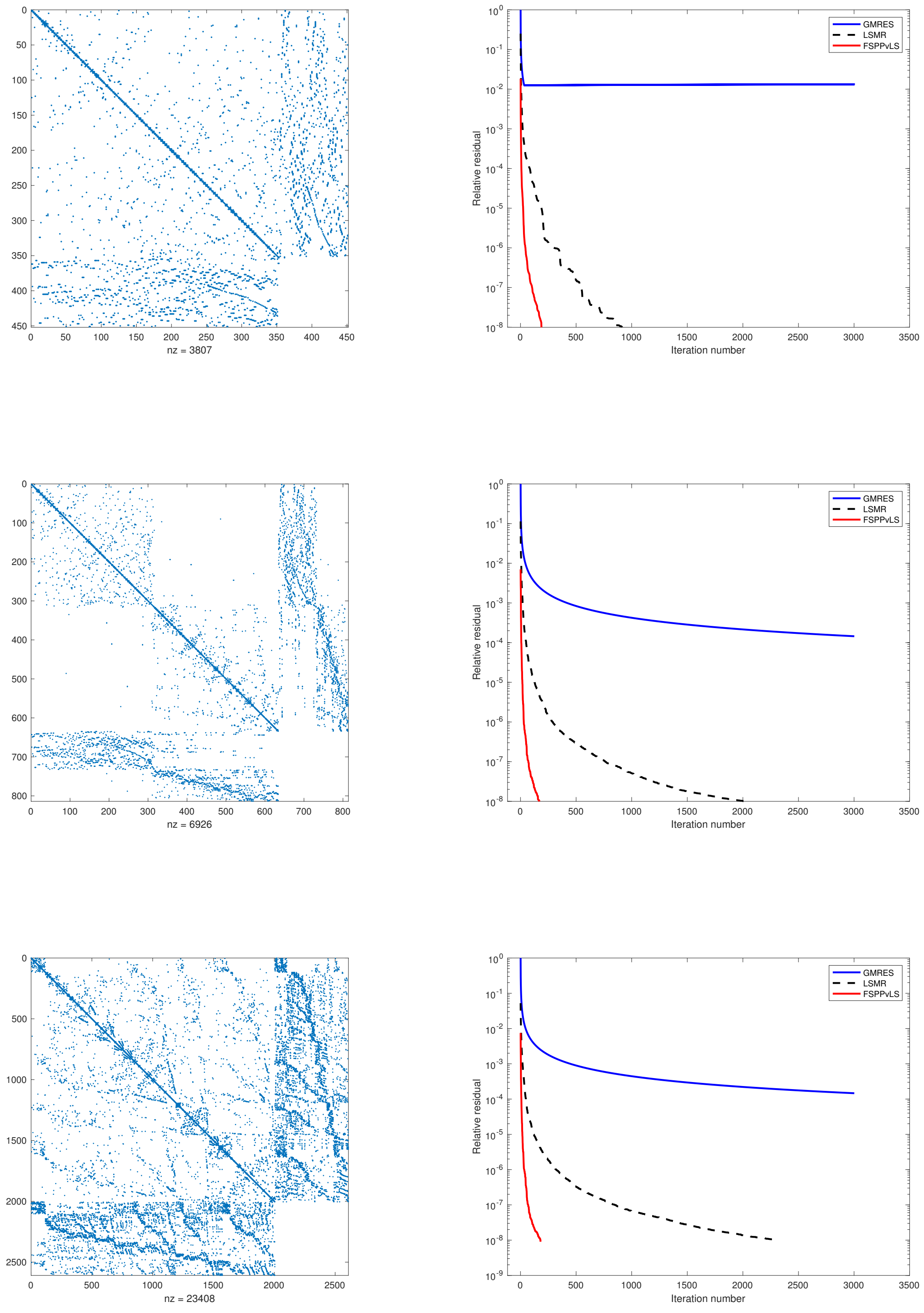

Figure 1. *Left* : Sparsity pattern of $\mathcal{A}$ formed by `lshape1` , `lshape2`, `lshape3`. *Right* : Relative residual *vs.* iteration number for `lshape1`, `lshape2`, `lshape3`.

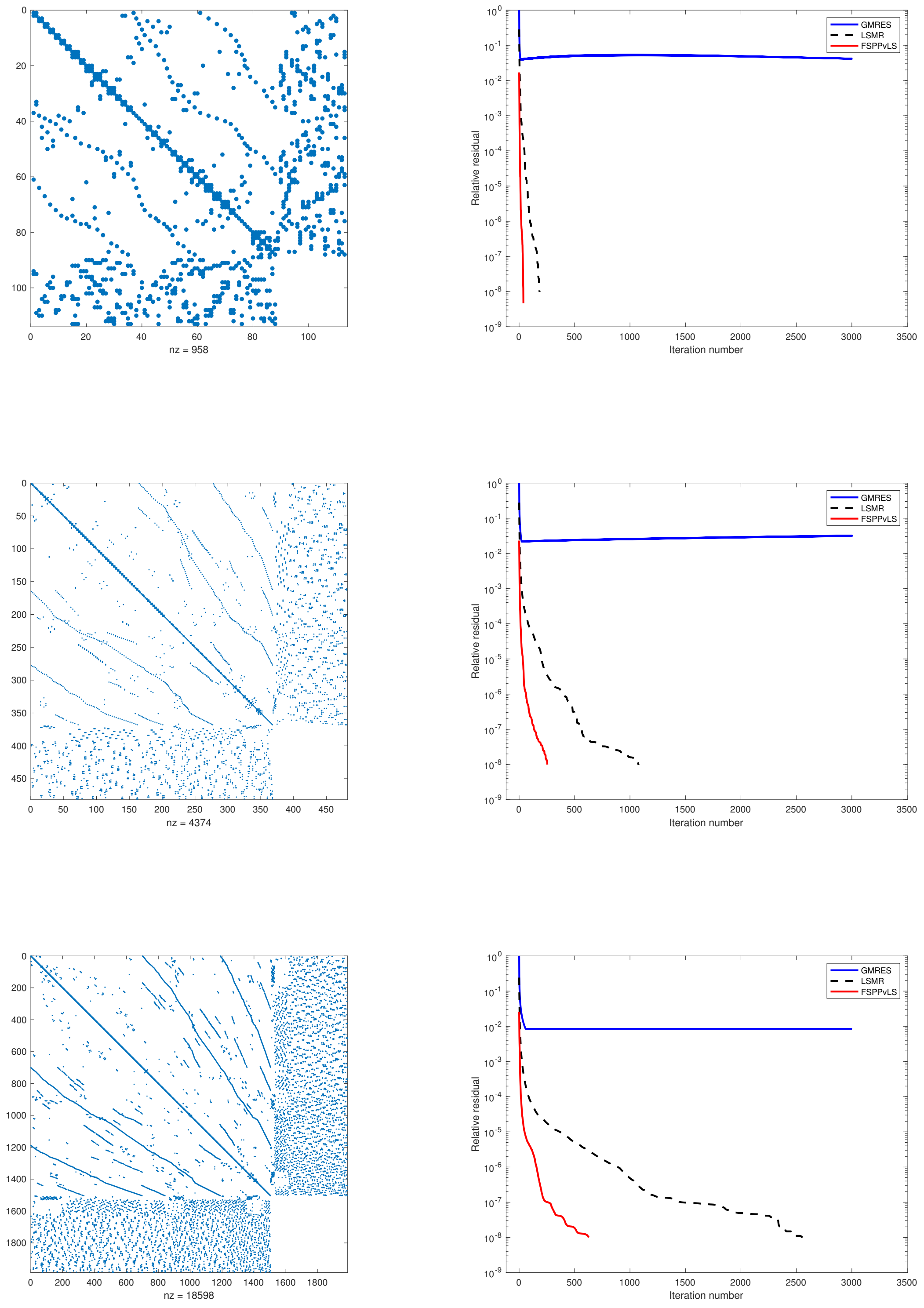

Figure 2. *Left* : Sparsity pattern of $\mathcal{A}$ formed by `maxwell1, maxwell2, maxwell3`. *Right* : Relative residual *vs.* iteration number for `maxwell1, maxwell2, maxwell3`.

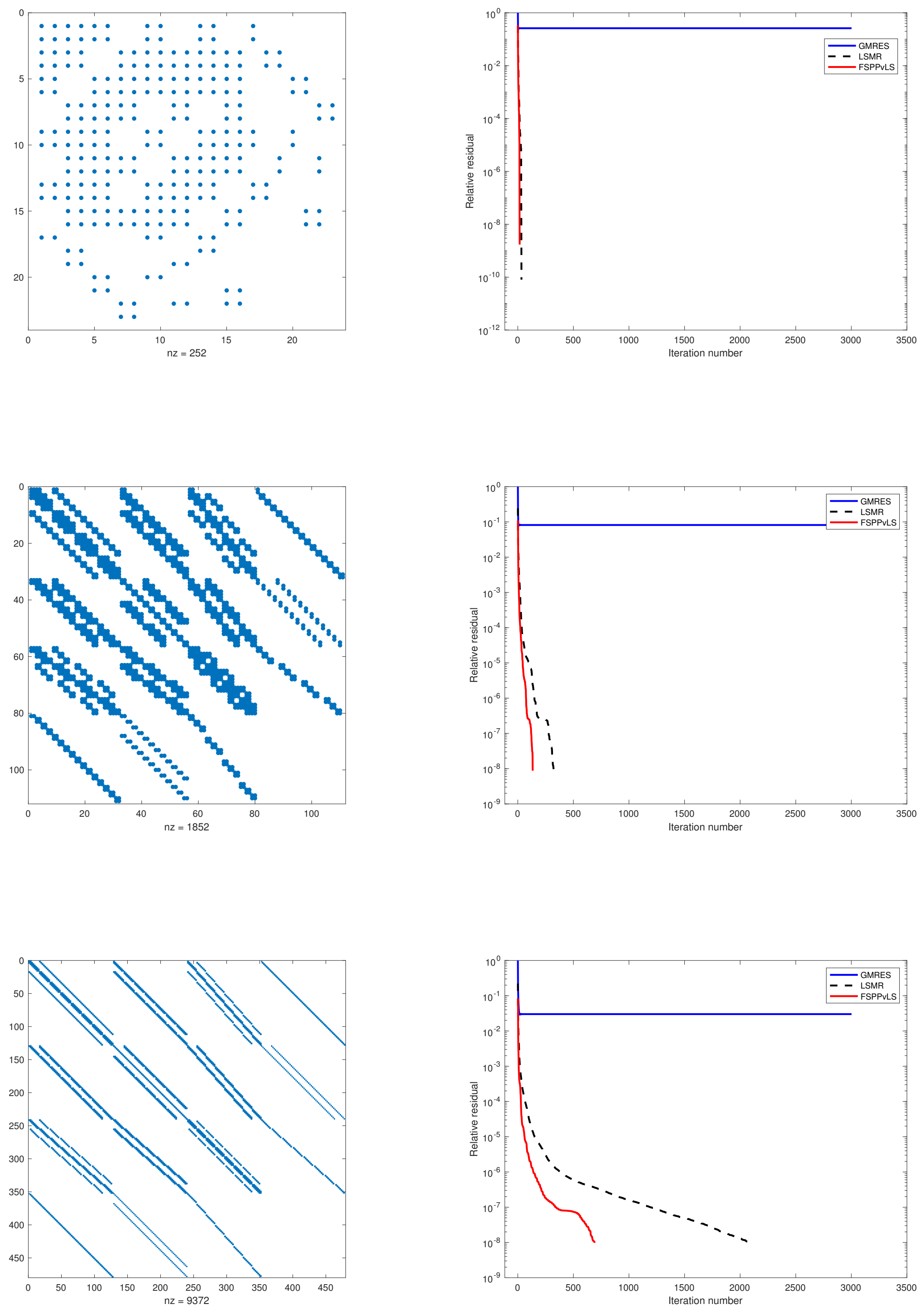

Figure 3. *Left* : Sparsity pattern of $\mathcal{A}$ formed by `navier_stokes_N2`, `navier_stokes_N4`, `navier_stokes_N8`. *Right* : Relative residual *vs.* iteration number for `navier_stokes_N2`, `navier_stokes_N4`, `navier_stokes_N8`.

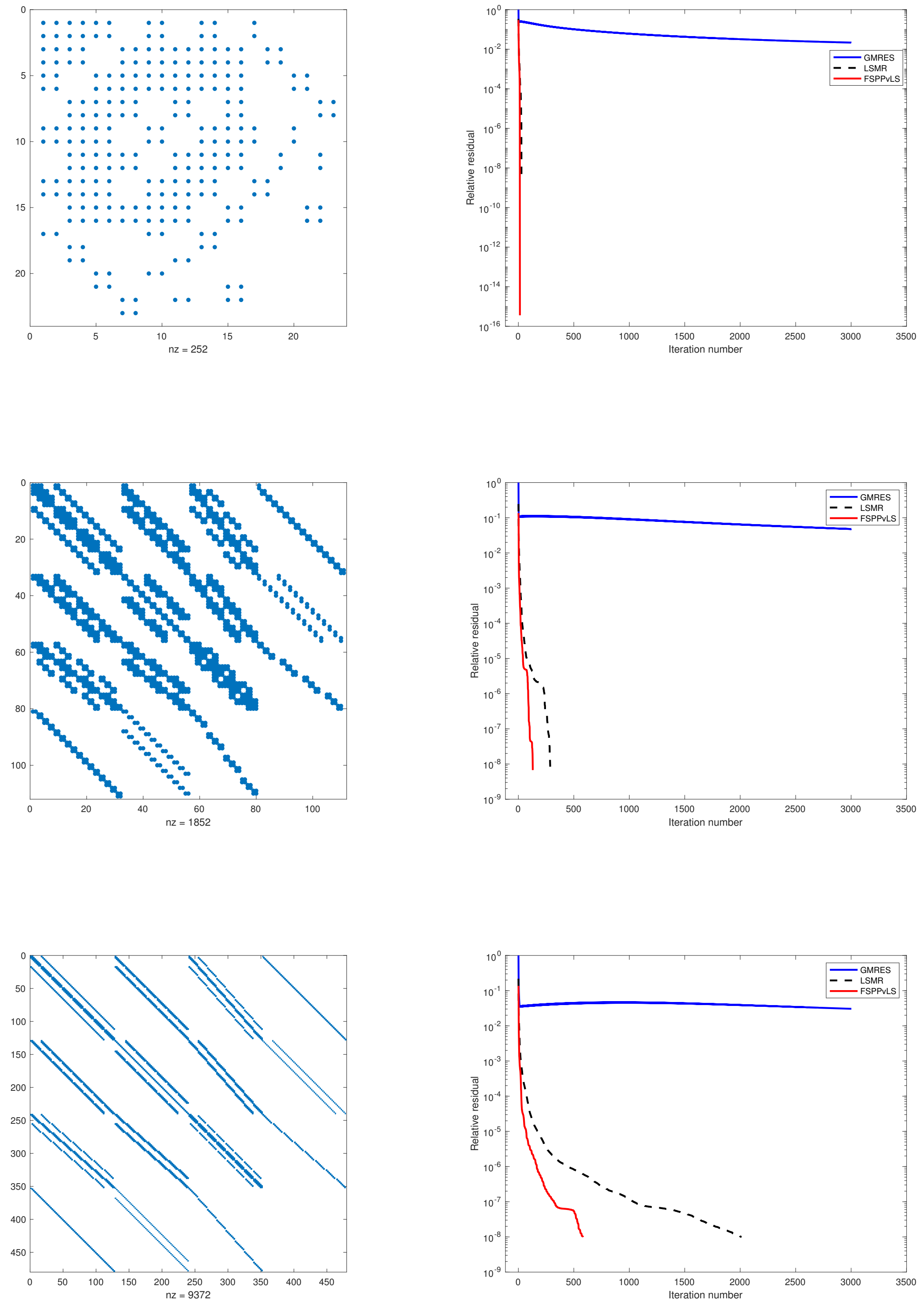

Figure 4. *Left* : Sparsity pattern of $\mathcal{A}$ formed by `stokes_N2, stokes_N4, stokes_N8`. *Right* : Relative residual *vs.* iteration number for `stokes_N2, stokes_N4, stokes_N8`.

Table 2. Iteration number for LSMR, FSPPvLS, and GMRES.

Matrix	*LSMR*	*FSPPvLS*	*GMRES*
`lshape1`	912	188	–
`lshape2`	2006	170	–
`lshape3`	2305	179	–
`maxwell1`	184	38	–
`maxwell2`	1075	254	–
`maxwell3`	2553	628	–
`navier_stokes_N2`	31	16	–
`navier_stokes_N4`	320	133	–
`navier_stokes_N8`	2087	691	–
`stokes_N2`	26	13	–
`stokes_N4`	285	129	–
`stokes_N8`	2010	583	–

4. Conclusions and future

This paper presents an iterative method for large and sparse saddle point systems (1.1). The main contribution of this paper is that the presented technique can be applied to a large class of saddle point problems. In other words, the technique does not necessarily require a specific form of block matrices except the (2,2)-block matrix in the saddle point matrix being 0-matrix and the (2,1)-block matrix B_2 being a full row rank. The advantage of our method it can be applied to systems with a nonzero right-hand side vector. It is one important highlight of the work.

In our method, a projection matrix related to B_2 is constructed and the original problem is reduced into a small size overdetermined least squares problem. Then the least squares problem is solved by LSMR, which is one of the Krylov subspace methods for solving the least squares problems. Numerical results demonstrate the advantages of our method, which are the fastest convergence rate and the easy implementation. One of the future work is to construct a precondition to improve our idea in this research.

References

[1] Arrow KJ, Hurwicz L, Uzawa H. Studies in Linear and Nonlinear Programming. Stanford University Press, Palo Alto, 1958.

[2] Axelsson O. Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations. Linear Algebra and its Applications 1980; 29: 1-16. doi: 10.1016/0024-3795(80)90226-8

[3] Axler S. Linear Algebra Done Right. Springer, New York, 1997.

[4] Bai ZZ, Golub GH, Ng MK. Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems for non-hermitian positive definite linear systems. SIAM Journal on Matrix Analysis and Applications 2003; 24 (3): 603-626. doi: 10.1137/S0895479801395458

[5] Fong DC, Saunders M. LSMR: An iterative algorithm for sparse least-squares problems. SIAM Journal on Scientific Computing 2010; 33 (5): 2950-2971. doi: 10.1137/10079687X

[6] Golub GH, Kahan W. Calculating the singular values and pseudo-inverse of a matrix. Journal of Society for Industrial and Applied Mathematics: Series B, Numerical Analysis 1965; 2 (2): 205-224.

[7] Kuhn HW, Tucker AW. Nonlinear programming. Proceedings of the 2nd Berkeley Symposium on Mathematics, Statistics and Probability, University of California Press, Berkeley, 1951; 481-492.

[8] Paige CC, Saunders MA. Solution of Sparse Indefinite Systems of Linear Equations. SIAM Journal on Numerical Analysis 1975; 12 (4): 617-629. doi: 10.1137/0712047

[9] Perugia I, Simoncini V, Arioli M. Linear Algebra Methods in a Mixed Approximation of Magnetostatic Problems. SIAM Journal on Scientific Computing 1999; 21 (3): 1085-1101. doi: 10.1137/S1064827598333211

[10] Pestana J, Rees T. Null-Space Preconditioners for Saddle Point Systems. SIAM Journal on Matrix Analysis and Applications 2016; 37 (3): 1103-1128. doi: 10.1137/15M1021349

[11] Reid N. Saddlepoint Methods and Statistical Inference. Statistical Science 1988; 3 (2): 213-227. doi: 10.1214/ss/1177012906

[12] Saad Y. Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, 2003. doi: 10.1137/1.9780898718003

[13] Saad Y, Schultz MH. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. SIAM Journal on Scientific and Statistical Computing 1986; 7 (3): 856-869. doi: 10.1137/0907058

[14] Tuma M. A Note on the LDLT Decomposition of Matrices from Saddle-Point Problems. SIAM Journal on Matrix Analysis and Applications 2002; 23 (4): 903-915. doi: 10.1137/S0895479897321088

[15] Turek S. Efficient Solvers for Incompressible Flow Problems - An Algorithmic and Computational Approach. Springer, Berlin, Heidelberg, 1999. doi: 10.1007/978-3-642-58393-3

[16] Watkins DS. Fundamentals of Matrix Computations. Wiley Interscience, New York, 2002.