Numerical Algebra, Control and Optimization
Vol. 15, No. 4, December 2025, pp. 1321-1339
doi:10.3934/naco.2025019

FLEXIBLE MODIFIED LSMR FOR LEAST SQUARES PROBLEMS

Mei Yang[*1] and Gul Karaduman[2] and Ren-Cang Li[3]

[1]Division of Data Science
University of Texas at Arlington, Arlington, TX 76019, USA

[2]Department of Mathematics
Karamanoglu Mehmetbey University, Karaman, 70100, Turkey

[3]Department of Mathematics
University of Texas at Arlington, Arlington, TX 76019-0408, USA

(Communicated by Jianlin Xia)

Abstract. LSMR is a widely recognized method for solving least squares problems via the double QR decomposition. Various preconditioning techniques have been explored to improve its efficiency. One issue that arises when implementing these preconditioning techniques is the need to solve two linear systems per iterative step. In this paper, to tackle this issue, among others, a modified LSMR method (MLSMR), in which only one linear system per iterative step needs to be solved instead of two, is introduced, and then it is integrated with the idea of flexible GMRES to yield a flexible MLSMR method (FMLSMR). Numerical examples are presented to demonstrate the efficiency of the proposed FMLSMR method.

1. **Introduction.** In this paper, we present a numerical method called the flexible modified LSMR method (FMLSMR), which is built upon our previous work [24], for solving the least square problem

$$\min_x \|Ax - b\|_2 , \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and either $m \geq n$ or $m < n$ is allowed. FMLSMR improves the well-known LSMR method [8] which is based on the Golub-Kahan bidiagonalization. LSMR seeks the best approximate solution in the Krylov subspace $\mathcal{K}_k\left(A^{\mathrm{T}}A, A^{\mathrm{T}}b\right)$ that minimizes $\|A^{\mathrm{T}}r\|_2$, while LSQR [18] seeks to minimize $\|r\|_2$ in the same Krylov subspace, where $r = b - Ax$. LSMR has been used for various problems, including saddle point problems, as demonstrated in recent publications [14, 15].

To speed up the convergence of LSMR, preconditioners with some desirable properties are typically used. There are various specific preconditioners, such as incomplete LU [20], incomplete QR [16], and preconditioners based on perturbed QR

2020 *Mathematics Subject Classification.* Primary: 65F08 ; Secondary: 65F20.
Key words and phrases. LSMR, flexible GMRES, FMLSMR, preconditioner, least squares problem, linear system.
The third author is supported in part by [NSF DMS-2407692].
*Corresponding author: Mei Yang.

factorizations [3]. However, determining whether a given preconditioner is suitable for a particular problem at hand is not straightforward.

In 1993, Saad proposed a flexible GMRES (FGMRES) [19] which still has an Arnoldi-like process, like original GMRES. In FGMRES, each matrix-vector multiplication involves a linear system solving that can be viewed as an application of some preconditioner that differs from one matrix-vector multiplication to another. Accelerating techniques to generate a better search space [4, 13] for GMRES can also be extended to FGMRES, resulting in variants of the method, such as in [9, 25]. These approaches aim at solving linear systems. In 2015, Morikuni and Hayami proposed an inner-outer iterative GMRES method [17] for solving (1), where the inner iterations are some stationary iterative methods like NR-SOR and NE-SOR to solve normal-equation-type equations in the form of $A^{\mathrm{T}}Av = A^{\mathrm{T}}p$ or $AA^{\mathrm{T}}v = p$.

In this paper, we will combine the two ideas above to form a flexible modified LSMR but use non-stationary methods to deal with normal-equation-type equations in the inner iteration. Previously, there are two linear systems to solve per iterative step for the right-preconditioned least squares problems, and that can be too demanding computationally sometimes. We adopt the concept of factorization-free LSQR (MLSQR) from [2] and merge the two linear systems into one of the normal-equation-type to reduce computational cost. In each iteration of FMLSMR, we apply a non-stationary method, such as MINRES, to solve this normal-equation-type equation, which implicitly yields a preconditioner in the Golub-Kahan bidiagonalization process. As the iteration goes, we obtain a different preconditioner each time, i.e. a flexible preconditioner. Our method FMLSMR retains the benefit of avoiding the decomposition of preconditioners, particularly in inverse problems [2]. The factorization-free strategy can be also found in genLSQR and genLSMR [7], which is based on the generalized Golub-Kahan bidiagonalization process proposed in [1]. In this paper, we mainly focus on accelerating LSMR rather than LSQR because LSMR exhibits better convergence properties than LSQR. However, it's not difficult to apply the same strategy of our FMLSMR to create flexible modified LSQR.

In [6], Chung and Gazzola proposed flexible LSMR (FLSMR) and flexible LSQR for ℓ_p regularization based on the flexible Golub-Kahan process, in which one upper Hessenberg and one upper triangular matrices are constructed via the Arnoldi process. Specifically, in FLSMR, two Arnoldi processes are required and so the computational cost is high for large scale problems especially when a really long Arnoldi process is needed due to orthogonalization. However, similar to LSMR, in our proposed FMLSMR, only one lower-bidiagonal matrix is generated via the Golub-Kahan process, a two-term recurrence, which keeps orthogonalization cost per step low and constant. This is a major difference between FMLSMR and FLSMR.

The rest of this paper is organized as follows. In Section 2, we present the flexible modified LSMR method based on the modified LSMR for right-preconditioned least squares problems and give some theoretical analysis of MLSMR and the flexible LSMR. In Section 3, we present the framework of the flexible MLSMR with implementation. Numerical experiments are shown in Section 4. Finally, the conclusion is drawn in Section 5.

Notation. $\mathbb{R}^{m\times n}$ is the set of all $m \times n$ real matrices, $\mathbb{R}^n = \mathbb{R}^{n\times 1}$. I_n or I (if its size is clear) is the $n \times n$ identity matrix, and e_j is its jth column. $(\cdot)^{\mathrm{T}}$ takes the transpose of a matrix or vector.

For a vector $u \in \mathbb{R}^n$, $u_{(i)}$ is its ith entry, and $\|\cdot\|_2$ is either ℓ_2-vector norm or the matrix spectral norm:

$$\|u\|_2 = \sqrt{\sum_i |u_{(i)}|^2}, \quad \|B\|_2 := \max_{v\neq 0} \frac{\|Bv\|_2}{\|v\|_2}.$$

The standard inner product $\langle u, v\rangle = u^{\mathrm{T}}v$ for vectors u and v of the same size, and in particular $\langle u, u\rangle = u^{\mathrm{T}}u = \|u\|_2^2$. Positive definite symmetric matrix $M \in \mathbb{R}^{n\times n}$ induces the M-inner product $\langle u, u\rangle_M = u^{\mathrm{T}}Mu$. Finally, the kth Krylov subspace $\mathcal{K}_k(X, u)$ of $X \in \mathbb{R}^{n\times n}$ on u is defined as

$$\mathcal{K}_k(X, u) = \operatorname{span}\{u, Xu, X^2u, \cdots, X^{k-1}u\},$$

i.e., spanned by vectors $u, Xu, X^2u, \cdots$, and $X^{k-1}u$. $\mathcal{R}(X)$ is the column space of X, spanned by its column vectors.

2. Modified LSMR and Flexible LSMR.

In this section, firstly, based on the modified LSQR in [2], the modified LSMR method (MLSMR), mentioned in [2] but never discussed in any published literature, for right-preconditioned least squares problems is introduced and some theoretical analysis of MLSMR is shown. Secondly, we review the flexible LSMR method (FLSMR) [6]. Lastly, we compare MLSMR and FLSMR from the aspects of theoretical analysis and implementation.

2.1. Modified LSMR.

The right-preconditioned least squares problem of (1) is as follows:

$$\min_{\hat{x}} \left\|AL^{-1}\hat{x} - b\right\|_2, \ x = L^{-1}\hat{x}, \tag{2}$$

where L is an invertible preconditioner. Any solution to (2) is a solution to the split-preconditioned normal equation [5]

$$L^{-\mathrm{T}}A^{\mathrm{T}}AL^{-1}\hat{x} = L^{-\mathrm{T}}A^{\mathrm{T}}b, \tag{3}$$

and vice versa.

Recall the Golub-Kahan bidiagonalization process [10], also known as the Lanczos bidiagonalization [5], for a rectangular matrix. It iteratively transforms a matrix into a lower bidiagonal matrix. With a right-preconditioner L as in (2), the Golub-Kahan bidiagonalization process for AL^{-1} on[1] b, is outlined in Algorithm 1.

If Algorithm 1 is executed without any breakdown, i.e., all $\alpha_k > 0$ and $\beta_k > 0$ for $1 \le k \le k_{\max}$, then we will have in theory

$$AL^{-1}V_k = U_{k+1}B_k, \tag{4a}$$

$$L^{-\mathrm{T}}A^{\mathrm{T}}U_{k+1} = V_kB_k^{\mathrm{T}} + \alpha_{k+1}v_{k+1}e_{k+1}^{\mathrm{T}}, \tag{4b}$$

where

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

$$V_k = [v_1, v_2, \cdots, v_k] \in \mathbb{R}^{n\times k}, \ U_k = [u_1, u_2, \cdots, u_k] \in \mathbb{R}^{m\times k}.$$

[1] Without loss of generality, we assume initial guess $x_0 = 0$ in association with the least squares problem (1); otherwise we can always reset b to $b - Ax_0$.

Algorithm 1: Golub-Kahan bidiagonalization for AL^{-1}

Input: $A \in \mathbb{R}^{m\times n}$, $L \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^m$, and tolerance ϵ;
Output: Partial bidiagonalization of $AL^{-1} \approx U_{k+1}B_kV_k^{\mathrm{T}}$.

1: $\beta_1 = \|b\|_2$, $u_1 = b/\beta_1$;
2: solve $L^{\mathrm{T}}\hat{u} = A^{\mathrm{T}}u_1$ for $\hat{u}$;
3: $\alpha_1 = \|\hat{u}\|_2$, $v_1 = \hat{u}/\alpha_1$;
4: **for** $k = 1, 2, \ldots, k_{\max}$ **do**
5: solve $L\tilde{v}_k = v_k$ for $\tilde{v}_k$;
6: $w = A\tilde{v}_k$, $q = w - \alpha_k u_k$, $\beta_{k+1} = \|q\|_2$;
7: **if** $\beta_{k+1} \le \epsilon\,\|w\|_2$ **then**
8: **break**;
9: **end if**
10: $u_{k+1} = q/\beta_{k+1}$;
11: solve $L^{\mathrm{T}}\hat{u} = A^{\mathrm{T}}u_{k+1}$ for $\hat{u}$;
12: $p = \hat{u} - \beta_{k+1}v_k$, $\alpha_{k+1} = \|p\|_2$;
13: **if** $\alpha_{k+1} \le \epsilon\,\|\hat{u}\|_2$ **then**
14: **break**;
15: **end if**
16: $v_{k+1} = p/\alpha_{k+1}$;
17: **end for**
18: **return** the last U_{k+1}, B_k, V_k.

Both V_k and U_k are orthonormal. Algorithm 1 reduces to the original Golub-Kahan process when $L = I$.

It is clear from Algorithm 1 that there are two linear systems in the form $Lv = w$ and $L^{\mathrm{T}}w = u$ to solve per for-loop. Our goal in what follows is to merge the two linear systems into one in the form $Mv = u$, where $M = L^{\mathrm{T}}L$. This does not help when L itself is structured such as being lower triangular for which case $Mv = u$ is solved directly via $L^{\mathrm{T}}w = u$ followed by $Lv = w$, but in the case when both $Lv = w$ and $L^{\mathrm{T}}w = u$ have to be solved iteratively, merging two linear systems into one per for-loop can bring big savings, not to mention the situation when only M is known but not L [7, 2].

Pre-multiplying both sides of (4b) by L^{-1}, we get

$$A\widetilde{V}_k = U_{k+1}B_k, \tag{5a}$$

$$M^{-1}A^{\mathrm{T}}U_{k+1} = \widetilde{V}_kB_k^{\mathrm{T}} + \alpha_{k+1}\tilde{v}_{k+1}e_{k+1}^{\mathrm{T}}, \tag{5b}$$

where $\widetilde{V}_k = L^{-1}V_k$. Notice $\alpha_{k+1} = \|p\|_2 = \sqrt{\langle p, p\rangle}$ at Line 12 of Algorithm 1.

It can be verified that

$$\langle p, p\rangle = \langle M^{-1}A^{\mathrm{T}}u_{k+1} - \beta_{k+1}\tilde{v}_k,\ M^{-1}A^{\mathrm{T}}u_{k+1} - \beta_{k+1}\tilde{v}_k\rangle_M,$$

upon using $M = L^{\mathrm{T}}L$. Denote by $s = M^{-1}A^{\mathrm{T}}u_{k+1} - \beta_{k+1}\tilde{v}_k$. Then, in Algorithm 1,

$$\beta_{k+1}u_{k+1} = A\tilde{v}_k - \alpha_k u_k,\ \ \alpha_{k+1} = \sqrt{\langle s, s\rangle_M},\ \ \tilde{v}_{k+1} = s/\alpha_{k+1}.$$

Define $\tilde{p}_k = M\tilde{v}_k$ and $\tilde{s} = A^{\mathrm{T}}u_{k+1} - \beta_{k+1}\tilde{p}_k$. We have

$$\begin{aligned} M^{-1}\tilde{s} &= M^{-1}A^{\mathrm{T}}u_{k+1} - \beta_{k+1}M^{-1}\tilde{p}_k \\ &= M^{-1}A^{\mathrm{T}}u_{k+1} - \beta_{k+1}\tilde{v}_k = s. \end{aligned}$$

Hence, $\alpha_{k+1}^2 = s^{\mathrm{T}}Ms = s^{\mathrm{T}}MM^{-1}\tilde{s} = \langle s, \tilde{s}\rangle$. Lines 6 - 16 can be restated as

$$\begin{aligned}
\beta_{k+1}u_{k+1} &= A\tilde{v}_k - \alpha_k u_k,\\
\tilde{s} &= A^{\mathrm{T}}u_{k+1} - \beta_{k+1}\tilde{p}_k,\\
s &= M^{-1}\tilde{s},\\
\alpha_{k+1} &= \sqrt{\langle s, \tilde{s}\rangle},\\
\tilde{v}_{k+1} &= s/\alpha_{k+1}.
\end{aligned}$$

It is understood, throughout the paper, that an expression like $s = M^{-1}\tilde{s}$ is really about solving linear system $Ms = \tilde{s}$ for s. Since $\tilde{v}_{k+1} = s/\alpha_{k+1}$ and $s = M^{-1}\tilde{s}$, we have

$$\tilde{p}_{k+1} = M\tilde{v}_{k+1} = Ms/\alpha_{k+1} = \tilde{s}/\alpha_{k+1}.$$

Therefore, a more computationally cost-effective version of the preconditioned Golub-Kahan bidiagonalization process can be summarized as

$$\begin{aligned}
\beta_{k+1}u_{k+1} &= A\tilde{v}_k - \alpha_k u_k, && \text{(6a)}\\
\tilde{p} &= A^{\mathrm{T}}u_{k+1} - \beta_{k+1}\tilde{p}_k, && \text{(6b)}\\
\tilde{v}_{k+1} &= M^{-1}\tilde{p}, && \text{(6c)}\\
\alpha_{k+1} &= \sqrt{\langle \tilde{v}_{k+1}, \tilde{p}\rangle}, && \text{(6d)}\\
\tilde{v}_{k+1} &= \tilde{v}_{k+1}/\alpha_{k+1}, && \text{(6e)}\\
\tilde{p}_{k+1} &= \tilde{p}/\alpha_{k+1}. && \text{(6f)}
\end{aligned}$$

This new formulation has only one linear system $M\tilde{v}_{k+1} = \tilde{p}$ from (6c) to solve.

It can be seen that U_{k+1} is orthonormal, while $\widetilde{V}_{k+1}$ isn't. For any $\hat{x} \in \mathcal{R}(V_k)$, $\hat{x} = V_k\hat{y}$ for some $\hat{y} \in \mathbb{R}^k$. Let $x = L^{-1}\hat{x}$ and recall $r = b - Ax$. Using (5a), we get

$$\begin{aligned}
L^{-\mathrm{T}}A^{\mathrm{T}}r &= L^{-\mathrm{T}}A^{\mathrm{T}}(b - AL^{-1}\hat{x}) = L^{-\mathrm{T}}A^{\mathrm{T}}(b - AL^{-1}V_k\hat{y})\\
&= L^{-\mathrm{T}}A^{\mathrm{T}}(b - U_{k+1}B_k\hat{y}) = L^{-\mathrm{T}}A^{\mathrm{T}}b - L^{-\mathrm{T}}A^{\mathrm{T}}U_{k+1}B_k\hat{y}\\
&= \underline{\beta}_1 v_1 - V_{k+1}\begin{bmatrix} B_k^{\mathrm{T}}B_k \\ \underline{\beta}_{k+1}e_k^{\mathrm{T}}\end{bmatrix}\hat{y}\\
&= V_{k+1}\left(\underline{\beta}_1 e_1 - \begin{bmatrix} B_k^{\mathrm{T}}B_k \\ \underline{\beta}_{k+1}e_k^{\mathrm{T}}\end{bmatrix}\hat{y}\right),
\end{aligned}$$

where $\underline{\beta}_1 = \alpha_1\beta_1$ and $\underline{\beta}_{k+1} = \alpha_{k+1}\beta_{k+1}$. For the preconditioned least squares problem (2), LSMR seeks an approximate solution $\hat{x}_k \in \mathcal{R}(V_k)$ that minimizes $\|L^{-\mathrm{T}}A^{\mathrm{T}}r\|_2$ of the preconditioned normal equation (3) over all $x \in \mathcal{R}(\tilde{V}_k)$. Because V_{k+1} is orthonormal, for $\hat{x} = V_k\hat{y}$ we have

$$\min_{\hat{x}\in\mathcal{R}(V_k)} \left\|L^{-\mathrm{T}}A^{\mathrm{T}}r\right\|_2 = \min_{\hat{y}} \left\|\underline{\beta}_1 e_1 - \begin{bmatrix} B_k^{\mathrm{T}}B_k \\ \underline{\beta}_{k+1}e_k^{\mathrm{T}}\end{bmatrix}\hat{y}\right\|_2, \tag{7}$$

and hence $\hat{x}_k = V_k\hat{y}_k$, where

$$\hat{y}_k = \arg\min_{\hat{y}} \left\|\underline{\beta}_1 e_1 - \begin{bmatrix} B_k^{\mathrm{T}}B_k \\ \underline{\beta}_{k+1}e_k^{\mathrm{T}}\end{bmatrix}\hat{y}\right\|_2 \tag{8}$$

which is a least squares problem that can be solved by the double QR factorization as in LSMR. Finally, the solution of the original least squares problem can then be

approximated by

$$x_k = L^{-1}\hat{x}_k = L^{-1}V_k\hat{y}_k = \widetilde{V}_k\hat{y}_k. \tag{9}$$

Putting all together, we have established the modified LSMR (MLSMR) method outlined in Algorithm 2. It should be noted that Lines 8 – 19 use the double QR decomposition, taken from LSMR [8].

If combining Algorithm 2.1 with the steps of the double-QR decomposition as in LSMR, we can obtain LSMR method for right-preconditioned least squares problems. Theoretically, LSMR for right-preconditioned problems is equivalent to Algorithm 2, but different in implementations. In LSMR for right-preconditioned problems, two linear systems in the forms of $Lv = w$ and $L^{\mathrm{T}}w = u$ need to be solved. However, MLSMR is more efficient because at each iteration, x_k is updated directly and only one linear system at Line 5 of Algorithm 2 is solved.

By (9), at the kth step of MLSMR we have

$$x_k \in \mathcal{K}_k(M^{-1}A^{\mathrm{T}}A, M^{-1}A^{\mathrm{T}}b) = L^{-1}\mathcal{K}_k(L^{-\mathrm{T}}A^{\mathrm{T}}AL^{-1}, L^{-\mathrm{T}}A^{\mathrm{T}}b). \tag{10}$$

According to the Golub-Kahan bidiagonalization process, we know V_{k+1} are orthonormal, i.e., $V_{k+1}^{\mathrm{T}}V_{k+1} = I_{k+1}$. Since $\widetilde{V}_{k+1} = L^{-1}V_{k+1}$, we get

$$I_{k+1} = V_{k+1}^{\mathrm{T}}V_{k+1} = \widetilde{V}_{k+1}^{\mathrm{T}}L^{\mathrm{T}}L\widetilde{V}_{k+1} = \widetilde{V}_{k+1}^{\mathrm{T}}M\widetilde{V}_{k+1} = \langle\widetilde{V}_{k+1}, \widetilde{V}_{k+1}\rangle_M.$$

That implies $\widetilde{V}_{k+1}$ is M-orthonormal.

When A has full column rank, (1) has a unique solution. Otherwise, there are infinitely many solutions that yield the minimum value of $\|Ax - b\|_2$. For the case that $A^{\mathrm{T}}Ax = A^{\mathrm{T}}b$ with singular $A^{\mathrm{T}}A$, it has been proved [8] that both LSQR and LSMR return the same minimum-norm solution to the least squares problem (1) at convergence. We state these conclusions as follows.

Theorem 2.1 ([8, Corollary 4.3])**.** *At convergence, LSMR returns the minimum-norm solution to* (1)*.*

Corollary 2.2. *Suppose that $\hat{x}_*$ is the solution to* (2) *obtained via MLSMR at convergence. Then $\hat{x}_*$ is the minimum-norm solution to* (2) *and $x_* = L^{-1}\hat{x}_*$ is the minimum-M-norm solution to* (1)*.*

Proof. Since $\hat{x}_*$ is the solution to (2) obtained via MLSMR, $\hat{x}_*$ is the minimum-norm solution to (2) according to Theorem 2.1. Because $\hat{x}_* = Lx_*$, we have $\|\hat{x}_*\|_2 = \|x_*\|_M$. This means that x_* is the minimum-M-norm solution to (1). □

Now we comment on how to develop the modified LSQR in a similar way. With the preconditioned Golub-Kahan bidiagonalization process (6), we can obtain the modified LSQR method (MLSQR), i.e., the factorization-free preconditioned LSQR in [2], by minimizing $\|r\|_2$ over $x \in \mathcal{R}(\widetilde{V}_k)$. This yields $x_k = \widetilde{V}_k\hat{y}_k^{\mathrm{MLSQR}}$, where

$$\hat{y}_k^{\mathrm{MLSQR}} = \operatorname*{argmin}_{\hat{y}} \|\beta_1 e_1 - B_k\hat{y}\|_2. \tag{11}$$

Given a nonsingular preconditioner L, $\hat{x} = Lx$ is the solution to (3), where x is the solution to the normal equation

$$A^{\mathrm{T}}Ax = A^{\mathrm{T}}b. \tag{12}$$

Theorem 2.3 ([8, Theorem 4.2])**.** *At convergence, LSQR returns the minimum-norm solution to* (1)*.*

Algorithm 2: Modified LSMR (MLSMR)

Input: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $M \in \mathbb{R}^{n\times n}$, and tolerance ϵ;
Output: Approximate solution to (1).

$\beta_1 = \|b\|_2, u_1 = b/\beta_1, \tilde{p} = A^{\mathrm{T}}u_1, \tilde{v}_1 = M^{-1}\tilde{p}, \alpha_1 = \langle \tilde{v}_1, \tilde{p}\rangle^{1/2}, \tilde{p} = \tilde{p}/\alpha_1, \tilde{v}_1 = \tilde{v}_1/\alpha_1,$
$\underline{\alpha}_1 = \alpha_1$, $\underline{\xi}_1 = \alpha_1\beta_1$, $\rho_0 = \underline{\rho}_0 = \underline{c}_0 = 1$, $\underline{s}_0 = 0, h_1 = \tilde{v}_1$, $\underline{h}_0 = 0$, $x_0 = 0$;
for $k = 1, 2, \ldots, k_{\max}$ **do**
  $\hat{u}_{k+1} = A\tilde{v}_k - \alpha_k u_k, \beta_{k+1} = \|\hat{u}_{k+1}\|_2, u_{k+1} = \hat{u}_{k+1}/\beta_{k+1}$;
  $\tilde{p} = A^{\mathrm{T}}u_{k+1} - \beta_{k+1}\tilde{p}$;
  $\tilde{v}_{k+1} = M^{-1}\tilde{p}$, $\alpha_{k+1} = \langle \tilde{v}_{k+1}, \tilde{p}\rangle^{1/2}$;
  $\tilde{p} = \tilde{p}/\alpha_{k+1}$;
  $\tilde{v}_{k+1} = \tilde{v}_{k+1}/\alpha_{k+1}$;
  $\rho_k = (\underline{\alpha}_k^2 + \beta_{k+1}^2)^{1/2}$;
  $c_k = \underline{\alpha}_k/\rho_k$;
  $s_k = \beta_{k+1}/\rho_k$, $\theta_{k+1} = s_k\alpha_k$;
  $\underline{\alpha}_{k+1} = c_k\alpha_{k+1}$;
  $\underline{\theta}_k = \underline{s}_{k-1}\rho_k$;
  $\underline{\rho}_k = ((\underline{c}_{k-1}\rho_k)^2 + \theta_{k+1}^2)^{1/2}$;
  $\underline{c}_k = \underline{c}_{k-1}\rho_k/\underline{\rho}_k$;
  $\underline{s}_k = \theta_{k+1}/\underline{\rho}_k$;
  $\xi_k = \underline{c}_k\underline{\xi}_k$, $\underline{\xi}_{k+1} = -\underline{s}_k\underline{\xi}_k$;
  $\underline{h}_k = h_k - (\underline{\theta}_k\rho_k/(\rho_{k-1}\underline{\rho}_{k-1}))\underline{h}_{k-1}$;
  $x_k = x_{k-1} + (\xi_k/(\rho_k\underline{\rho}_k))\underline{h}_k$;
  $h_{k+1} = \tilde{v}_{k+1} - (\theta_{k+1}/\rho_k)h_k$;
  $r_k = b - Ax_k$;
  if $\|A^{\mathrm{T}}r_k\|_2 \le \epsilon\,\|A\|_2(\|b\|_2 + \|A\|_2\|x_k\|_2)$ **then**
    break;
  end if
end for
return the last x_k.

Corollary 2.4. *Suppose that $\hat{x}_*^{\mathrm{MLSQR}}$ is the solution to (2) via MLSQR at convergence. Then $\hat{x}_*^{\mathrm{MLSQR}}$ is the minimum-norm solution to (2) and $x_*^{\mathrm{MLSQR}} = L^{-1}\hat{x}_*^{\mathrm{MLSQR}}$ is the minimum-M-norm solution to (1).*

Proof. Since $\hat{x}_*^{\mathrm{MLSQR}}$ is the solution to (2) obtained via MLSQR at convergence, $\hat{x}_*^{\mathrm{MLSQR}}$ is the minimum-norm solution to (2) according to Theorem 2.3. Because $\hat{x}_*^{\mathrm{MLSQR}} = Lx_*^{\mathrm{MLSQR}}$, we have $\|\hat{x}_*^{\mathrm{MLSQR}}\|_2 = \|x_*^{\mathrm{MLSQR}}\|_M$. This means that x_*^{MLSQR} is the minimum-M-norm solution to (1). □

Remark 2.5. MLSMR minimizes the residual of the normal equation (3) by solving the subproblem (8). If $\underline{\beta}_{k+1} = 0$ for some k, then $\alpha_{k+1} = 0$ or $\beta_{k+1} = 0$. In this case, (8) becomes $\min_{\hat{y}} \|\underline{\beta}_1 e_1 - B_k^{\mathrm{T}}B_k\hat{y}\|_2$ and $B_k^{\mathrm{T}}B_k\hat{y}_k = \underline{\beta}_1 e_1$ because B_k has full rank. Hence, at convergence MLSMR returns the same solution as by MLSQR. When $L = I$, MLSMR reduces to LSMR.

2.2. **Flexible LSMR Method.** In [6], Chung and Gazzola proposed two flexible Krylov subspace methods for ℓ_p regularization: the flexible LSQR (FLSQR) and

the flexible LSMR (FLSMR). The basic idea is to apply FGMRES to construct a flexible variant of the Golub-Kahan process (FGK). The framework of FGK is shown in Algorithm 3, where N_k is the preconditioner at the kth iteration determined at runtime.

Algorithm 3: Flexible Golub-Kahan Process (FGK)

Input: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and tolerance ϵ;
Output: Z_k, $\widetilde{U}_{k+1}$, W_{k+1}, T_{k+1}, and P_k.

$\tilde{\beta}_1 = \|b\|_2, \tilde{u}_1 = b/\tilde{\beta}_1, w_0 = 0$;
for $k = 1, 2, \ldots, k_{\max}$ **do**
  $v = A^{\mathrm{T}}\tilde{u}_k$, $\xi = \|v\|_2$;
  for $j = 1, 2, \ldots, k-1$ **do**
    $t_{j,k} = v^{\mathrm{T}} w_j$;
    $v = v - t_{j,k} w_j$;
  end for
  $t_{k,k} = \|v\|_2$;
  if $t_{k,k} \le \epsilon\xi$ **then**
    break;
  end if
  $w_k = v/t_{k,k}$;
  solve $N_k z_k = w_k$ for z_k where $N_k \in \mathbb{R}^{n\times n}$ is some varying preconditioner determined at runtime;
  $v = Az_k$, $\xi = \|v\|_2$;
  for $j = 1, \ldots, k$ **do**
    $p_{j,k} = v^{\mathrm{T}}\tilde{u}_j$;
    $v = v - p_{j,k}\tilde{u}_j$;
  end for
  $p_{k+1,k} = \|v\|_2$;
  if $p_{k+1,k} \le \epsilon\xi$ **then**
    break;
  end if
  $\tilde{u}_{k+1} = v/p_{k+1,k}$;
end for
return the last $Z_k, \widetilde{U}_{k+1}, W_{k+1}, T_{k+1}, P_k$.

After the kth iteration, we have

$$AZ_k = \widetilde{U}_{k+1}P_k, \qquad A^{\mathrm{T}}\widetilde{U}_{k+1} = W_{k+1}T_{k+1}, \tag{13}$$

where

$$\begin{aligned} Z_k &= [z_1, z_2, \ldots, z_k] = [N_1^{-1}w_1, N_2^{-1}w_2, \ldots, N_k^{-1}w_k] \in \mathbb{R}^{n\times k},\\ \widetilde{U}_{k+1} &= [\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_{k+1}] \in \mathbb{R}^{m\times(k+1)},\ W_{k+1} = [w_1, w_2, \ldots, w_{k+1}] \in \mathbb{R}^{n\times(k+1)},\\ P_k &= [p_{i,j}] \in \mathbb{R}^{(k+1)\times k},\ T_{k+1} = [t_{i,j}] \in \mathbb{R}^{(k+1)\times(k+1)}. \end{aligned}$$

It can be seen that both $\widetilde{U}_{k+1}$ and W_{k+1} are orthonormal, and P_k and T_{k+1} are upper Hessenberg and upper triangular matrices, respectively. The kth approximate

solution of FLSMR is given by $x_k^{\mathrm{FLSMR}} = Z_k y_k^{\mathrm{FLSMR}}$, where

$$y_k^{\mathrm{FLSMR}} = \underset{y}{\operatorname{argmin}} \|\beta_1 t_{1,1} e_1 - T_{k+1} P_k y\|_2. \tag{14}$$

As to the residual of the normal equation (12) at x_k^{FLSMR}, we have

$$\begin{aligned} A^{\mathrm{T}} r_k &\equiv A^{\mathrm{T}}(b - A x_k^{\mathrm{FLSMR}}) \\ &= \beta_1 t_{1,1} w_1 - A^{\mathrm{T}} A Z_k y_k^{\mathrm{FLSMR}} \\ &= \beta_1 t_{1,1} w_1 - W_{k+1} T_{k+1} P_k y_k^{\mathrm{FLSMR}} \\ &= W_{k+1} \left(\beta_1 t_{1,1} e_1 - T_{k+1} P_k y_k^{\mathrm{FLSMR}}\right). \end{aligned}$$

Similarly for FLSQR, $x_k^{\mathrm{FLSQR}} = Z_k y_k^{\mathrm{FLSQR}}$, where

$$y_k^{\mathrm{FLSQR}} = \underset{y}{\operatorname{argmin}} \|\beta_1 e_1 - P_k y\|_2.$$

It is shown [6] that x_k^{FLSQR} obtained at the kth step minimizes the residual norm $\|b - Ax\|_2$ over $x \in \mathcal{R}(Z_k)$, while x_k^{FLSMR} minimizes $\|A^{\mathrm{T}}(b - Ax)\|_2$ over $x \in \mathcal{R}(Z_k)$. This is the theoretical difference between FLSQR and FLSMR. We outline FLSMR in Algorithm 4.

Algorithm 4: Flexible LSMR

Input: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $N_k \in \mathbb{R}^{n\times n}$ for each $k \geq 1$, and tolerance ϵ;
Output: Approximate solution x_k to (1).

1: $\tilde{\beta}_1 = \|b\|_2, \tilde{u}_1 = b/\tilde{\beta}_1, w_0 = 0$;
2: **for** $k = 1, 2, \ldots, k_{\max}$ **do**
3: Execute Line 3 - 23 in Algorithm 3;
4: solve for y_k^{FLSMR} from (14)
5: $x_k = Z_k y_k^{\mathrm{FLSMR}}$, $r_k = b - Ax_k$;
6: **if** $\|A^{\mathrm{T}} r_k\|_2 \leq \epsilon \|A\|_2(\|b\|_2 + \|A\|_2\|x_k\|_2)$ **then**
7: **break**;
8: **end if**
9: **end for**
10: **return** the last x_k.

2.3. **A Brief Comparison of FLSMR and MLSMR.** In this subsection, we briefly compare FLSMR with MLSMR. At appearance, FLSMR possibly employs different preconditioners, i.e., different N_k at Line 13 of Algorithm 3 for each k, while MLSMR uses the same preconditioner, i.e., M in Algorithm 2. When all N_k are taken to be the same as M, we have the following result.

Theorem 2.6. *If the preconditioners N_k in Algorithm 3 are the same as the preconditioner M in Algorithm 2, then the solutions by FLSMR and MLSMR satisfy*

$$x_k^{\mathrm{FLSMR}} = \underset{x}{\operatorname{argmin}} \|A^{\mathrm{T}}(b - Ax)\|_2 \quad \textit{over} \quad x \in \mathcal{K}_k(M^{-1}A^{\mathrm{T}}A, M^{-1}A^{\mathrm{T}}b), \tag{15}$$

$$x_k^{\mathrm{MLSMR}} = \underset{x}{\operatorname{argmin}} \|A^{\mathrm{T}}(b - Ax)\|_{M^{-1}} \quad \textit{over} \quad x \in \mathcal{K}_k(M^{-1}A^{\mathrm{T}}A, M^{-1}A^{\mathrm{T}}b), \tag{16}$$

respectively.

Proof. According to (13), we have

$$A^{\mathrm{T}}AM^{-1}W_k = W_{k+1}T_{k+1}P_k,$$

where W_k is orthonormal and $T_{k+1}P_k$ is upper Hessenberg. Hence, FLSMR is exactly the same as GMRES applied to the right-preconditioned linear system

$$A^{\mathrm{T}}AM^{-1}\hat{x} = A^{\mathrm{T}}b, \quad x = M^{-1}\hat{x}.$$

Therefore, at the kth iteration, we have

$$x_k^{\mathrm{FLSMR}} = M^{-1}\hat{x}_k \in M^{-1}\mathcal{K}_k(A^{\mathrm{T}}AM^{-1}, A^{\mathrm{T}}b) = \mathcal{K}_k(M^{-1}A^{\mathrm{T}}A, M^{-1}A^{\mathrm{T}}b).$$

According to (10), both FLSMR and MLSMR search their kth approximate solutions over

$$x \in \mathcal{R}(Z_k) = \mathcal{K}_k(M^{-1}A^{\mathrm{T}}A, M^{-1}A^{\mathrm{T}}b).$$

This implies (15) because x_k^{FLSMR} minimizes $\|A^{\mathrm{T}}(b - Ax)\|_2$ over $\mathcal{R}(Z_k)$. On the other hand, if we have $M = L^{\mathrm{T}}L$, then by (7), we know that x_k^{MLSMR} can be written as $x_k^{\mathrm{MLSMR}} = L^{-1}\hat{x}_k$, where $\hat{x}_k$ is the kth approximate solution obtained from LSMR for the right-preconditioned least square problem, i.e., $\hat{x}_k$ satisfies

$$\hat{x}_k = \operatorname*{argmin}_{\hat{x}\in\mathcal{K}_k(L^{-\mathrm{T}}A^{\mathrm{T}}AL^{-1}, L^{-\mathrm{T}}A^{\mathrm{T}}b)} \|L^{-\mathrm{T}}A^{\mathrm{T}}(b - AL^{-1}\hat{x})\|_2.$$

Since $\|L^{-\mathrm{T}}A^{\mathrm{T}}(b - AL^{-1}\hat{x})\|_2^2 = \|A^{\mathrm{T}}(b - Ax)\|_{M^{-1}}^2$ and

$$L^{-1}\mathcal{K}_k(L^{-\mathrm{T}}A^{\mathrm{T}}AL^{-1}, L^{-\mathrm{T}}A^{\mathrm{T}}b) = \mathcal{K}_k(M^{-1}A^{\mathrm{T}}A, M^{-1}A^{\mathrm{T}}b),$$

we conclude

$$x_k^{\mathrm{MLSMR}} = \operatorname*{argmin}_{x\in\mathcal{R}(Z_k)} \|A^{\mathrm{T}}(b - Ax)\|_{M^{-1}},$$

as was to be shown. □

Theorem 2.6 states that FLSMR with a fixed preconditioner and MLSMR solve two different optimization problems over the same search space.

As we have discussed in the proof of Theorem 2.6, if $M = L^{\mathrm{T}}L$ is known, the kth approximate solution x_k obtained from MLSMR satisfies

$$x_k = L^{-1}\hat{x}_k, \quad \hat{x}_k = \operatorname*{argmin}_{\hat{x}\in\mathcal{K}_k(L^{-\mathrm{T}}A^{\mathrm{T}}AL^{-1}, L^{-\mathrm{T}}A^{\mathrm{T}}b)} \|L^{-\mathrm{T}}A^{\mathrm{T}}r\|_2, \tag{17}$$

where $r = b - AL^{-1}\hat{x}$, $\hat{x}_k$ is the kth approximate solution to (2), and

$$\mathcal{K}_k(L^{-\mathrm{T}}A^{\mathrm{T}}AL^{-1}) = \mathcal{R}(V_k)$$

in which V_k is orthonormal and obtained in Algorithm 1. Thus, we can always rewrite $\hat{x} = V_k\hat{y}$ for some $\hat{y} \in \mathbb{R}^k$. Based on the previous discussion about MLSMR in Subsection 2.1, we know that

$$L^{-\mathrm{T}}A^{\mathrm{T}}r = V_{k+1}\left(\alpha_1\beta_1 e_1 - \begin{bmatrix} B_k^{\mathrm{T}}B_k \\ \alpha_{k+1}\beta_{k+1}e_k^{\mathrm{T}} \end{bmatrix}\hat{y}\right). \tag{18}$$

Denote by $D_{k+1} = \begin{bmatrix} B_k^{\mathrm{T}}B_k \\ \alpha_{k+1}\beta_{k+1}e_k^{\mathrm{T}} \end{bmatrix}$. Multiplying both sides of (18) by L^{-1}, we get

$$M^{-1}A^{\mathrm{T}}(b - AL^{-1}\hat{x}) = \widetilde{V}_{k+1}\left(\alpha_1\beta_1 e_1 - D_{k+1}\hat{y}\right),$$

where $\widetilde{V}_{k+1} = L^{-1}V_{k+1}$, which is defined in Subsection 2.1. Thus,

$$A^{\mathrm{T}}r = M\widetilde{V}_{k+1}\left(\alpha_1\beta_1 e_1 - D_{k+1}\hat{y}\right). \tag{19}$$

Let $M\widetilde{V}_{k+1} = \check{Q}_{k+1}\check{R}_{k+1}$ be the QR decomposition of $M\widetilde{V}_{k+1}$, where $\check{Q}_{k+1} \in \mathbb{R}^{n\times(k+1)}$ is orthonormal and $\check{R}_{k+1} \in \mathbb{R}^{(k+1)\times(k+1)}$ is upper triangular. We then rewrite (19) as

$$A^{\mathrm{T}}r = \check{Q}_{k+1}\check{R}_{k+1}\left(\alpha_1\beta_1 e_1 - D_{k+1}\hat{y}\right).$$

Naturally, this leads to a new way for the original least squares problem. Namely, instead of (17), we seek an approximation as follows:

$$\tilde{x}_k = \underset{x\in\mathcal{K}_k(M^{-1}A^{\mathrm{T}}A, M^{-1}A^{\mathrm{T}}b)}{\operatorname{argmin}} \|A^{\mathrm{T}}r\|_2 = \widetilde{V}_k\hat{z}_k \tag{20}$$

where

$$\hat{z}_k = \underset{\hat{y}}{\operatorname{argmin}} \left\|\alpha_1\beta_1\check{R}_{k+1}e_1 - \check{R}_{k+1}D_{k+1}\hat{y}\right\|_2.$$

Notice that $\alpha_1\check{R}_{k+1}e_1 = \alpha_1 r_{1,1}e_1 = \|A^{\mathrm{T}}u_1\|_2 e_1 = t_{1,1}e_1$, and $\check{R}_{k+1}D_{k+1}$ is upper Hessenberg, we get

$$\hat{z}_k = \underset{\hat{y}}{\operatorname{argmin}} \left\|\beta_1 t_{1,1}e_1 - \check{R}_{k+1}D_{k+1}\hat{y}\right\|_2, \tag{21}$$

which takes the same form as FLSMR's reduced problem (14). With the above analysis, we can get a variant of MLSMR, which is the preconditioned Golub-Kahan process (4) followed by solving (21). We can see that this new variant of MLSMR is equivalent to FLSMR since they both minimize the same objective over the same search space, $x \in \mathcal{R}(\widetilde{V}_k) = \mathcal{R}(Z_k) = \mathcal{K}_k(M^{-1}A^{\mathrm{T}}A, M^{-1}A^{\mathrm{T}}b)$.

3. Flexible modified LSMR.

In MLSMR (Algorithm 2), the most extreme but impractical preconditioner is $M = A^{\mathrm{T}}A$, with which x_1 is the exact solution to the normal equation (22). However, that is not feasible in practice for large n. Some approximate inverse of $A^{\mathrm{T}}A$ has to be used, or, equivalently, to solve approximately

$$A^{\mathrm{T}}A\tilde{v} = \tilde{p}, \tag{22}$$

for $\tilde{v}$ at Lines 1 and 5 in Algorithm 2. By doing so, we implicitly determine some approximations, likely unknown but exist, of $(A^{\mathrm{T}}A)^{-1}$ in the inner iterations. Using stationary methods such as the Jacobi and SOR-type methods to solve (22) yields preconditioners M that remains the same for each inner iteration, but applying non-stationary methods like CG or MINRES dynamically selects varying preconditioners, i.e., M changes as $\tilde{p}$ changes from one iteration to the next.

The latter leads to our new approach, namely the *flexible modified LSMR method* (FMLSMR), as outlined in Algorithm 5. In our numerical tests later in Section 4, MINRES is used to solve (22) for FMLSMR.

Symbolically, we may write $\tilde{v}_1 = M_1^{-1}\tilde{p}_1$ at Line 2 and $\tilde{v}_{k+1} = M_{k+1}^{-1}\tilde{p}_{k+1}$ at Line 8, where M_1 and M_{k+1} are dependent of vectors $\tilde{p}_1$ and $\tilde{p}_{k+1}$, respectively, and of computed approximations $\tilde{v}_1$ and $\tilde{v}_{k+1}$, respectively, as well. Exactly, what these M_k are is not important as far as executing Algorithm 5 is concerned.

After the kth step, the approximate solution x_k is sought in $\mathcal{R}(\widetilde{V}_k)$, and we have

$$\begin{aligned} A\widetilde{V}_k &= U_{k+1}B_k, \\ A^{\mathrm{T}}U_{k+1} &= [M_1\tilde{v}_1, M_2\tilde{v}_2, \cdots, M_{k+1}\tilde{v}_{k+1}] \begin{bmatrix} B_k^{\mathrm{T}} \\ \alpha_{k+1}\beta_{k+1}e_{k+1}^{\mathrm{T}} \end{bmatrix}. \end{aligned}$$

Let $x_k = \widetilde{V}_k\tilde{y}_k$. $\|A^{\mathrm{T}}r_k\|_2$ can be expressed as follows:

$$\left\|A^{\mathrm{T}}r_k\right\|_2 = \left\|A^{\mathrm{T}}b - A^{\mathrm{T}}Ax_k\right\|_2 = \left\|A^{\mathrm{T}}b - A^{\mathrm{T}}U_{k+1}B_k\tilde{y}_k\right\|_2$$

Algorithm 5: Flexible MLSMR

Input: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and tolerance ϵ;
Output: Approximate solution to (1).

1: $\beta_1 = \|b\|_2, u_1 = b/\beta_1, \tilde{p}_1 = A^{\mathrm{T}}u_1$;
2: solve $A^TA\tilde{v}_1 = \tilde{p}_1$ approximately for $\tilde{v}_1$;
3: $\alpha_1 = \langle \tilde{v}_1, \tilde{p}_1\rangle^{1/2}$, $\hat{p}_1 = \tilde{p}_1/\alpha_1, \tilde{v}_1 = \tilde{v}_1/\alpha_1$;
4: $\underline{\alpha}_1 = \alpha_1$, $\underline{\xi}_1 = \alpha_1\beta_1$, $\rho_0 = \underline{\rho}_0 = \underline{c}_0 = 1$, $\underline{s}_0 = 0, h_1 = \tilde{v}_1$, $\underline{h}_0 = 0$, $x_0 = 0$;
5: **for** $k = 1, 2, \ldots, k_{\max}$ **do**
6: $\quad \hat{u}_{k+1} = A\tilde{v}_k - \alpha_k u_k, \beta_{k+1} = \|\hat{u}_{k+1}\|_2, u_{k+1} = \hat{u}_{k+1}/\beta_{k+1}$;
7: $\quad \tilde{p}_{k+1} = A^{\mathrm{T}}u_{k+1} - \beta_{k+1}\hat{p}_k$;
8: $\quad$ solve $A^TA\tilde{v}_{k+1} = \tilde{p}_{k+1}$ approximately for $\tilde{v}_{k+1}$;
9: $\quad \alpha_{k+1} = \langle \tilde{v}_{k+1}, \tilde{p}_{k+1}\rangle^{1/2}$;
10: $\quad \hat{p}_{k+1} = \tilde{p}_{k+1}/\alpha_{k+1}$, $\tilde{v}_{k+1} = \tilde{v}_{k+1}/\alpha_{k+1}$;
11: $\quad$ Lines 8 – 23 of Algorithm 2;
12: **end for**
13: **return** the last x_k.

$$
\begin{aligned}
&= \left\| A^{\mathrm{T}}b - [M_1\tilde{v}_1, \cdots, M_{k+1}\tilde{v}_{k+1}] \begin{bmatrix} B_k^{\mathrm{T}}B_k \\ \alpha_{k+1}\beta_{k+1}e_k^{\mathrm{T}} \end{bmatrix} \tilde{y}_k \right\|_2 \\
&= \left\| [M_1\tilde{v}_1, \cdots, M_{k+1}\tilde{v}_{k+1}] \left(\alpha_1\beta_1 e_1 - \begin{bmatrix} B_k^{\mathrm{T}}B_k \\ \alpha_{k+1}\beta_{k+1}e_k^{\mathrm{T}} \end{bmatrix} \tilde{y}_k\right) \right\|_2 \\
&\le \|[\hat{p}_1, \cdots, \hat{p}_{k+1}]\|_2 \left\| \alpha_1\beta_1 e_1 - \begin{bmatrix} B_k^{\mathrm{T}}B_k \\ \alpha_{k+1}\beta_{k+1}e_k^{\mathrm{T}} \end{bmatrix} \tilde{y}_k \right\|_2,
\end{aligned}
$$

where

$$
\tilde{y}_k = \operatorname*{argmin}_{\tilde{y}} \left\| \alpha_1\beta_1 e_1 - \begin{bmatrix} B_k^{\mathrm{T}}B_k \\ \alpha_{k+1}\beta_{k+1}e_k^{\mathrm{T}} \end{bmatrix} \tilde{y} \right\|_2.
$$

Denote $[\hat{p}_1, \cdots, \hat{p}_{k+1}]$ by G_{k+1}. Hence, if $\|G_{k+1}\|_2$ is not too large, x_k can be a good approximate solution to the original least squares problem (1). Because G_{k+1} is not orthonormal, the residuals by FMLSMR may not be monotonically decreasing, unlike in FLSMR. However, the computation cost of FMLSMR is less than that of FLSMR because in FLSMR, a Gram-Schmit orthogonalization step is conducted twice at Lines 4 – 7 and Lines 15 – 18 of Algorithm 3.

As discussed before, we can choose any non-stationary method to solve (22), and it's not easy to find some general theoretical bound on $\|G_{k+1}\|_2$. We leave this problem for the future research but show some numerical results in the next section to demonstrate that $\|G_{k+1}\|_2$ is usually modest.

In numerical experiments, at each iteration of our method, the inner solver is MINRES with the ℓ-step Lanczos, where ℓ is a small number compared to the size of coefficient matrix. The number in Lanczos iterations can be adjusted based on some tolerance setting for the inner iteration. But it may take many inner iterations and have stagnation [21] if the tolerance is not appropriately set. From [21] and [11], we know that the inner-outer iteration method often converges for fairly large thresholds in the inner iteration. Ideally, ℓ should be varied for best performance, but finding a best strategy is difficult. For the moment, we simply choose a small fixed ℓ in our implementation.

4. **Numerical experiments.** In this section, we perform numerical tests to demonstrate the advantage of our proposed method FMLSMR.

Firstly, we compare the computational cost of LSMR, FLSMR, and FMSLMR. Table 1 lists the numbers of flops of all methods where the ℓ-step MINRES is used in the inner solver for both FLSMR and FMLSMR. The symbol "MV" denotes the number of flops required for a single matrix-vector multiplication with $A \in \mathbb{R}^{m\times n}$, which is taken to be twice the number of nonzero entries in A. Only the dominant terms are included in flops in each iteration, which are matrix-vector multiplications, solutions of the inner linear systems and vector-vector operations. Both the computational costs of FLSMR and FMLSMR are more than LSMR because of their inner iterations, and at the same time, for the same inner solver, the computational cost of FMLSMR is less than that of FLSMR. Later we will also report CPU times for all examples.

TABLE 1. Flops of LSMR Variants

k-step of LSMR	$(3k)(\texttt{MV}) + (8n + 2m)k$
k-step of FLSMR(ℓ)	$(5 + 2\ell)k(\texttt{MV}) + 2/3(n + m)k^3 + (n^2 + m^2)k^2 +$ $(13n/3 + m/3)k + 6n\ell k + 4\ell^2 k$
k-step of FMLSMR (ℓ)	$(5 + 2\ell)k(\texttt{MV}) + (12n + 2m)k + 6n\ell k + 4\ell^2 k$

Secondly, we compare the storage requirements for the three methods after the kth iteration in Table 2. In the table, the 2nd and 3rd columns show the numbers of stored vectors at the kth iteration. The last column displays the stored matrices for each methods. FLSMR is the only one requiring matrix storage, which increases quadratically in k. However, FMLSMR consumes the same amount of storage as LSMR, which is far less than that of FLSMR. The results of our numerical examples will confirm this advantage of FMLSMR over FLSMR.

TABLE 2. Storage of LSMR Variants (besides A and b)

	# of vectors in $\mathbb{R}^m$	# of vectors in $\mathbb{R}^n$	Matrices
k-step of LSMR	3	5	
k-step of FLSMR	$k + 4$	$k + 3$	T_{k+1}, P_{k+1}
k-step of FMLSMR	3	7	

Third, we report our numerical results on 8 testing problems which are drawn from the SuiteSparse Matrix Collection[2] and Matrix Market[3].

Among the problems, `biplane-9` comes with a right-hand side b. A random vector b is generated by `rand` for each of the rest of problems. Table 3 lists some of their important characteristics, including the matrix size m and n, the number `nnz` of nonzero entries in A, and the sparsity `nnz`$/(mn)$. These are representatives of many other problems from the collections we have tested. All tests are done by MATLAB (version R2020b) on a Mac PC with 2.7 GHz Intel Core i7 and 16GB memory.

[2]https://sparse.tamu.edu/
[3]https://math.nist.gov/MatrixMarket

TABLE 3. Testing Matrices

ID	matrix	m	n	nnz	sparsity
1	well1850	1850	712	8755	1.45×10^{-2}
2	cat_ears_3_4	13271	5226	39592	5.7087×10^{-4}
3	delaunay_n16	65536	65536	393150	9.1537×10^{-5}
4	biplane-9	21701	21701	84076	1.7853×10^{-4}
5	flower_7_4	67593	27693	202218	1.0803×10^{-4}
6	crack	10240	10240	60760	5.7936×10^{-4}
7	fe_body	45087	45087	327468	1.6109×10^{-4}
8	stufe-10	24010	24010	92828	1.6103×10^{-4}

The stopping criteria are either

$$\text{NRes} = \frac{\|A^{\mathrm{T}}(Ax-b)\|_2}{\|A\|_1(\|A\|_1\|x\|_2+\|b\|_2)} \le 10^{-12}, \tag{23}$$

on normalized residual (NRes) or the number of iterations reaches 10^5, where using matrix ℓ_1-norm $\|A\|_1$ is for its easiness in computation. Another assessment is the backward error for an approximate least squares solution x, which measures the perturbation to A that would make x an exact least squares solution to a perturbed least squares problem:

$$\mu(x) \equiv \min_E \|E\|_F \quad \text{s.t.} \quad (A+E)^{\mathrm{T}}(A+E)x = (A+E)^{\mathrm{T}}b, \tag{24}$$

Waldén [23] et al. and Higham [12] proved that the backward error $\mu(x)$ is the smallest singular value of the matrix

$$\left[A \quad \tfrac{\|r\|_2}{\|x\|_2}\left(I - \tfrac{rr^{\mathrm{T}}}{\|r\|_2^2}\right)\right].$$

In our numerical experiments, we use the following easily computable estimate of backward error in Stewart [22],

$$\hat{E} = -\frac{rr^{\mathrm{T}}A}{\|r\|_2^2}, \quad \|\hat{E}\|_2 = \frac{\|A^{\mathrm{T}}r\|_2}{\|r\|_2}, \tag{25}$$

which satisfies the constraint in (24) but does not achieve the minimum there. Backward error is widely used to estimate the accuracy and stability of a method for least squares problems. It is usually accepted that the smaller the backward error is, the more accurate the approximate solution is.

Table 4 collects the numbers of iterations by the methods on the eight problems, where the best results appear in boldface and for the places marked with "–" it means that a method fails to solve a corresponding problem, i.e., satisfying (23) within 10^5 iterations. Once again the parameter ℓ in Table 4 indicates that the ℓ-step Lanczos method is used with MINRES to solve (22) in FLSMR and FMLSMR. NRes and backward errors are displayed in Figures 1 and 2 for six of the eight problems. The total CPU times for each example are shown in Table 5.

In Table 6, we display the estimates of $\|G_{k+1}\|_2$. We observe that for several different random right-side vectors, for the fixed $\ell = 10$ for crack, the norms of all G_{k+1} are around 34.8. For the other examples, such as cat_ears_3_4 and

TABLE 4. Number of Iterations

ID	matrix	ℓ	LSMR	FLSMR	FMLSMR
1	well1850	8	463	167	**117**
2	cat_ears_3_4	8	163	26	**25**
3	delaunay_n16	30	–	–	**14571**
4	biplane-9	15	–	–	**24611**
5	flower_7_4	8	195	29	**28**
6	crack	10	54015	–	**7400**
7	fe_body	30	–	–	**11200**
8	stufe-10	10	28047	–	**3297**

TABLE 5. CPU Time (sec.)

ID	matrix	ℓ	LSMR	FLSMR	FMLSMR
1	well1850	8	**0.0321**	0.4880	0.0518
2	cat_ears_3_4	8	0.2999	0.4761	**0.1220**
3	delaunay_n16	30	–	–	**1306.3191**
4	biplane-9	15	–	–	**452.9867**
5	flower_7_4	8	0.4535	1.0227	**0.4042**
6	crack	10	30.8903	–	**28.7795**
7	fe_body	30	–	–	**517.2462**
8	stufe-10	10	23.8165	–	**23.0792**

flower_7_4, we also have similar observations that for a given ℓ, $\|G_{k+1}\|_2$ is moderate.

TABLE 6. $\|G_{k+1}\|_2$

ID	matrix	ℓ	# of Columns	$\|G_{k+1}\|_2$
1	well1850	8	117	4.0758
2	cat_ears_3_4	8	25	3.1714
3	delaunay_n16	30	12599	36.2201
4	biplane-9	15	24504	27.1250
5	flower_7_4	8	28	3.1924
6	crack	10	7424	34.8286
7	fe_body	30	9301	28.2897
8	stufe-10	10	3297	11.9839

We have the following observations from Table 4, 5, 6 and Figures 1 and 2.

- The FMLSMR can solve all eight problems while LSMR succeeds on five of them and FLSMR on only three. Overall, FMLSMR has the best performance in terms of iteration numbers and CPU times, except for well1850 on which both FLSMR and FMLSMR are fast while FLSMR holds an edge. Specifically, for well1850 and stufe-10, LSMR and FMLSMR have comparable performance in computational time. For cat_ears_3_4 and flower_7_4, FMLSMR

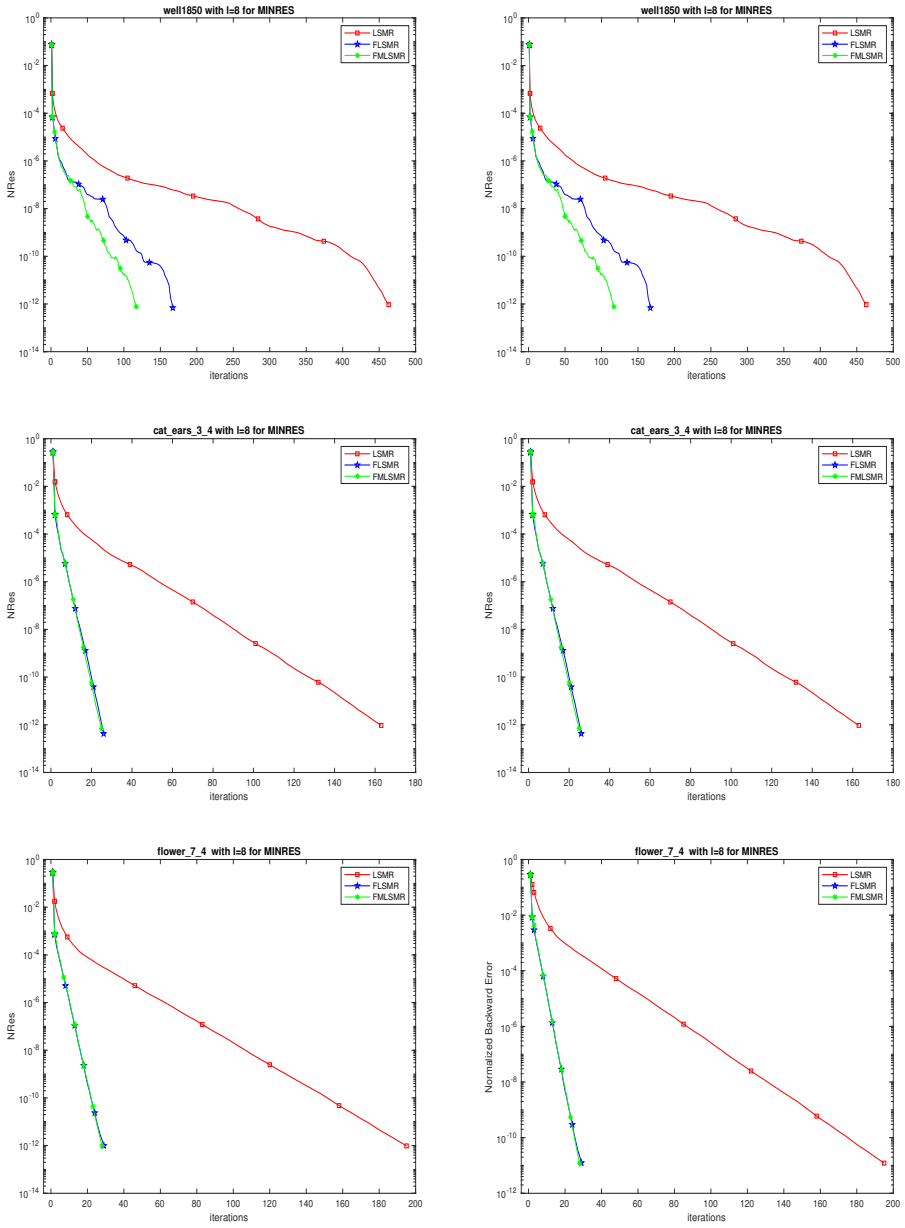

FIGURE 1. NRes (*left panel*) and normalized backward error $\|\hat{E}\|_2/\|A\|_1$ (*right panel*) for `well1850`, `cat_ears_3_4`, and `flower_7_4`.

is better than LSMR in terms of the number of iterations and CPU time. On `delaunay_n16`, `biplane-9`, `fe_body`, `brack2`, and `stufe-10`, FLSMR fails to satisfy the stopping criteria even for hours. According to Table 2, as the number of iterations increase, FLSMR uses much more storage and spends more

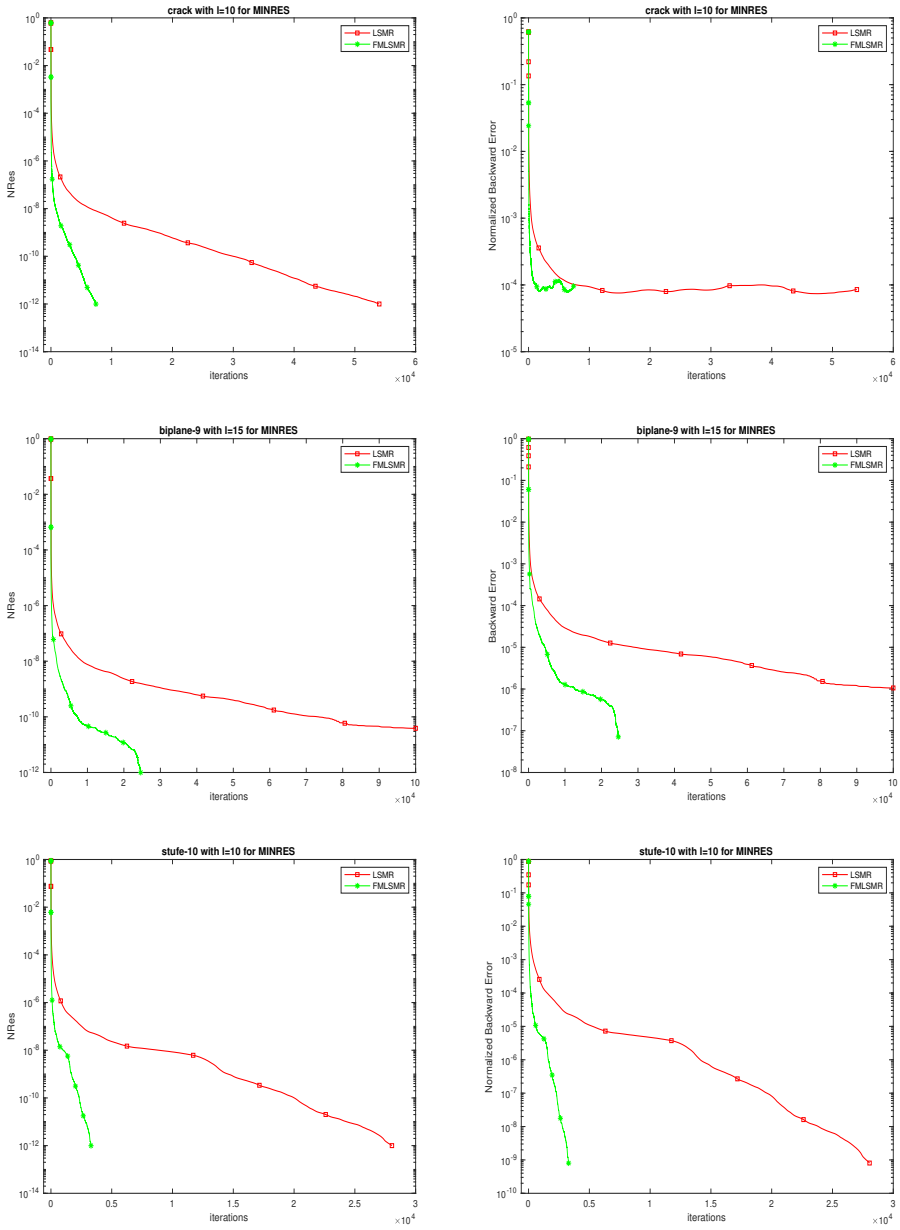

FIGURE 2. NRes (*left panel*) and normalized backward error $\|\hat{E}\|_2/\|A\|_1$ (*right panel*) for `crack`, `biplane-9`, and `stufe-10`.

on orthogonalization. The plots in left column of Figures 1 and 2 demonstrate a consistent decrease in relative residual across all methods. Notably, FMLSMR exhibits the fastest convergence among all. Therefore, considering the storage advantage of FMLSMR and its simple implementation, we can say

that FMLSMR is a very good choice, especially for difficult problems, over FLSMR and LSMR.

- The plots in the right column of Figures 1 and 2 show backward errors (25) for selective problems, and they display very similar patterns to that of NRes (23). Both $\|\hat{E}^{\mathrm{FMLSMR}}\|$ and $\|\hat{E}^{\mathrm{FLSMR}}\|$ are less than $\|\hat{E}^{\mathrm{LSMR}}\|$, which indicates that FMLSMR and FLSMR compute more accurate solutions than LSMR does for the same number of iterations.
- The norms of G_{k+1} in Table 6 are modest. This indicates that the solutions obtained by FMLSMR are good approximates of solutions to the original least square problem (1) as we discussed in Section 3.

5. **Conclusion.** In this paper, we present a new method, the Flexible Modified LSMR (FMLSMR), which integrates the key ideas from the Modified LSMR and Flexible GMRES algorithms. We conduct a theoretical analysis of the Modified LSMR and compare it with the Flexible LSMR (FLSMR) when using a given fixed preconditioner. Through numerical experiments, we illustrate the efficiency of FMLSMR from various angles. The advantages of our method in terms of storage and computational cost position it as a promising numerical method for tackling challenging problems in practical applications.

Acknowledgment. The authors wish to thank anonymous referees for their constructive comments that improve the presentation considerably.

REFERENCES

[1] M. Arioli, Generalized Golub-Kahan bidiagonalization and stopping criteria, *SIAM. J. Matrix Anal. Appl.*, **34** (2013), 571-592.
[2] S. R. Arridge, M. M. Betcke and L. Harhanen, Iterated preconditioned LSQR method for inverse problems on unstructured grids, *Inverse Problems*, **30** (2014), 1-27.
[3] H. Avron, E. Ng and S. Toledo, Using perturbed QR factorizations to solve linear least squares problems, *SIAM J. Matrix Anal. Appl.*, **31** (2009), 674-693.
[4] A. H. Baker, E. R. Jessup and T. Manteuffel, A technique for accelerating the convergence of restarted GMRES, *SIAM J. Matrix Anal. Appl.*, **26** (2005), 962-984.
[5] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadephia, 1996.
[6] J. Chung and S. Gazzola, Flexible Krylov methods for ℓ_p regularization, *SIAM J. Sci. Comput.*, **41** (2019), S149-S171.
[7] J. Chung and A. K. Saibaba, Generalized hybrid iterative methods for large-scale Bayesian inverse problems, *SIAM J. Sci. Comput.*, **39** (2017), S24-S46.
[8] D. C.-L. Fong and M. Saunders, LSMR: an iterative algorithm for sparse least-squares problems, *SIAM J. Sci. Comput.*, **33** (2011), 2950-2971.
[9] L. Giraud, S. Gratton, X. Pinel and X. Vasseur, Flexible GMRES with deflated restarting, *SIAM J. Sci. Comput.*, **32** (2010), 1858-1878.
[10] G. Golub and W. Kahan, Calculating the singular values and pseduo-inverse of a matrix, *SIAM J. Numer. Anal. Ser. B*, **2** (1965), 205-224.
[11] G. Golub and Q. Ye, Inexact preconditioned conjugate gradient method with inner-outer iteration, *SIAM J. Sci. Comput.*, **21** (1999), 1305-1320.
[12] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2^{nd} edition, SIAM, Philadephia, 2002.
[13] A. Imakura, R.-C. Li and S. L. Zhang, Locally optimal and heavy ball GMRES methods, *Jpn. J. Ind. Appl. Math.*, **33** (2016), 471-499.
[14] G. Karaduman and M. Yang, An alternative method for SPP with full rank (2, 1)-block matrix and nonzero right-hand side vector, *Turk. J. Math.*, **46** (2022), 1330-1341.
[15] G. Karaduman, M. Yang and R.-C. Li, A least squares approach for saddle point problems, *Jpn. J. Ind. Appl. Math.*, **40** (2023), 95-107.
[16] N. Li and Y. Saad, MIQR: A mutilevel incomplete QR preconditioner for large sparse least-squares problems, *SIAM J. Matrix Anal. Appl.*, **28** (2006), 524-550.

[17] K. Morikuni and K. Hayami, Convergence of inner-iteration GMRES methods for rank-deficient least squares problems, *SIAM J. Sci. Comput.*, **36** (2015), 225-250.
[18] C. C. Paige and M. A. Saunders, LSQR: An algorithm for sparse linear equations and sparse least squares, *ACM Trans. Math. Software*, **8** (1982), 43-71.
[19] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, *SIAM J. Sci. Comput.*, **14** (1993), 461-469.
[20] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2^{nd} edition, SIAM, Philadelphia, 2003.
[21] V. Simoncini and D. B. Szyld, Flexible inner-outer Krylov subspace methods, *SIAM J. Numer. Anal.*, **40** (2002), 2219-2239.
[22] G. W. Stewart, Research, development, and LINPACK, in *Mathematical Software*, Academic Press, New York, (1977), 1-14.
[23] B. Waldén, R. Karlson, and J.-G. Sun, Optimal backward perturbation bounds for the linear least squares problem, *Numer. Linear Algebra Appl.*, **2** (1995), 271-286.
[24] M. Yang, *Optimizing Krylov Subspace Methods for Linear Systems and Least Squares Problems*, Ph.D. thesis, University of Texas at Arlington, 2018.
[25] M. Yang and R.-C. Li, Heavy ball flexible GMRES method for nonsymmetric linear systems, *J. Comput. Math.*, **40** (2022), 711-727.

Received September 2024; 1^{st} revision March 2025; final revision June 2025; early access July 2025.