OPTIMIZING KRYLOV SUBSPACE METHODS FOR LINEAR SYSTEMS AND LEAST SQUARES PROBLEMS

by

MEI YANG

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON
August 2018

To my parents Fangrong Zhang and Wei Yang,

And my husband Zicong Zhou and little boy Wenyuan Zhou,

who give me a happy life.

ACKNOWLEDGEMENTS

First of all, I would like to thank my academic advisor, Professor Ren-Cang Li. I feel very fortunate and honored for having had the opportunity to work with him. Without Prof. Li's help, I couldn't come to the US and finish my doctral degree. I thank Prof. Li from the bottom of my heart for his constant support, patience, encouragement and for putting so much effort and time into guiding me.

I would also like to thank Prof. Guojun Liao, Prof. Chaoqun Liu, and Prof. Li Wang for their interest in my research and for taking time to serve in my dissertation committee. I wish to thank everyone in the Department of Mathematics at UTA, for being kind, supportive and encouraging during my doctoral studies.

I thank all my friends during doctoral study. I want to thank Gul Karaduman, Sat Byul Seo, Tiffany Tsai and Morgan Choi who shared nice friendships and unforgettable memory with me for the past several years. I am grateful to have lovely church families who helped me a lot when I was in need, especially when the baby was young. All of them made my intensive study life colorful.

I want to express my deepest gratitude and love for my wonderful parents and my grandparants. Their spiritual support is the most biggest motivation for me to pursue my Ph.D degree. They didn't limit my choices but gave my fully encouragement and valuable trust to let me pursue my dream.

Finally, I want to express my love to my dearest husband, Zicong Zhou, who loves, cares, and supports my throughout the past few years. He is alway being there

with me and help me in every aspects, from daily cooking to mathematics dissusion. He is the best husband I've ever seen. Without his presence, none of my success would have been possible.

August 10, 2018

ABSTRACT

OPTIMIZING KRYLOV SUBSPACE METHODS FOR LINEAR SYSTEMS AND LEAST SQUARES PROBLEMS

Mei Yang, Ph.D.

The University of Texas at Arlington, 2018

Supervising Professor: Ren-Cang Li

The linear system and the linear least squares problem are two fundamental numerical linear algebra problems. Krylov subspace methods are the most practical and common techniques to build solvers. In this thesis, we focus on optimizing Krylov subspace methods for nonsymmetric linear systems and least squares problems.

For nonsymmetric linear systems $Ax = b$, one of Krylov subspace methods is GMRES, which seeks approximate solutions over the Krylov subspace $\mathcal{K}_k(A, b)$ (with given initial guess $x_0 = 0$). Constucting different search spaces and applying restart strategy are two techniques used to deal with possible slow convergence and to reduce computational cost in GMRES and variants of GMRES, such as restarted GMRES and flexible GMRES. In this thesis, we present a numerical method called the heavy ball flexible GMRES method for nonsymmetric linear systems, which is designed to salvage the lost convergence speed of restarted flexible GMRES while keeping the benefit of the restarted flexible GMRES in limiting memory usage and controlling orthogonalization cost. The search space is extended in every iteration by applying the heavy ball idea in optimization. Comparison of the restarted flexible

GMRES and the heavy ball flexible GMRES are shown in numerical experiments to illustrate the efficiency of our method.

The linear least squares problem, $\min_x \|Ax - b\|_2$, widely occurs in statistics, geodetics, photogrammetry, signal processing, and control. Based on the Golub-Kahan process, LSMR seeks approximate solutions x_k over the Krylov subspace $\mathcal{K}_k\left(A^TA, A^Tb\right)$. We are aiming to optimize LSMR in two ways. Firstly, we propose the heavy ball minimal residual method by utilizing the restarted technique to combine LSMR with the heavy ball idea. The restarted Golub-Kahan bidiagonalization process is applied to control the memory usage and reorthogonalization cost. We add a new direction to include previous iterations' information in extending the searching subspace, which also speeds up the convergence. Secondly, we present a flexible preconditioned iterative method for least squares problems. Because of lacking of effective preconditioners, LSMR sometimes stagnates as iterations go on. Applying a good preconditioner in LSMR is critical to speed up convergence. Usually, it's impossible to tell whether or not a given preconditioner is suitable for the problem beforehand. So, one may attempt many possible preconditioners together and switch periodically among them. We want to design an algorithm that can switch among preconditioners in the outer iterations instead of restarting LSMR. Our flexible preconditioned LSMR method takes an outer-inner iteration to constuct changable preconditioners, and applies the factorization-free preconditioning technique to the Golub-Kahan process to reduce computational cost. Numerical examples demonstrate the efficiency of our method compared to LSMR.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

Solving systems of linear equations has been being an old but fascinating problem since the creation of computers. Computational algorithms for finding solutions are important parts of numerical linear algebra, and play prominent roles in engineering, physics, chemistry, computer science and economics. For linear systems with dense coefficient matrices, direct methods such as LU factorization [28, 29], Cholesky factorization and QR factorization [27] are popular for finding solutions, but not economic and efficient for large and sparse problems. In this thesis, we focus on the solutions of linear systems with large and sparse coefficient matrices.

1.1 Problem Statement

We consider the problem of solving a system of linear equations in the form

$$Ax = b, \tag{1.1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$. A is typically large and sparse, or is available through a matrix-vector product subroutine.

We call (1.1) a consistent system if a solution x that satisfies (1.1) exists. If such x doesn't exist, we have an inconsistent system. In this case, we look for an optimal x of the following least squares problem instead:

$$\min_x \|Ax - b\|_2 . \tag{1.2}$$

We denote the exact solution of (1.1) or (1.2) by x^*, and by x_k the approximate solution of k-th iteration obtained by any iterative method.

In sections 1.2 and 1.3, we review a number of Krylov subspace methods for square linear systems and least squares problems, respectively.

1.2 Krylov Subspace Methods for Square Linear Systems

Basic iterative methods like Jacobi iteration, Gauss-Seidel iteration [27] and successive over-relaxation (SOR) [67] require less memory than Krylov subspace methods that we will discuss, but they are often slow and will not converge to the exact solution x^* in a finite number of iterations even with exact arithmetic. They work only for a few type of matrices such as diagonal dominant matrices.

In this section, we focus on square systems, i.e., $m = n$, solved by Krylov subspace methods. If $\mathcal{K}_k$ is a search space with dimension k, then, in general, k constraints must be imposed in order to be able to extract an approximate solution of (1.1). Usually, the residual $r_k = b - Ax_k$ is constrained to be orthogonal to another k dimension subspace $\mathcal{L}_k$, i.e.,

$$b - Ax_k \perp \mathcal{L}_k, \tag{1.3}$$

which is known as the Petrov-Galerkin condition. Note that $\mathcal{L}_k$ and $\mathcal{K}_k$ can be the same or different.

A Krylov subspace method is a projection method with the search space to be a Krylov subspace,

$$\mathcal{K}_k(A, r_0) = \text{span}\left(r_0, Ar_0, A^2r_0, \cdots, A^{k-1}r_0\right), \tag{1.4}$$

where $r_0 = b - Ax_0$ and x_0 represents an arbitrary initial guess. When there is no ambiguity, $\mathcal{K}_k(A, r_0)$, will be denoted by $\mathcal{K}_k$. The approximate solution x_k satisfies $x_k \in x_0 + \mathcal{K}_k$. The different versions of Krylov subspace methods arise from different choices of the subspace $\mathcal{L}_k$ and from the ways in which the system is preconditioned.

1.2.1 The Arnoldi Process

The Arnoldi process [3] is an algorithm for building an orthogonal basis of the Krylov subspace $\mathcal{K}_k$ in (1.4). It transforms partially an unsymmetric matrix A

Algorithm 1.1 Arnoldi process

1: Given a vector t, set $\beta_1 = \|t\|_2, v_1 = t/\beta_1$
2: **for** $j = 1, 2, \ldots k$ **do**
3: $\quad w_j = Av_j$
4: $\quad$ **for** $i = 1, 2, \ldots, j$ **do**
5: $\quad\quad h_{ij} = w_j^T v_i$
6: $\quad\quad w_j = w_j - h_{ij} v_i$
7: $\quad$ **end for**
8: $\quad v_{j+1} = w_j/\beta_{j+1}$
9: **end for**

into an upper Heisenberg matrix H_k with an orthonormal matrix V_k. Algorithm 1.1 is one version of the Arnoldi process by using the modified Gram-Schmidt (MGS) algorithm.

The process can be summarized by

$$\begin{aligned} AV_k &= V_k H_k + h_{k+1,k} v_{k+1} e_k^T \\ &= V_{k+1} \check{H}_k, \\ V_k^T A V_k &= H_k, \end{aligned} \tag{1.5}$$

where $V_k = [v_1, v_2, \cdots, v_k]$ is an orthonormal matrix, i.e., $V_k^T V_k = I_k$, and

$$V_{k+1} = [V_k \quad v_{k+1}], \quad \check{H}_k = [H_k^T \quad \beta_{k+1} e_k]^T,$$

$$H_k = \begin{bmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1k} \\ \beta_2 & h_{22} & \cdots & \cdots & h_{2k} \\ 0 & \beta_3 & \cdots & \cdots & h_{3k} \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & \beta_k & h_{kk} \end{bmatrix}.$$

FOM

Given an initial guess x_0, take $\mathcal{L}_k = \mathcal{K}_k = \mathcal{K}_k(A, r_0)$, and $t = r_0 \equiv b - Ax_0$ in the Arnoldi process. An approximate solution x_k is sought from the affine subspace $x_0 + \mathcal{K}_k$ by imposing the Galerkin condition

$$b - Ax_k \perp \mathcal{K}_k,$$

which implies

$$V_k^T(b - Ax_k) = 0.$$

According to (1.5), we know

$$x_k = x_0 + V_k y_k, \quad y_k = H_k^{-1}(\beta_1 e_1). \tag{1.6}$$

The full orthogonalization method (FOM) [51] is based on the above approach and is shown in Algorithm 1.2.

Algorithm 1.2 FOM

1: Given an initial guess x_0, set $r_0 = b - Ax_0, \beta_1 = \|r_0\|_2$, and $v_1 = r_0/\beta_1$

2: Compute H_k and V_k as in lines 2-9 in Algorithm 1.1

3: $y_k = H_k^{-1}(\beta_1 e_1)$, and $x_k = x_0 + V_k y_k$

GMRES

Take $\mathcal{L}_k = A\mathcal{K}_k = A\mathcal{K}_k(A, r_0)$. The Galerkin condition becomes

$$b - Ax_k \perp A\mathcal{K}_k,$$

i.e., x_k is the minimizer of $\min_{x_k \in x_0 + \mathcal{K}_k} \|b - Ax\|_2$. After k steps of the Arnoldi process, the approximate solution x_k can be written as

$$x_k = x_0 + V_k y_k, \quad y_k = \arg\min_y \|\beta_1 e_1 - \check{H}_k y\|_2. \tag{1.7}$$

The summary of generalized minimal residual (GMRES) method [55] is shown in Algorithm 1.3.

Algorithm 1.3 GMRES

1: Given an initial guess x_0, set $r_0 = b - Ax_0, \beta_1 = \|r_0\|_2$, and $v_1 = r_0/\beta_1$

2: Compute H_k and V_k as in lines 2-9 in Algorithm 1.1

3: $y_k = \arg\min_y \|\beta_1 e_1 - \check{H}_k y\|_2$, and $x_k = x_0 + V_k y_k$

1.2.2 The Lanczos Process

When A is symmetric, the Arnoldi process can be reduced to the symmetric Lanczos process [44]. The Heisenberg matrix H_k becomes an symmetric tridiagonal matrix T_k. This leads to a three-term recurrence in the Arnoldi process and a short-term recurrence for x_k as shown in Algorithm 1.4.

The process can be summarized in the matrix form as

$$AV_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T = V_{k+1} \check{H}_k \tag{1.8}$$

with $V_k = [v_1, v_2, \cdots, v_k]$ and

$$T_k = \begin{bmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_k \\ & & \beta_k & \alpha_k \end{bmatrix}, \quad \check{H}_k = \begin{bmatrix} T_k \\ \beta_{k+1} e_k^T \end{bmatrix}.$$

Algorithm 1.4 Lanczos process

1: Given a vector t, set $\beta_1 = \|t\|_2, v_1 = t/\beta_1, v_0 = 0$

2: **for** $j = 1, 2, \ldots k$ **do**

3: $\quad w = Av_j$

4: $\quad \alpha_j = w^T v_j$

5: $\quad v_{j+1} = w - \alpha_j v_j - \beta_j v_{j-1}$

6: $\quad \beta_{j+1} = \|v_{j+1}\|_2$

7: $\quad v_{j+1} = v_{j+1}/\beta_{j+1}$

8: **end for**

CG

The conjugate gradient (CG) [34, 44] method is one of the best known iterative methods for solving a symmetric positive definite (SPD) linear system. Mathematically, CG is equivalent to FOM, i.e., $\mathcal{L}_k = \mathcal{K}_k = \mathcal{K}_k(A, r_0)$ and the same Galerkin condition. It has one formulation which can be derived from Lanczos process [54]. The k-th approximate solution x_k is given by

$$x_k = x_0 + V_k y_k, \quad y_k = \arg\min_{y \in \mathcal{K}_k} \phi(V_k y), \tag{1.9}$$

where $\phi(x) = \frac{1}{2}x^T Ax - b^T x$ is the quadratic form. In CG, all residual vectors r_j's are orthogonal, and the search direction p_j's are A-orthogonal, i.e., conjugate. Algorithm 1.5 shows one version of the CG method.

Algorithm 1.5 CG

1: Given an initial guess x_0, set $r_0 = b - Ax_0, p_1 = r_0, \rho_0 = r_0^T r_0$
2: **for** $j = 1, 2, \ldots k$ **do**
3: $\quad q_j = Ap_j$
4: $\quad \alpha_j = \rho_{j-1}/p_j^T q_j$
5: $\quad x_j = x_{j-1} + \alpha_j p_j$
6: $\quad r_j = r_{j-1} - \alpha_j q_j$
7: $\quad \rho_j = r_j^T r_j$
8: $\quad \beta_j = \rho_j/\rho_{j-1}$
9: $\quad p_{j+1} = r_j + \beta_j p_j$
10: **end for**

MINRES

If applying GMRES to a symmetric linear system, it reduces to the minimal residual (MINRES) method [48]. The minimization problem is with the same form as in GMRES but with $\check{H}_k = [T_k^T \quad \beta_{k+1} e_k]^T$, i.e.,

$$x_k = x_0 + V_k y_k, \quad y_k = \arg\min_y \|\beta_1 e_1 - \check{H}_k y\|_2.$$

MINRES is applicable to both definite and indefinite systems. Similar to MINRES approach for symmetric linear systems, there are a number of methods proposed to minimize the norm of residual over all vectors in the Krylov subspace, such as ORTHODIR [38], ORTHOMIN [64] and Axelsson's method [6].

MINRES applies the QR factorization to transform $[H_k, \beta_1 e_1]$ into an upper tridiagonal matrix

$$\begin{bmatrix} R_k & z_k \\ 0 & \bar{\zeta}_{k+1} \end{bmatrix},$$

Then, define W_k satisfying $R_k^T W_k^T = V_k^T$, and construct an recurrence of solutions by $x_k = V_k y_k = W_k R_k y_k = W_k z_k = x_{k-1} + \zeta_k w_k$. LSQR uses the same techniques for least squares problems, which will be discussed later.

CR

The conjugate residual (CR) method [60] is another version of GMRES for an SPD A. In this case, the residual vectors are made A-orthogonal, i.e., conjugate. The search directions are $A^T A$-orthogonal. The CR method is shown in Algorithm 1.6. Since CR and MINRES both work for SPD systems as the special case of GMRES,

Algorithm 1.6 CR

1: Given an initial guess x_0, set $r_0 = b - Ax_0, s_0 = Ar_0, p_0 = r_0, \rho_0 = (r_0, s_0), q_0 = s_0$
2: **for** $j = 1, 2, \ldots, k$ **do**
3: $\quad \alpha_j = \rho_{j-1} / \|q_{j-1}\|_2$
4: $\quad x_j = x_{j-1} + \alpha_j p_{j-1}$
5: $\quad r_j = r_{j-1} - \alpha_j q_{j-1}$
6: $\quad s_j = Ar_j$
7: $\quad \rho_j = r_k^T s_j$
8: $\quad \beta_j = \rho_j / \rho_{j-1}$
9: $\quad p_j = r_j + \beta_j p_{j-1}$
10: $\quad q_j = s_j + \beta_j q_{j-1}$
11: **end for**

they generate the same iterations on SPD systems. For nonsymmetric linear systems, we have the generalized conjugate gradient method (GCR) [18] shown in Algorithm 1.7 which is equivalent to the full GMRES.

Algorithm 1.7 GCR

Given an initial guess x_0, set $r_0 = b - Ax_0, s_0 = Ar_0, p_0 = r_0, q_0 = s_0, \rho_0 = r_0^T q_0$
for $j = 1, 2, \ldots, k$ **do**
    $\alpha_j = \rho_{j-1}/\|q_{j-1}\|_2$
    $x_j = x_{j-1} + \alpha_j p_{j-1}$
    $r_j = r_{j-1} - \alpha_k q_{j-1}$
    $s_j = Ar_j$
    for $i = 1, 2, \ldots, j$ **do**
        $\beta_i = s_j^T q_i/\|q_i\|_2$
        $p_j = r_j - \beta_i p_i$
        $q_j = s_j - \beta_i q_i$
    end for
    $\rho_j = r_j^T q_j$
end for

If A is not symmetric, there is an unsymmetric Lanczos process which relax the orthogonality requirement of V_k. This leads to Bi-CG[21], an extension of CG, and the quasi-minimum residual method (QMR) [24] which is an analog version of MINRES. There are some related methods based on Bi-CG such as Bi-CGS [58], Bi-CGSTAB [63] and Bi-CGSTAB(ℓ) [57].

1.3 Krylov Subspace Methods for Least Squares Problems

In this section, we introduce a number of Krylov subspace methods for the matrix equation $Ax = b$, where A is an m-by-n square or rectangular matrix. When $m > n$, we solve the least squares problem $\min_x \|Ax - b\|_2$. This is the most common case. In this thesis, we focus on solving overdtermined least squares problems. When $m < n$, $Ax = b$ is underdetermined.

When $m > n$, we also call $Ax = b$ an overdetermined system because the number of equations is more than unknowns. It is proved that solutions of the least squares problem (1.2) are solutions of the normal equation [50]

$$A^T Ax = A^T b, \tag{1.10}$$

for which one can apply any of the Krylov subspace methods for the symmetric linear system.

1.3.1 The Golub-Kahan process

Golub and Kahan [25] proved a matrix $A \in \mathbb{R}^{m\times n}$ with $m > n$ can be decomposed into the bidiagonal form through Householder transformations [36],

$$A = U \begin{bmatrix} B \\ 0 \end{bmatrix} V^T, \quad U^T U = I_m, \quad V^T V = I_n,$$

where $U = [u_1, u_2, \cdots, u_m], V = [v_1, v_2, \cdots, v_n]$, and B is lower bidiagonal. The Golub-Kahan process is also referred to as the Lanczos bidiagonalization [50] for a rectangular matrix. Golub and Kahan gave an iterative version of bidiagonalization as shown in Algorithm 1.8.

After the k-th step, we have

$$AV_k = U_{k+1}B_k \quad \text{and} \quad A^T U_{k+1} = V_{k+1}L_{k+1}^T, \tag{1.11}$$

Algorithm 1.8 Golub-Kahan process

1: Given t and set $u_1 = t/\left\|t\right\|_2, \beta_1 = \left\|t\right\|_2$,and $\alpha_1 = \left\|A^T u_1\right\|_2, v_1 = A^T u_1/\alpha_1$

2: **for** $j = 1, 2, \ldots k$ **do**

3: $\quad q = Av_j - \alpha_j u_j, \beta_{j+1} = \left\|q\right\|_2, u_{j+1} = q/\beta_{j+1}$

4: $\quad p = A^T u_{j+1} - \beta_{j+1} v_j, \alpha_{j+1} = \left\|p\right\|_2, v_{j+1} = p/\alpha_{j+1}$

5: **end for**

where

$$V_k = [v_1, v_2, \cdots, v_k], \quad U_{k+1} = [u_1, u_2, \cdots, u_{k+1}],$$

$$B_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix}, \quad L_{k+1} = [B_k \quad \alpha_{k+1} e_{k+1}].$$

The above process is equivalent to the symmetric Lanczos process working on the matrix $A^T A$ with starting vector $A^T t$. For given initial guess x_0, let $t = r_0 \equiv b - Ax_0$, the k-th approximate solution x_k is $x_k = x_0 + V_k y_k$, and the residual r_k can be written as

$$r_k = b - A(x_0 + V_k y_k) = r_0 - AV_k y_k = \beta_1 u_1 - U_{k+1} B_k y_k = U_{k+1}(\beta_1 e_1 - B_k y_k).$$

We can tell that $x_k \in x_0 + \text{span}(V_k)$, and $\text{span}(V_k) = \mathcal{K}_k(A^T A, A^T r_0)$. How to find an approximate solution y_k to solve $B_k y_k \approx \beta_1 e_1$ determines different methods.

LSQR

LSQR [15] solves $\min\limits_x \|r\|_2 \equiv \min\limits_y \|\beta_1 e_1 - B_k y\|_2$ by the QR factorization on the lower bidiagonal matrix B_k, which is a similar technique as mentioned in MINRES.

Algorithm 1.9 is one adapted version of LSQR. At each iteration, $\|\beta_1 e_1 - B_j y\|_2$ is minimized over a larger Krylov subspace, which implies that r_j is monotonically decreasing.

Algorithm 1.9 LSQR

1: Given initial guess x_0 and set $r_0 = b - Ax_0, \beta_1 u_1 = r_0, \alpha_1 v_1 = A^T u_1, w_1 = v_1, \bar{\phi}_1 = \beta_1, \bar{\rho}_1 = \alpha_1$

2: **for** $j = 1, 2, \ldots k$ **do**

3: $\quad \beta_{j+1} u_{j+1} = A v_j - \alpha_j u_j$

4: $\quad \alpha_{j+1} v_{j+1} = A^T u_{j+1} - \beta_j v_j$

5: $\quad \rho_j = (\bar{\rho}_j^2 + \beta_{j+1}^2)^{1/2}$

6: $\quad c_j = \bar{\rho}_j / \rho_j$

7: $\quad s_j = \beta_{j+1} / \rho_j$

8: $\quad \theta_{j+1} = s_j \alpha_{j+1}$

9: $\quad \bar{\rho}_{j+1} = c_j \alpha_{j+1}$

10: $\quad \phi_j = c_j \bar{\phi}_j$

11: $\quad \bar{\phi}_{j+1} = -s_j \bar{\phi}_j$

12: $\quad x_j = x_{j-1} + (\phi_j / \rho_j) w_j$

13: $\quad w_{j+1} = v_{j+1} - (\theta_{j+1} / \rho_j) w_j$

14: **end for**

CGLS

Mathematically, LSQR is equivalent to CG applied to the normal equation (1.10), which is called CGLS [34, 15]. See Algorithm 1.10.

Algorithm 1.10 CGLS

Given an initial guess x_0, set $r_0 = b - Ax_0, S_0 = A^T r_0, p_1 = s_0, \rho_0 = s_0^T s_0$

for $j = 1, 2, \ldots k$ **do**

  $q_j = Ap_j$

  $\alpha_j = \rho_{j-1}/p_j^T q_j$

  $x_j = x_{j-1} + \alpha_j p_j$

  $r_j = r_{j-1} - \alpha_j q_j$

  $s_j = A^T r_j$

  $\rho_j = s_j^T s_j$

  $\beta_j = \rho_j/\rho_{j-1}$

  $p_{j+1} = s_j + \beta_j p_j$

end for

LSMR

LSMR [23, 22] minimizes $\|A^T r\|_2$ over the Krylov subspace $\mathcal{K}_k(A^T A, A^T r_0)$. According to (1.11), the minimization can be simplified to be a subproblem

$$\min_y \left\| \bar{\beta}_1 e_1 - \begin{bmatrix} B_k^T B_k \\ \bar{\beta}_{k+1} e_k^T \end{bmatrix} y \right\|_2, \tag{1.12}$$

where $\bar{\beta}_k = \alpha_k \beta_k$. By applying double QR decomposition to (1.12), the approximate solution x_k is iteratively obtained as shown in Algorithm 1.11. Mathematically, LSMR is equivalent to MINRES applied to the normal equation (1.10).

Algorithm 1.11 LSMR

Given initial guess x_0 and set $r_0 = b - Ax_0, \beta_1 u_1 = r_0, \alpha_1 v_1 = A^T u_1, \bar{\alpha}_1 = \alpha_1, \bar{\zeta}_1 = \alpha_1\beta_1, \rho_0 = 1, \bar{\rho}_0 = 1, \bar{c}_0 = 1, \bar{s}_0 = 0, h_1 = v_1, \bar{h}_0 = 0$

for $j = 1, 2, \ldots k$ **do**

$\beta_{j+1} u_{j+1} = A v_j - \alpha_j u_j$

$\alpha_{j+1} v_{j+1} = A^T u_{j+1} - \beta_j v_j$

$\rho_j = (\bar{\alpha}_j^2 + \beta_{j+1}^2)^{1/2}$

$c_j = \bar{\alpha}_j / \rho_j$

$s_j = \beta_{j+1} / \rho_j$

$\theta_{j+1} = s_j \alpha_{j+1}$

$\bar{\alpha}_{j+1} = c_j \alpha_{j+1}$

$\bar{\theta}_j = \bar{s}_{j-1} \bar{\rho}_j$

$\bar{\rho}_j = ((\bar{c}_{j-1}\rho_j)^2 + \theta_{j+1}^2)^{1/2}$

$\bar{c}_j = \bar{c}_{j-1}\rho_j / \bar{\rho}_j$

$\bar{s}_j = \theta_{j+1} / \bar{\rho}_j$

$\zeta_j = \bar{c}_j \bar{\zeta}_j$

$\bar{\zeta}_{j+1} = -\bar{s}_j \bar{\zeta}_j$

$\bar{h}_j = h_j - (\bar{\theta}_j \rho_j / (\rho_{j-1}\bar{\rho}_{j-1}))\bar{h}_{j-1}$

$x_j = x_{j-1} + (\zeta_j / (\rho_j \bar{\rho}_j))\bar{h}_j$

$h_{j+1} = v_{j+1} - (\theta_{j+1}/\rho_j) h_j$

end for

1.4 Preconditioning

Preconditioning [43, 61, 33, 20, 19, 10, 56, 46] is one important and popular technique to improve the performance and reliability of Krylov subspace methods. The term preconditioning refers to transforming a linear system (1.1) into another system with more favorable properties to iterative methods. A preconditioner is a matrix that conducts such a transformation. Most preconditioners are chosen to improve the spectral properties of the coefficient matrix, such as a smaller spectral condition number, and/or eigenvalues clustered around 1.

Given a nonsingular matrix M, a preconditioned linear system is

$$M^{-1}Ax = M^{-1}b, \tag{1.13}$$

which has the same solutions as (1.1) but may be easier to solve. We call M a preconditioner. There are many ways to construct preconditioners for instance, incomplete factorization [46, 32, 66, 52, 14, 11], multilevel preconditioners [54, p. 371] [7, 8, 45] and sparse approximate inverses [9, 41, 42, 13, 31, 12], and so on.

System (1.13) is preconditioned from the left, but it can also be preconditioned from the right, which is usually used to precondition a least squares problem. A right preconditioned least squares problem is written as

$$\min_y \|(AM^{-1})y - b\|_2, \quad Mx = y. \tag{1.14}$$

In general, a good preconditioner M should have the following properties [50, 43],

- The preconditioned system should be easy to solve, i.e. $M^{-1}A$ or AM^{-1} should be better conditioned than A and/or has only a few clustered singular values.
- The preconditioner should be cheap to construct and apply, i.e., equations with matrices M and M^T should be cheap to solve .

1.5 Overview

Chapter 2 compares some variants of GMRES and presents the heavy ball flexible GMRES method for nonsymmetric linear systems. Chapter 3 shows a heavy ball minimal residual method for least squares problems. In chapter 4, we propose the flexible preconditioned iterative method for least squares problems. The corresponding numerical examples of each method are shown in each chapter. We give our conclusions in chapter 5. The notations used in this thesis are summarized in Table 1.1.

Table 1.1: Notation

A	matrix, sparse matrix or linear operator	
A_{ij}	the element of matrix A at the cross of the i-th row and the j-th column	
$b, p, r, t, u, v, x, y, \cdots$	vectors	
k	subscript index for iteration number. E.g. x_k is the approximate solution generated at the k-th iteration of an iterative solver such as GMRES	
ℓ	superscript index for cycle number of solvers with restarted process. E.g. $x_k^{(\ell)}$ is the approximate solution generated at ℓ-th cycle of REGMRES with k-steps inner iteration .	
c_k, s_k	non-identity elements in a Givens rotation	
B_k	$(k+1)$-by-k lower bidiagonal matrix	
e_k	the k-th column of an identity matrix	
x^*	the unique solution to a nonsingular squares system $Ax = b$, or more generally the pseudo-inverse solution of a rectangular system $Ax \approx b$	
$\text{cond}(A)$	condition number of the coefficient matrix A	$\langle \cdot, \cdot \rangle$ dot product, i.e., $\langle u, v \rangle = v^T u$
$\langle \cdot, \cdot \rangle_M$	M-product, i.e., $\langle u, v \rangle_M = v^T M u$	

CHAPTER 2

HEAVY BALL FLEXIBLE GMRES METHOD FOR NONSYMMETRIC LINEAR SYSTEMS

In this chapter, we present a numerical method called the heavy ball flexible GMRES method (HBGMRES) for the nonsymmetric linear system (1.1). HBFGMRES is designed to salvage the lost convergence speed of the restarted flexible GMRES (REFGMRES) while keep the benefit of REFGMRES in limiting memory usage and controlling orthogonalization cost. In section 2.1, we review the restarted GMRES (REGMRES), the heavy ball GMRES (HBGMRES) and the flexible GMRES (FGMRES). In section 2.2, we show the motivation of our work and derive HBFGMRES in theory. Numerical examples in section 2.3 demonstrate the efficiency of HBFGMRES compared with REFGMRES.

2.1 Variants of GMRES

Let's recall the basic idea of GMRES. Given an initial guess x_0, the k-th approximate solution x_k is sought so that the k-th residual $r_k = b - Ax_k$ satisfies

$$\|r_k\|_2 = \min_{z \in \mathcal{K}_k(A, r_0)} \|b - A(x_0 + z)\|_2, \tag{2.1}$$

where the k-th Krylov subspace of A on r_0 is defined as

$$\mathcal{K}_k(A, r_0) = \text{span}(r_0, Ar_0, \cdots, A^{k-1}r_0).$$

Any $z \in \mathcal{K}_k(A, r_0)$ can be expressed as $z = V_k y$ for some $y \in \mathbb{R}^k$, and thus

$$b - Ax_k = r_0 - Az = r_0 - AV_k y = r_0 - V_{k+1}\check{H}_k y = V_{k+1}(\beta e_1 - \check{H}_k y),$$

Algorithm 2.1 (k-step) GMRES

Given any initial guess $x_0 \in \mathbb{R}^n$ and an integer $k \geq 1$, this algorithm computes a generalized minimal residual solution to the linear system $Ax = b$.

1: $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$

2: $V_{(:,1)} = r_0/\beta$, $\check{H} = 0_{(k+1)\times k}$ (the $(k+1) \times k$ zero matrix)

3: **for** $j = 1, 2, \ldots, k$ **do**

4: $\quad f = AV_{(:,j)}$

5: $\quad \check{H}_{(1:j,j)} = V_{(:,1:j)}^T f$, $f = f - V_{(:,1:j)}\check{H}_{(1:j,j)}$

6: $\quad \check{H}_{(j+1,j)} = \|f\|_2$

7: $\quad$ **if** $\check{H}_{(j+1,j)} > 0$ **then**

8: $\quad\quad V_{(:,j+1)} = f/\check{H}_{(j+1,j)}$

9: $\quad$ **else**

10: $\quad\quad$ reset $k = j$, $\check{H} = \check{H}_{(1:j,1:j)}$

11: $\quad\quad$ **break**

12: $\quad$ **end if**

13: **end for**

14: $y_k = \arg\min_y \|\beta e_1 - \check{H}y\|_2$

15: **return** $x_k = x_0 + V_k y_k$ as an approximate solution to $Ax = b$

where the basis matrix V_{k+1} and upper-Heisenberg matrix $\check{H}_k$ both are generated by the Arnoldi process as shown in (1.5) with the starting vector $v_1 = r_0/\|r_0\|_2$. Therefore

$$\min_{z \in \mathcal{K}_k(A,r_0)} \|b - A(x_0 + z)\|_2 = \min_{z \in \mathcal{K}_k(A,r_0)} \|r_0 - Az\|_2 = \min_y \|\beta e_1 - \check{H}_k y\|_2, \qquad (2.2)$$

where $\beta = \|r_0\|_2$. Solving the last problem in (2.2) yields the optimal solution y_{opt} which in turn gives $z_{\text{opt}} = V_k y_{\text{opt}}$ and finally the k-step GMRES solution is given by [30, 55]

$$x_k = x_0 + z_{\text{opt}} = x_0 + V_k y_{\text{opt}}. \tag{2.3}$$

For convenience and consistence in this chapter, we outline the k-step version of GMRES in Algorithm 2.1.

2.1.1 Restarted GMRES

For large scale linear systems, GMRES can be very expensive for large k. Heavy memory cost is demanded for storing all basis vectors v_j, and the orthogonalization cost in computing v_j increases quadratically in k. So restarted GMRES [55] is naturally born. We denote it by REGMRES(k) which attempts to control memory usage and orthogonalization cost by fixing k and repeatedly iterating the k-step GMRES with the current initial guess $x_0^{(\ell)}$ being the very previous k-step GMRES solution. The frame work is sketched in Algorithm 2.2 which includes inner iterations at Line 6 and outer cycle indexed by ℓ.

By limiting the number k of Arnoldi steps in REGREMES(k), possibly heavy memory burden and computation cost is successfully alleviated. However, it may cause severe degradation in convergence behavior because the loss of information when we restart the iteration from scratch its hard to maintain orthogonality against vectors weve thrown away As we know, the larger k is, the more accurate solutions are. This means the ℓ-th solution $x_k^{(\ell)}$ is always no better than the solution returned by ℓk-step GMRES in the sense that

$$\|b - Ax_k^{(\ell)}\|_2 \geq \|b - Ax_{\ell k}^{(1)}\|_2.$$

But we don't know how much worse it could be. So finding a way to narrow down the gap between the above two residuals is worth a try.

Algorithm 2.2 REGMRES(k)

Given any initial guess $x_0^{(1)} \in \mathbb{R}^n$ and an integer $k \geq 1$, this algorithm computes an approximate solution to the linear system $Ax = b$ via the restarted GMRES.

$r_0^{(1)} = b - Ax_0^{(1)}$
for $\ell = 1, 2, \ldots,$ **do**
    if $\|r_0^{(\ell)}\|_2 \leq \texttt{tol} \times (\|A\|_1 \|x_0^{(\ell)}\|_2 + \|b\|_2)$ **then**
        break
    else
        call Algorithm 2.1 with input $x_0^{(\ell)}$ and k, and let $x_k^{(\ell)}$ be the returned approximation
        $x_0^{(\ell+1)} = x_k^{(\ell)}$
        $r_0^{(\ell+1)} = b - Ax_0^{(\ell+1)}$
    end if
end for
return $x_0^{(\ell)}$ as the computed solution to $Ax = b$

2.1.2 Heavy Ball GMRES

As shown in Algorithm 2.2, the initial guess for the current GMRES cycle is the approximate GMRES solution of the very previous cycle, which completely ignores all the Krylov subspaces built in the previous cycles for the purpose of cost control in memory and flops. Based on this observation, Imakura, Li, and Zhang proposed the heavy ball GMRES method (HBGMRES) [37] . The so-called *heavy ball* method [49, p. 65] is the one:

$$x^{(\ell+1)} = \arg\min_{s,t} f(x^{(\ell)} + t\nabla f(x^{(\ell)}) + s(x^{(\ell)} - x^{(\ell-1)})),$$

drawing its name from the motion of a "heavy ball" in a potential field under the force of friction. The heavy ball method is a kind of multi-step methods. It brings in more information of the previous cycles by just including the difference of last two approximate solutions, which is considered to be sufficient to bring history information before $x^{(\ell)}$ into current search to make up the lost of previous search spaces. Therefore, the search space of every HBGMRES cycle is extended to be $\mathcal{K}_k(A, r_0^{(\ell)}) + \text{span}(x_0^{(\ell)} - x_0^{(\ell-1)})$. The framework of HBGMRES is presented in Algorithm 2.3.

Algorithm 2.3 HBGMRES(k)

Given any initial guess $x_0^{(1)} \in \mathbb{R}^n$ and an integer $k \geq 1$, this algorithm computes an approximate solution to the linear system $Ax = b$ via the heavy ball GMRES.

$r_0^{(1)} = b - Ax_0^{(1)}$, $x_0^{(0)} = 0$

for $\ell = 1, 2, \ldots,$ **do**

  if $\|r_0^{(\ell)}\|_2 \leq \texttt{tol} \times (\|A\|_1 \|x_0^{(\ell)}\| + \|b\|_2)$ **then**

    break

  else

    compute

$$x_k^{(\ell)} = x_0^{(\ell)} + \arg \min_{z \in \mathcal{K}_k(A, r_0^{(\ell)}) + \mathrm{span}(x_0^{(\ell)} - x_0^{(\ell-1)})} \|b - A(x_0^{(\ell)} + z)\|_2 \tag{2.4}$$

    $x_0^{(\ell+1)} = x_k^{(\ell)}$

    $r_0^{(\ell+1)} = b - Ax_0^{(\ell+1)}$

  end if

end for

return $x_0^{(\ell)}$ as the computed solution to $Ax = b$

Algorithm 2.4 Arnoldi process for (2.5)

Given initial guess x_0, set $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$ and $v_1 = r_0/\beta$

for $j = 1, 2, \ldots, k$ **do**

  $V_{(:,1)} = v_1$, $\check{H} = 0_{(k+1)\times\ell}$ (the $(k+1) \times k$ zero matrix)

  $z_j = M^{-1} V_{(:,j)}$

  $f = Az_j$

  $\check{H}_{(1:j,j)} = V_{(:,1:j)}^T f$, $f = f - V_{(:,1:j)} \check{H}_{(1:j,j)}$

  $\check{H}_{(j+1,j)} = \|f\|_2$

  $V_{(:,j+1)} = f/\check{H}_{(j+1,j)}$

end for

2.1.3 Flexible GMRES

Flexible GMRES is a variant of the GMRES algorithm presented by Saad [53] in 1993. Given a nonsingular matrix M, the right preconditioned linear system is

$$AM^{-1}(Mx) = b. \tag{2.5}$$

The Arnoldi process can be modified to work for (2.5) as shown in Algorithm 2.4. The blue parts show the modification from the original Arnoldi process. After the k-th iteration, the approximate solution is $x_k = x_0 + M^{-1}V_k y_k$, where $y_k = \min_{y \in \mathbb{R}^k} \|\beta e_1 - \check{H}_k y\|_2$. It's clear that the search Krylov subspace is

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

In every iteration of the Arnoldi process, we don't explicitly form M^{-1}, but solve a linear system like $Mz = v$. We don't require the exact z but an approximate solution solving by any linear solver within a given tolerance. This means that the preconditioner M is not "constant" but allowed to vary from one to another in the outer iteration, which inspires the main idea of the flexible GMRES. The flexible GMRES selects different M for each Arnoldi iteration by defining $z_i = M_i^{-1}v_i$ and solving $M_i z_i = v_i$ in the inner iteration. After the k-th outer iteration, the approximate solution is $x_k = x_0 + Z_k y_k$, where $y_k = \arg\min_y \|\beta e_1 - \check{H}y\|_2$. The blue part in Algorithm 2.5 shows the difference from Algorithm 2.4. Another similar Krylov subspace method with inner-outer iteration is GMRESR [17] which consists of GCR as the outer iteration and GMRES as the inner iteration. For FGMRES, if the inner linear system is chosen to be $Az = v$, i.e. M^{-1} is an approximate of A^{-1}, and applying GMRES as the inner solver, FGMRES and GMRESR can obtain different but comparable solutions of the original linear system [65]. In what follows, we will present a heavy ball flexible GMRES (HBFGMRES), inspired by HBGMRES and FGMRES.

Algorithm 2.5 Flexible GMRES

Given any initial guess $x_0 \in \mathbb{R}^n$ and an integer $\ell \geq 1$, this algorithm computes an approximate solution to the linear system $Ax = b$ via FGMRES.

1: $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$ and $v_1 = r_0/\beta$
2: **for** $j = 1, 2, \ldots, k$ **do**
3: $V_{(:,1)} = v_1$, $\check{H} = 0_{(k+1)\times k}$ (the $(k+1) \times k$ zero matrix)
4: $z_j = M_j^{-1} V_{(:,j)}$
5: $f = Az_j$
6: $\check{H}_{(1:j,j)} = V_{(:,1:j)}^T f$ $f = f - V_{(:,1:j)} \check{H}_{(1:j,j)}$
7: $\check{H}_{(j+1,j)} = \|f\|_2$
8: **if** $\check{H}_{(j+1,j)} > 0$ **then**
9: $V_{(:,j+1)} = f/\check{H}_{(j+1,j)}$
10: **else**
11: reset $k = j$, $\check{H} = \check{H}_{(1:j,1:j)}$
12: **break**
13: **end if**
14: **end for**
15: $Z_k = [z_1, \cdots, z_k]$
16: $y_k = \arg\min_y \|\beta e_1 - \check{H} y\|_2$
17: **return** $x_k = x_0 + Z_k y_k$ as the computed solution to $Ax = b$

2.2 Motivation and Main Idea

In Table 2.1, we list the numbers of flops for GMRES and its variants we introduced in section 2.1, where (MV) is the number of flops by one matrix-vector multiplication with A. We take it to be twice the number of nonzeros entries in A. For FGMRES, we choose the inner solver to be m-step GMRES. For simplicity, only

the leading terms of flops by three major actions within each inner iteration are kept: matrix-vector multiplications, orthogonalization, and solutions of the reduced least squares problems.

Table 2.1: Flops of GMRES Variants

GMRES of k-steps	$(k+1)(\text{MV})+2k^2n+4k^2$
per cycle of REGMRES(k)	$(k+1)(\text{MV})+2k^2n+4k^2$
per cycle of HBGMRES(k)	$(k+2)(\text{MV})+2(k+2)^2n+4(k+1)^2$
FGMRES of k-steps	$(k+m+2)(\text{MV})+2(k^2+m^2)n+4(k^2+m^2)$

We compare the restarted GMRES, heavy ball GMRES and flexible GMRES with a comparable computation cost. Our numerical examples in Figure 2.1 show that FGMRES has the best performance among all. So it's more meaningful to optimize FGMRES.

From Algorithm 2.5 and Table 2.1, we can see that as k increases, FGMRES requires more orthogonalization cost and memory for z_k and V_k. In addition, the orthogonality of V_k may be lost in FGMRES which will slow down the convergence in a way. Similarly to GMRES, the restarted FGMRES can be implemented to control the memory usage and reorthogonalization cost of V_k by fixing k. Meanwhile, in order to make up the slow convergence in FGMRES, we can include more Krylov subspace information of previous cycles by adding the difference of two previous solutions to extend the search space, i.e., another direct application of the heavy ball idea. We call this new method the heavy ball flexible GMRES (HBFGMRES).

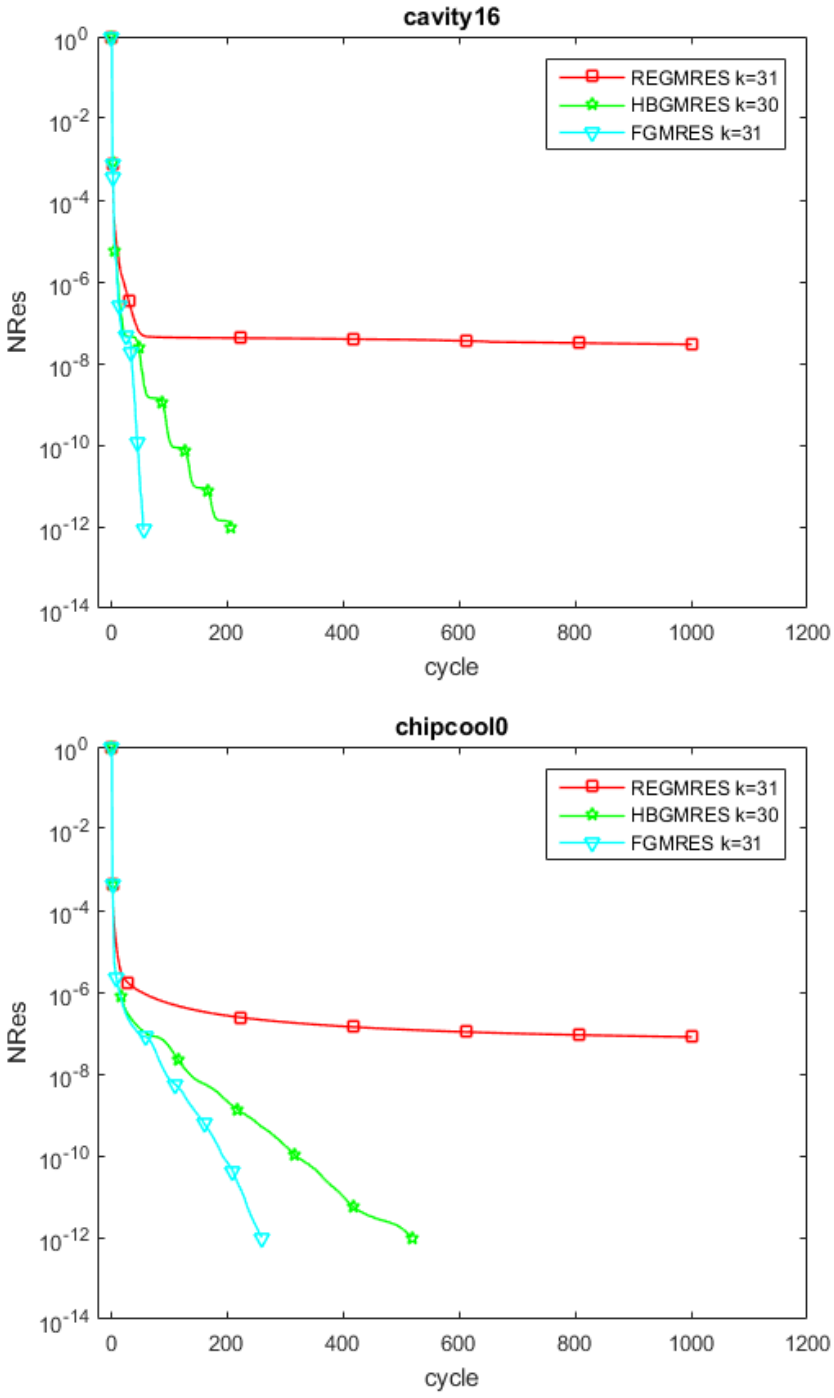

Figure 2.1: FGMRES *vs.* HBGMRES *vs.* REGMRES for `cavity16` and `chipcool0`

2.3 Heavy Ball FGMRES Method

Suppose that we apply a certain m-step solver to solve the inner linear system $M_i v_i = z_i$. In the k-step restarted FGMRES (REFGMRES), the solution can be expressed as

$$x_k^{(\ell)} = x_0^{(\ell)} + Z_k y_k \in x_0^{(\ell)} + \text{span}\,(Z_k).$$

Add $x_d = x_0^{(\ell)} - x_0^{(\ell-1)}$ to the search space span (Z_k). The solution can be written as

$$x_k^{(\ell)} = x_0^{(\ell)} + Z_k y_k + \alpha x_d.$$

Then, the ℓ-th residual becomes

$$\begin{aligned} r_k^{(\ell)} &= b - A x_k^{(\ell)} \\ &= r_0^{(\ell)} - V_{k+1}\check{H} y_k - \alpha A x_d. \end{aligned} \tag{2.6}$$

Denote by $p = Ax_d$. If $p = 0$, this case is exactly the restarted FGMRES. We assume $p \neq 0$. Now we orthogonalize p against V_{k+1} , and define

$$d = V_{k+1}^T p, \quad h = p - V_{k+1} d.$$

Case 1 $h \neq 0$

This is the most generic and common case. Define

$$\check{v}_{k+2} = h/\|h\|_2, \check{V}_{k+2} = [V_{k+1} \quad \check{v}_{k+2}].$$

We have

$$\begin{aligned} r_k^{(\ell)} &= r_0^{(\ell)} - V_{k+1}\check{H} y_k - \alpha A x_d \\ &= r_0^{(\ell)} - V_{k+1}\check{H} y_k - \alpha(\check{v}_{k+2}\|h\|_2 + V_{k+1} d) \\ &= r_0^{(\ell)} - \check{V}_{k+2} \begin{bmatrix} \check{H} & d \\ 0 & \|h\|_2 \end{bmatrix} \begin{bmatrix} y_k \\ \alpha \end{bmatrix}. \end{aligned}$$

By solving the following least squares problem

$$\min_{y,\alpha} \left\| \beta e_1 - \begin{bmatrix} \check{H} & d \\ 0 & \|h\|_2 \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} \right\|_2, \tag{2.7}$$

where $\beta = \|r_0^{(\ell)}\|_2$, we can obtain the optimal y_k and α_{opt} to form the approximate solution $x_k^{(\ell)}$ that satisfies

$$x_k^{(\ell)} - x_0^{(\ell)} \in \text{span}(Z_k) + \text{span}(x_d)$$

in each cycle.

Case 2 $h = 0$

In this case, we have

$$\|h\|_2 = 0 \quad \text{and} \quad Ax_d = V_{k+1}d.$$

Then

$$\begin{aligned} r_k^{(\ell)} &= r_0^{(\ell)} - V_{k+1}\check{H}y_k - \alpha Ax_d \\ &= r_0^{(\ell)} - V_{k+1}\check{H}y_k - \alpha(V_{k+1}d) \\ &= r_0^{(\ell)} - V_{k+1}\begin{bmatrix} \check{H} & d \end{bmatrix}\begin{bmatrix} y_k \\ \alpha \end{bmatrix}. \end{aligned}$$

We need to solve

$$\min_{y,\alpha} \left\| \beta e_1 - \begin{bmatrix} \check{H} & d \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} \right\|_2. \tag{2.8}$$

The details of HBFGMRES is shown in Algorithm 2.6.

Algorithm 2.6 Heavy Ball FGMRES

Given any initial guess $x_0 \in \mathbb{R}^n$ and dimension $k \geq 1$ of the Krylov subspace. This algorithm computes an approximate solution to the linear system $Ax = b$ via the heavy ball FGMRES.

$r_0^{(1)} = b - Ax_0^{(1)}$, $\beta = \|r_0^{(1)}\|_0$ and $v_1 = r_0^{(1)}/\beta$
for $\ell = 1, 2, \ldots,$ **do**
  if $\|r_0^{(\ell)}\|_2 \leq \texttt{tol} \times (\|A\|_1 \|x_0^{(\ell)}\|_2 + \|b\|_2)$ **then**
    break
  else
    $V_{(:,1)} = v_1$, $\check{H} = 0_{(k+1)\times k}$ (the $(k+1) \times k$ zero matrix)
    for $j = 1, 2, \ldots, k$ **do**
      $z_j = M_j^{-1} V_{(:,j)}$
      $f = Az_j$
      $\check{H}_{(1:j,j)} = V_{(:,1:j)}^H f$, $f = f - V_{(:,1:j)} \check{H}_{(1:j,j)}$
      $\check{H}_{(j+1,j)} = \|f\|_2$
    end for
    $Z_k = [z_1, \cdots, z_k]$
    (y_k, α) is the solution of (2.7) or (2.8)
    $x_0^{(\ell+1)} = x_0^{(\ell)} + Z_k y_k + \alpha x_d$
  end if
end forreturn $x_0^{(\ell)}$ as a computed solution to $Ax = b$

2.4 Numerical Experiments

We present several numerical tests to compare the restarted FGMRES (REFGMRES) and the heavy ball FGMRES (HBFGMRES). The error measurement is the normalized residual defined as

$$\text{NRes} = \frac{\|r\|_2}{\|A\|_1 \times \|x\|_2 + \|b\|_2}.$$

Numerical results show NRes against cycles. Test matrices are taken from SuiteSparse Matrix Collection [2] and Matrix Market [1]. Each comes with a right-hand side b. Table 2.2 lists 8 testing examples and their characteristic, where n is the size of matrix, `nnz` is the number of nonzero entries, and sparsity is $\texttt{nnz}/n^2$. These are representatives of our numerical examples.

Table 2.2: Testing Matrices

matrix	n	`nnz`	sparsity	application
`cavity10`	2597	76367	0.0113	computational fluid dynamics
`cavity16`	4562	137887	0.0066	computational fluid dynamics
`comsol`	1500	97645	0.0434	structural problem
`chipcool0`	20082	281150	6.9715e-04	model reduction problem
`flowmeter5`	9669	67391	7.2084e-04	computational fluid dynamics
`e20r0000`	4241	131413	0.0073	computational fluid dynamics
`e20r0100`	4241	131413	0.0073	computational fluid dynamics
`sherman3`	5005	20033	7.9972e-04	computational fluid dynamics

In order to fairly compare algorithms and make it easy to understand, we use the following testing scenarios:

- We run REFGMRES($k+1$) and HBFGMRES(k) in order to best illustrate convergence against cycle indices. Here are two considerations for this choice. One is to make sure that all approximate solutions at a cycle are computed from a subspace of dimension $k+1$: $\text{span}(x_0) + \text{span}(Z_{k+1})$ for REFGMRES($k+1$),

and $\text{span}(x_0) + \text{span}(Z_k) + \text{span}(x_d)$ for HBFGMRES(k). Secondly, we want to align the costs per cycle for both algorithms.

- In FGMRES, M_i^{-1} is a flexible preconditioner which changes for each iteration. If M_i^{-1} is a flexible approximate of A^{-1}, then it might be a good preconditioner. Based on this idea, we choose to solve $Az = v$ by m-step GMRES to construct a preconditioner to approximate A^{-1}, i.e., using a degree $(m-1)$ polynomial $P_m(A)$ [26] to approximate A^{-1}. Of course, other solvers mentioned in chapter 1 can be also used to solve $Az = v$.

Numerical results of testing examples are listed in Table 2.3. For each matrix, we ran different k and m. Table 2.3 lists the number of cycles needed by algorithms to achieve NRes less than or equal to 10^{-12}. In the table, RE and HB stand for REFGMRES and HBFGMRES. The table clearly demonstrates huge savings achieved by HBFGMRES over REFGMRES.

We also plot Figures 2.2–2.9 to show how NRes in computed solutions moves against the cycle index. All figures can give us an impression as how each algorithm behaves. Together with Table 2.3, we make the following observations.

- HBFGMRES is faster than REFGMRES for certain k and m.
- There is little difference in the number of cycles with always reorthogonalization or selective reorthogonalization. For HBFGMRES, the number of cycles is smaller for most of the examples. These two points make selective reorthogonalziation a better choice for cost consideration.

Table 2.3: Number of Cycles

matrix	always reorth		selective reorth	
	RE	HB	RE	HB
cavity10	107	38	110	38
cavity16	179	183	216	142
comsol	37	17	55	15
chipcool0	70	21	66	21
flowmeter5	433	333	458	330
e20r0000	400	14	639	14
e20r0100	47	13	54	14
sherman3	249	42	248	44

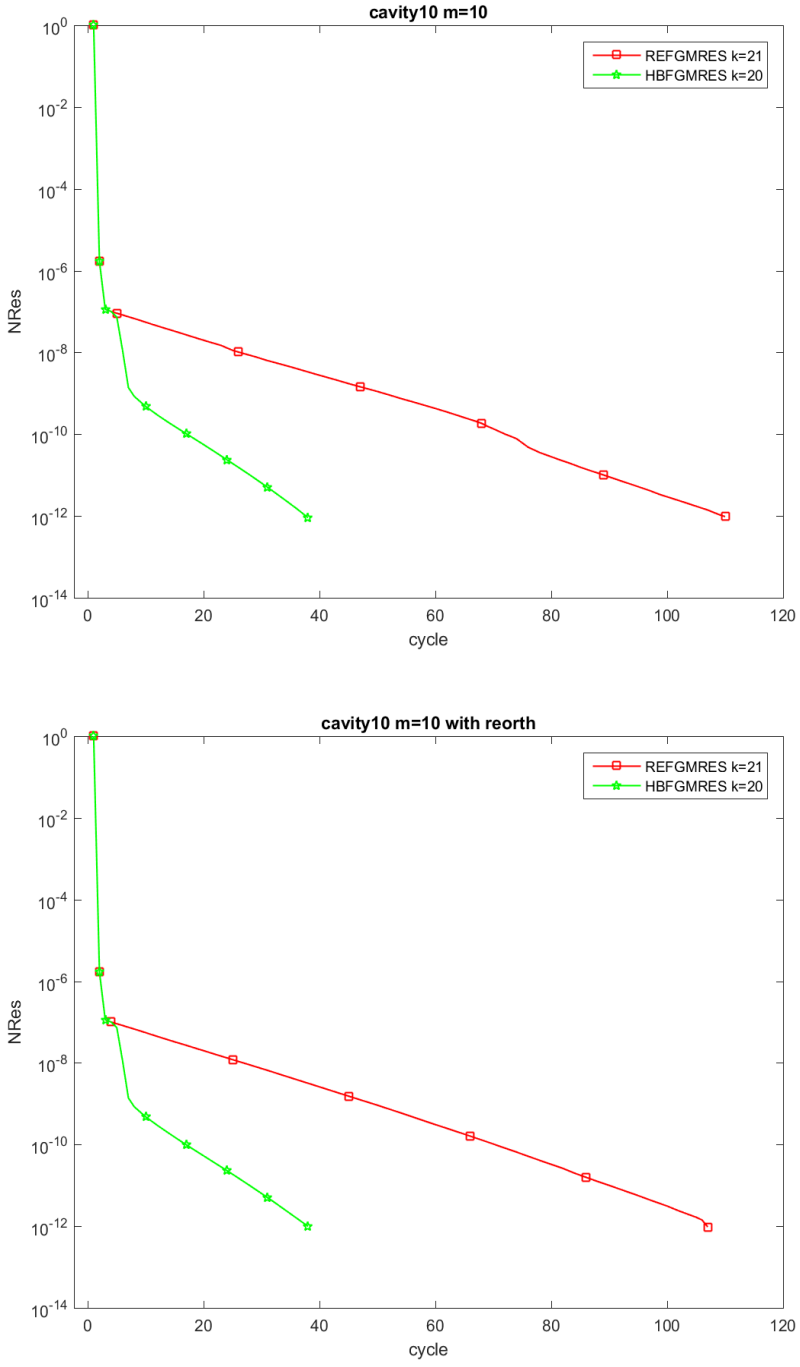

Figure 2.2: NRes *vs.* cycle for `cavity10`. *Top* : selective reorthogonalization; *Bottom* : always reorthogonalization

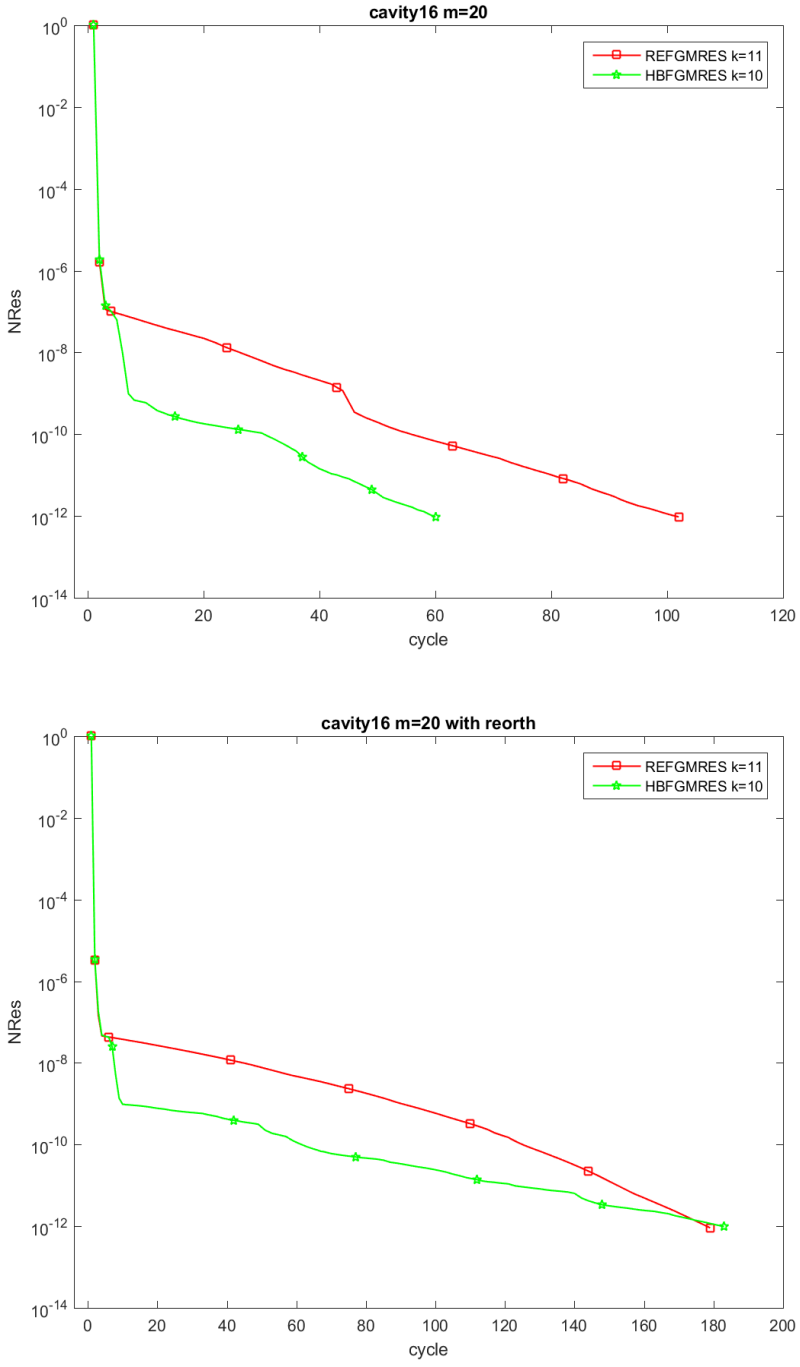

Figure 2.3: NRes *vs.* cycle for `cavity16`. *Top* : selective reorthogonalization; *Bottom* : always reorthogonalization

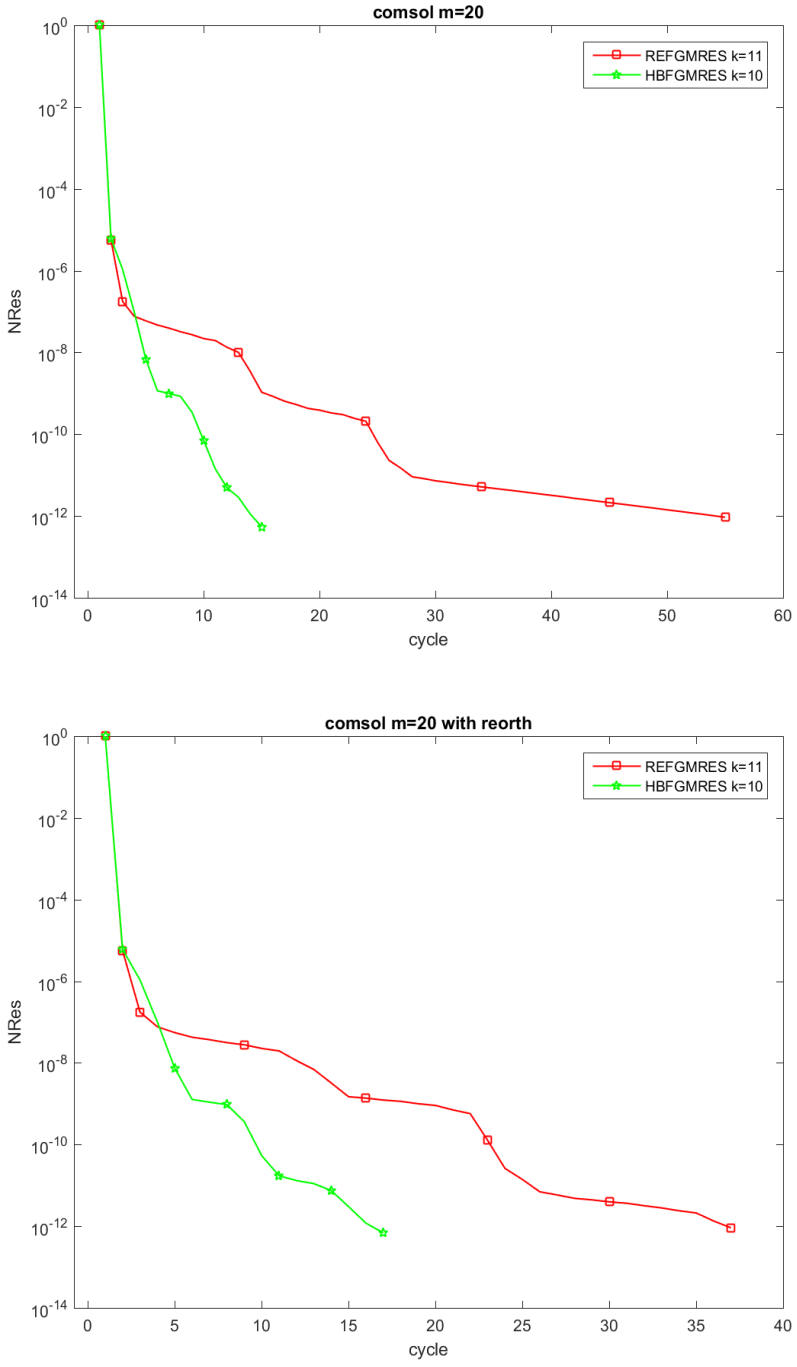

Figure 2.4: NRes *vs.* cycle for `comsol`. *Top* : selective reorthogonalization; *Bottom* : always reorthogonalization

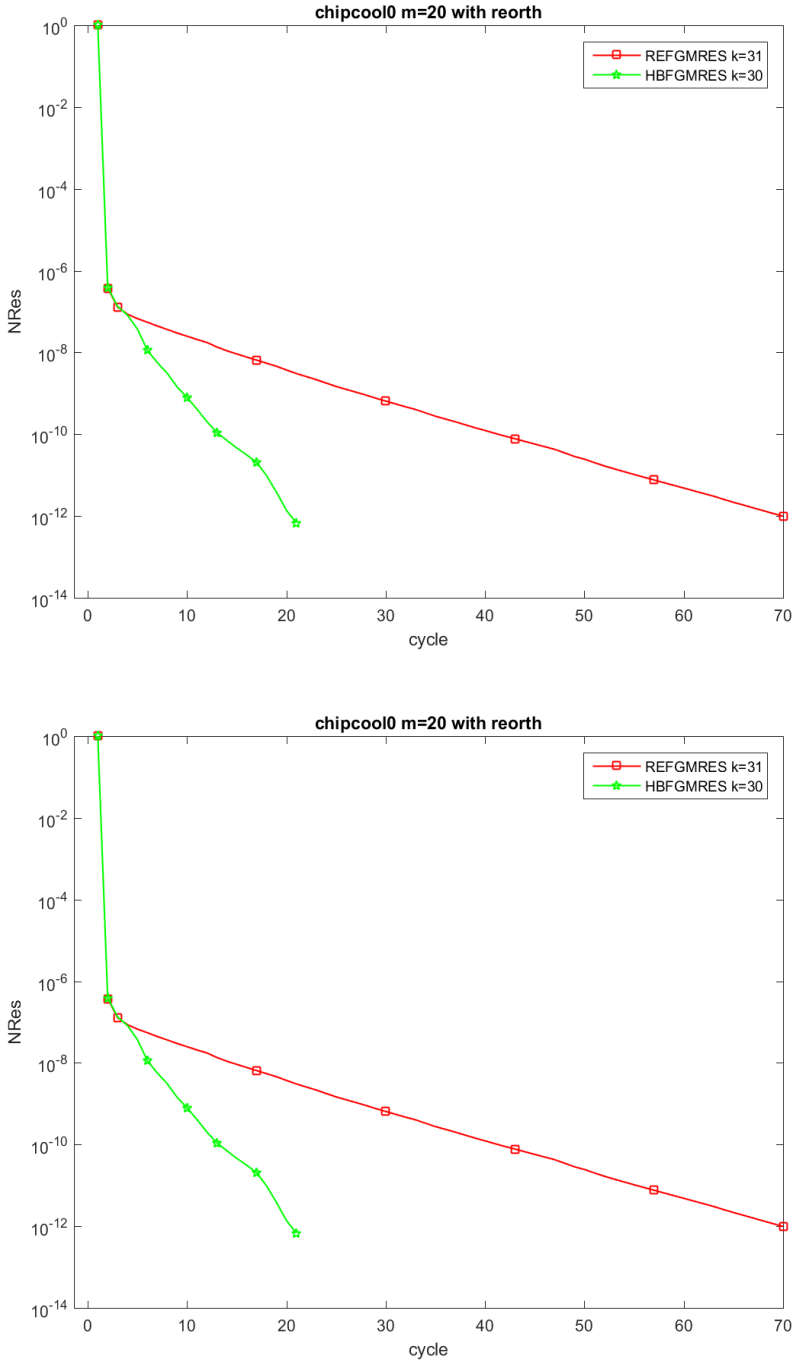

Figure 2.5: NRes *vs.* cycle for `chipcool0`. *Top* : selective reorthogonalization; *Bottom* : always reorthogonalization

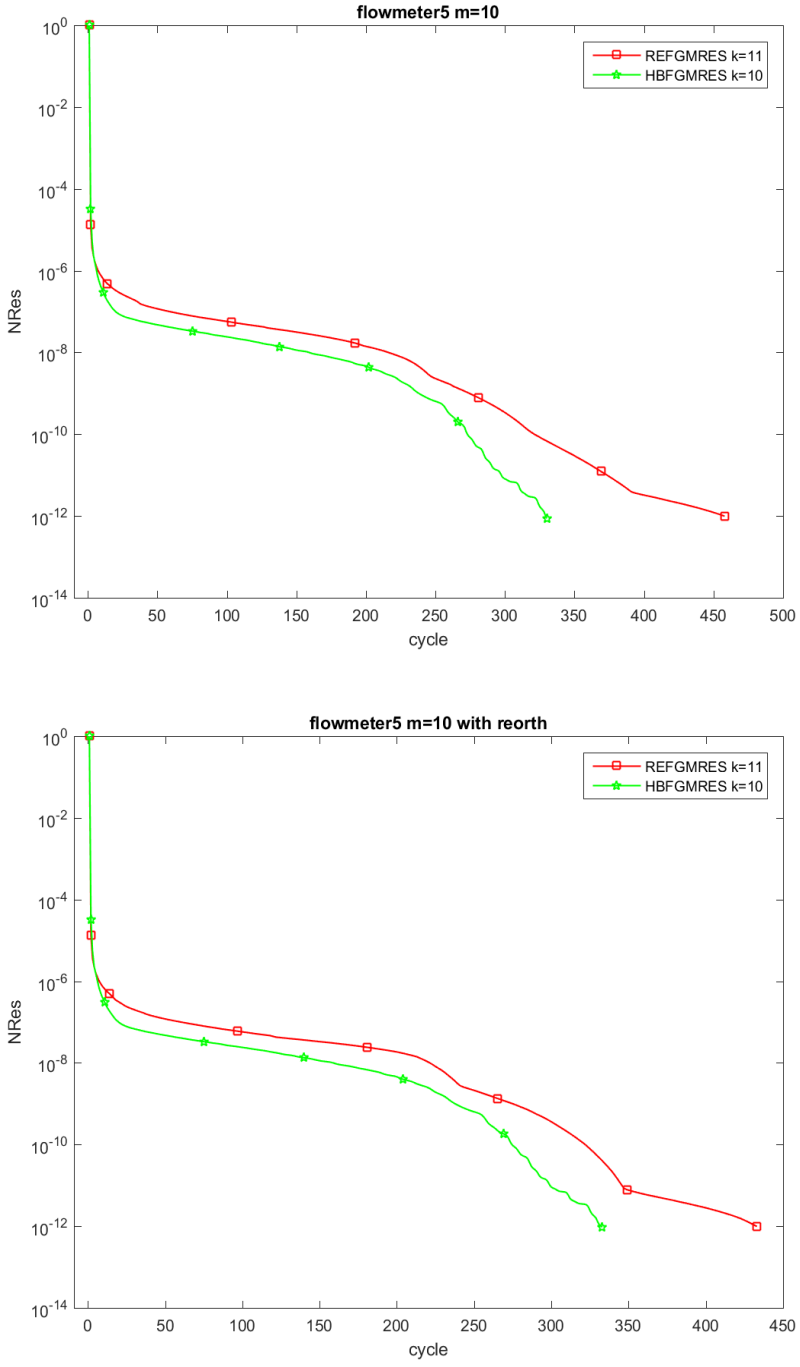

Figure 2.6: NRes *vs.* cycle for `flowmeter5`. *Top* : selective reorthogonalization; *Bottom* : always reorthogonalization

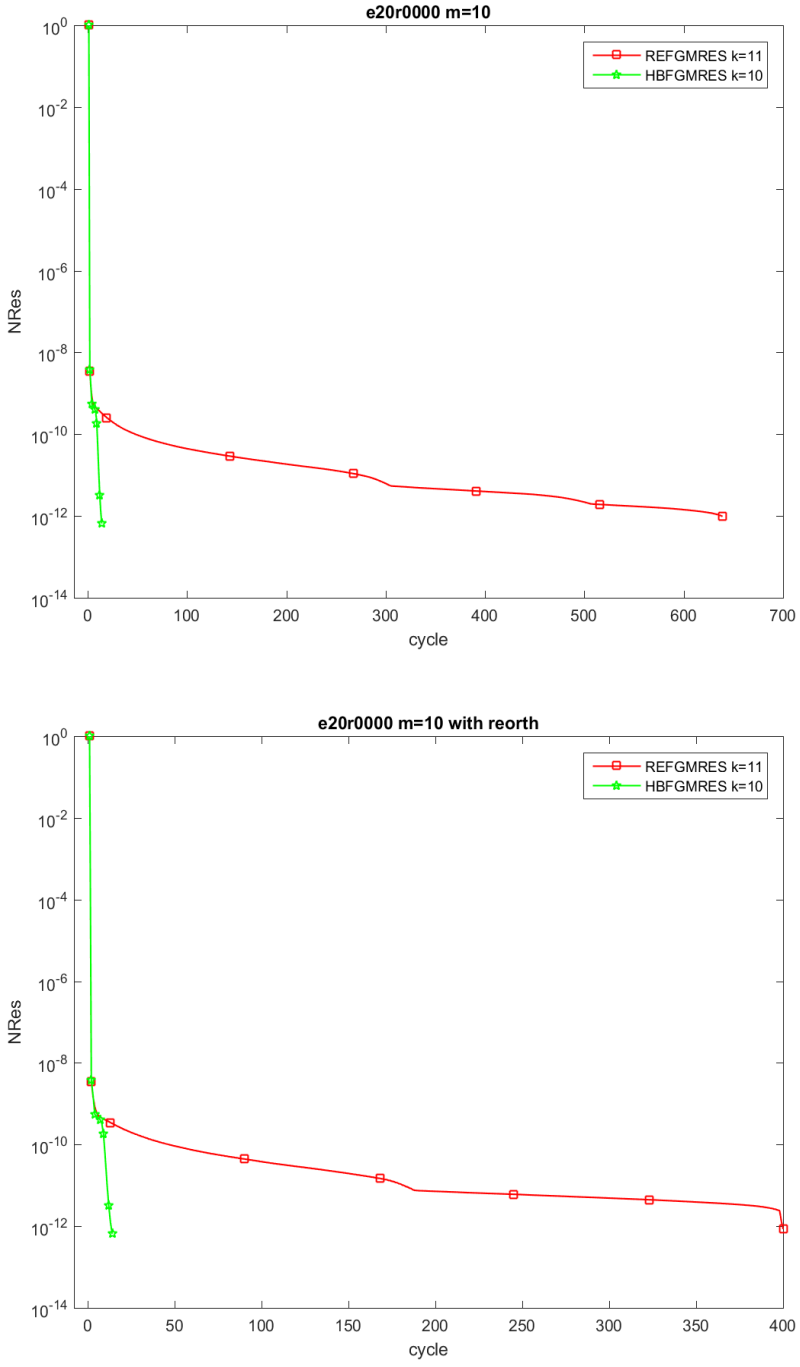

Figure 2.7: NRes *vs.* cycle for `e20r0000`. *Top* : selective reorthogonalization; *Bottom* : always reorthogonalization

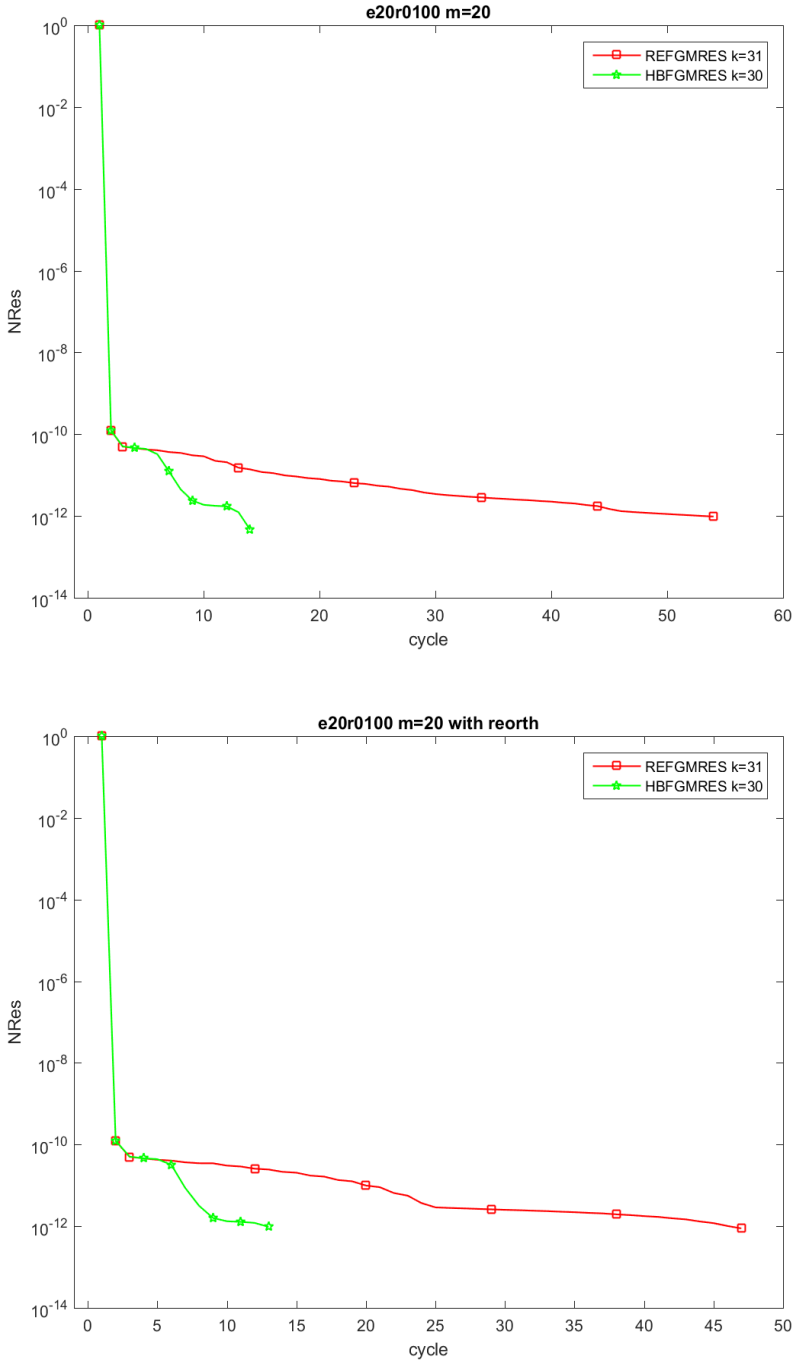

Figure 2.8: NRes *vs.* cycle for `e20r0100`. *Top* : selective reorthogonalization; *Bottom* : always reorthogonalization

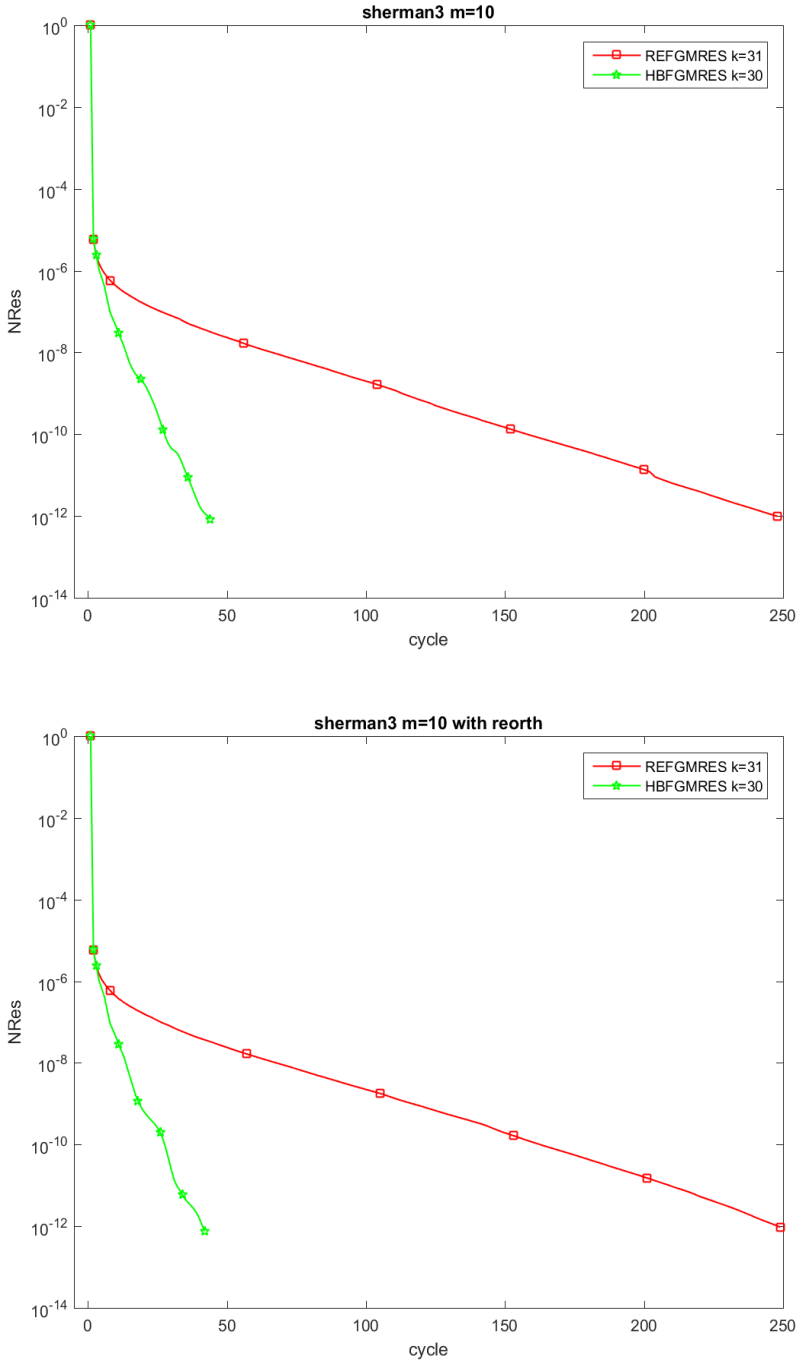

Figure 2.9: NRes *vs.* cycle for `sherman3`. *Top* : selective reorthogonalization; *Bottom* : always reorthogonalization

CHAPTER 3

HEAVY BALL MINIMAL RESIDUAL METHOD FOR LEAST SQUARES PROBLEMS

Inspired by the heavy ball GMRES (HBGMRES) [37] method for nonsymmetric linear systems, which successfully integrates the heavy ball method and GMRES [55], we proposed the heavy ball minimal residual method (HBMR). HBMR utilizes the restart technique to combine the LSMR method with the heavy ball method. In section 3.1, the derivation of HBMR is shown. In section 3.2, we mathematically derive HBMR. Section 3.3 shows the pseudo-code of the extended LSMR for solving the new subproblem arising in HBMR. Numerical experiments in section 3.4 illustrates that our method works often better than LSMR. The backward error [40] results indicate HBMR is numerically stable.

3.1 Motivation and Main Idea

Let's review the overdetermined least squares problem (LS)

$$\min_x \|Ax - b\|_2 , \tag{3.1}$$

where $A \in \mathbb{R}^{m\times n}, m > n$, $b \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$. We denote the residual vector by $r = b - Ax$.

Let $E = \{b - Ax | x \in \mathbb{R}^n\}$. This set is non-empty, closed and convex. So there exists a unique global minimum point in E. This means there exists $x \in \mathbb{R}^n$ such that $\min_x \|b - Ax\|_2$ is minimized, but x is not unique when A is not full column rank.

Lemma 3.1.1. [50] *Let* $\mathcal{S} = \{x \in \mathbb{R}^n | \min = \|b - Ax\|_2\}$ *be the set of solutions, and let* $r_x = b - Ax$ *denote the residual for a specific* x*. Then*

$$x \in \mathcal{S} \Longleftrightarrow A^T r_x = 0 \Longleftrightarrow r_x \perp \mathcal{R}(A),$$

where $\mathcal{R}(A)$ *represents the range of* A*.*

From lemma 3.1.1, we know that the solutions of LS are the same as the solutions of the normal equation

$$A^T Ax = A^T b. \tag{3.2}$$

This suggests two ways for solving a least squares problem. One is directly applying all well-known methods to (3.2) such as CG and MINRES. Another way is finding new methods to solve (3.2) such as LSQR by minimizing $\|r\|_2$ and LSMR by minimizing $\|A^T r\|_2$. As introduced in chapter 1, the Golub-Kahan bidiagonalization process partially reduces the matrix A into a lower bidiagonal matrix B_k by following iterative process:

$$\beta_{k+1}u_{k+1} = Av_k - \alpha_k u_k \ \text{ and } \ \alpha_{k+1}v_{k+1} = A^T u_{k+1} - \beta_{k+1}v_k,$$

where $\beta_1 u_1 = r_0$, $\beta_1 = \|r_0\|_2$, $\alpha_1 v_1 = A^T u_1$, and $\alpha_1 = \|A^T u_1\|_2$. After k steps, we have

$$AV_k = U_{k+1}B_k, \ A^T U_{k+1} = V_k B_k^T + \alpha_{k+1}v_{k+1}e_{k+1}^T.$$

Given $u_1 = b/\|b\|_2$, LSMR seeks the optimal approximate solutions of (3.1) over the Krylov subspace

$$\mathcal{K}_k(A^T A, A^T u_1) = \text{span}\left(A^T u_1, A^T A(A^T u_1), \cdots, (A^T A)^{k-1}(A^T u_1)\right), \tag{3.3}$$

with the starting vector $A^T u_1$, which is the same as the subspace span(V_k).

Theoretically, when the iteration number k is n, the optimal approximation over the subspace $\mathcal{K}_n(A^T A, A^T u_1)$ is the exact solution of LS. However, in practice

we cannot get the global optimal approximate solution as expected due to rounding errors [35, 16] and the loss of orthogonality of V_k, which slows the convergence. If reorthogonalizing v_{j+1} with $v_i, i = 1, 2, \ldots, j$ in every iteration, it will cost too much storage of memory in keeping all the previous vectors v_i as well as computations of reorthogonalization. A good way to cope with these shortcomings is to use a restarted process.

Simple restarting strategy ignores the information of previous Krylov subspaces. Inspired by the heavy ball GMRES method [37], we apply the heavy ball idea to the restarted LSMR to make up this problem so as to accelerate convergence.

In summary, we use the restarted Golub-Kahan process to control the memory and reorthogonalization cost by fixing k. Let ℓ be the index of the restarted process. At the same time, in each cycle we keep some information of the approximate solution $x_k^{(\ell)}$ and $x_k^{(\ell-1)}$ from the previous two cycles.

3.2 Heavy Ball Minimal Residual Method

Given initial guess x_0, denote by $r_0 = b - Ax_0$. Write the solution of normal equation (3.2) x as $x = x_0 + \Delta x$. Then

$$A^T r = A^T(b - Ax) = A^T(r_0 - A\Delta x).$$

Apply the k-step Golub-Kahan process to approximate x with $x_k^{(\ell)}$ at each cycle $\ell = 1, 2, \ldots$. That is

$$x_k^{(\ell)} = x_0^{(\ell)} + \Delta x,$$

where $x_0^{(\ell)} = x_k^{(\ell-1)}$.

Denote by $r_0^{(\ell)} = b - Ax_0^{(\ell)}$. Solve for Δx:

$$\Delta x = \arg\min_z \left\| A^T \left(r_0^{(\ell)} - Az \right) \right\|_2.$$

The optimization is done over the Krylov subspace $\mathcal{K}_k\left(A^TA, A^T\hat{u}_1\right)$, where $\hat{u}_1 = r_0^{(\ell)}/\|r_0^{(\ell)}\|_2$. The solution of the restarted LSMR $x_k^{(\ell)}$ satisfies

$$x_k^{(\ell)} - x_0^{(\ell)} \in \mathcal{K}_k\left(A^TA, A^T\hat{u}_1\right).$$

Extend this subspace by adding the vector $x_d \equiv x_0^{(\ell)} - x_0^{(\ell-1)}$. The optimal solution $x_k^{(\ell)}$ of HBMR is in

$$x_0^{(\ell)} + \mathcal{K}_k\left(A^TA, A^T\hat{u}_1\right) + \text{span}\left(x_d\right).$$

Therefore

$$x_k^{(\ell)} = x_0^{(\ell)} + V_k y_k + \alpha x_d.$$

For simplicity, we still use $V_k, U_{k+1}, B_k, \alpha_k$, and β_k to denote the corresponding matrices and vectors after the k-step Golub-Kahan process with starting vector $\hat{u}_1 = r_0^{(\ell)}/\|r_0^{(\ell)}\|_2$. For every cycle, we have

$$\begin{aligned} A^Tr^{(\ell)} &= A^Tb - A^TAx_k^{(\ell)} \\ &= A^Tb - A^TA\left(x_0^{(\ell)} + V_k y_k + \alpha x_d\right) \\ &= A^Tr_0^{(\ell)} - A^TAV_k y_k - A^TA\alpha x_d \\ &= A^Tr_0^{(\ell)} - V_{k+1}\begin{bmatrix} B_k^TB_k \\ \alpha_{k+1}\beta_{k+1}e_k^T \end{bmatrix} y_k - \alpha A^TAx_d. \end{aligned}$$

Assume $A^TAx_d \neq 0$. We orthogonalize A^TAx_d against V_{k+1} to define

$$\hat{h} = V_{k+1}^TA^TAx_d, \quad \hat{g} = A^TAx_d - V_{k+1}\hat{h}.$$

We need to consider if $\hat{g} \neq 0$ or not. That is testing if $\|\hat{g}\|_2 \leq \texttt{tol} \times \|\hat{h}\|_2$.

Case $\hat{g} \neq 0$:

This is the most generic and common case. In this case we set

$$\hat{\delta} = \|\hat{g}\|_2, \quad \hat{v}_{k+2} = \hat{g}/\hat{\delta}.$$

Then we have

$$
\begin{aligned}
A^T r_k^{(\ell)} &= A^T r_0^{(\ell)} - V_{k+1} \begin{bmatrix} B_k^T B_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{bmatrix} y_k - \alpha A^T A x_d \\
&= \beta_1 \alpha_1 v_1 - \begin{bmatrix} V_{k+1} & \hat{v}_{k+2} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} B_k^T B_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{bmatrix} & \hat{h} \\ 0 & \hat{\delta} \end{bmatrix} \begin{bmatrix} y_k \\ \alpha \end{bmatrix} \\
&= \begin{bmatrix} V_{k+1} & \hat{v}_{k+2} \end{bmatrix} \begin{bmatrix} \alpha_1 \beta_1 e_1 - \begin{bmatrix} \begin{bmatrix} B_k^T B_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{bmatrix} & \hat{h} \\ 0 & \hat{\delta} \end{bmatrix} \begin{bmatrix} y_k \\ \alpha \end{bmatrix} \end{bmatrix}.
\end{aligned}
$$

Because of the orthogonality of $[V_{k+1} \quad \hat{v}_{k+2}]$, we obtain a new subproblem

$$
\min_x \left\| A^T r \right\|_2 = \min_{y,\alpha} \left\| \alpha_1 \beta_1 e_1 - \begin{bmatrix} \begin{bmatrix} B_k^T B_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{bmatrix} & \hat{h} \\ 0 & \hat{\delta} \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} \right\|
$$

Case $\hat{g} = 0$:

In this case, we have $x_d \in \mathcal{K}_k \left(A^T A, A^T \hat{u}_1^{(\ell)} \right), \hat{u}_1^{(\ell)} = r_0^{(\ell)} / \|r_0^{(\ell)}\|_2$. Thus

$$
\begin{aligned}
x_k^{(\ell)} - x_0^{(\ell)} &\in \mathcal{K}_k \left(A^T A, A^T \hat{u}_1^{(\ell)} \right) + \operatorname{span}(\hat{g}) \\
&= \mathcal{K}_k \left(A^T A, A^T \hat{u}_1^{(\ell)} \right).
\end{aligned}
$$

So the subproblem becomes

$$
\min_x \left\| A^T r \right\|_2 = \min_{y,\alpha} \left\| \alpha_1 \beta_1 e_1 - \begin{bmatrix} \begin{bmatrix} B_k^T B_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{bmatrix} & \hat{h} \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} \right\|_2.
$$

We outline the framework of HBMR in Algorithm 3.1.

Algorithm 3.1 HBMR

1: Given initial guess $x_0^{(\ell)} \in \mathbb{R}^n$, $\ell = 1$, `tol`=10e-12, and integer $k \geq 1$. Set $x_0^{(0)} = 0$

2: $r_0^{(\ell)} = b - Ax_0^{(\ell)}$

3: **while** $\left\|A^T r_0^{(\ell)}\right\|_2 > \texttt{tol} \times \|A\|_1 \times \|r_0^{(\ell)}\|_2$ **do**

4: compute

$$x_k^{(\ell)} = \arg\min_x \left\|A^T r\right\|_2 \text{ over}$$

$$x_0^{(\ell)} + \mathcal{K}_k\left(A^T A, A^T \hat{u}_1\right) + \text{span}\left(x_0^{(\ell)} - x_0^{(\ell-1)}\right)$$

5: $x_0^{(\ell+1)} = x_k^{(\ell)}$

6: $r_0^{(\ell+1)} = b - Ax_0^{(\ell+1)}$

7: $\ell = \ell + 1$

8: **end while**

9: **return** $x_0^{(\ell)}$

3.3 Pseudo-code of Extended LSMR

In this part, we show how to modify LSMR for solving the generic subproblem in HBMR:

$$\begin{aligned}\min_x \left\|A^T r\right\|_2 &= \min_{y,\alpha} \left\| \alpha_1\beta_1 e_1 - \begin{bmatrix} \begin{bmatrix} B_k^T B_k \\ \alpha_{k+1}\beta_{k+1} e_k^T \end{bmatrix} & \hat{h} \\ 0 & \hat{\delta} \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} \right\|_2 \\ &= \min_{y,\alpha} \left\| \bar{\beta}_1 e_1 - \begin{bmatrix} \begin{bmatrix} B_k^T B_k \\ \bar{\beta}_{k+1} e_k^T \end{bmatrix} & \hat{h} \\ 0 & \hat{\delta} \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} \right\|_2,\end{aligned}$$

where $\bar{\beta}_1 = \alpha_1\beta_1$ and $\bar{\beta}_k = \alpha_k\beta_k$.

As in LSQR, we form the QR factorization of B_k as

$$Q_{k+1}B_k = \begin{bmatrix} R_k \\ 0 \end{bmatrix}, \quad R_k = \begin{bmatrix} \rho_1 & \theta_2 & & \\ & \rho_2 & \ddots & \\ & & \ddots & \theta_k \\ & & & \rho_k \end{bmatrix}.$$

Then $B_k^T B_k = R_k^T R_k$. Because B_k is lower bidiagonal, Q_{k+1} can be efficiently computed by Givens rotations [16]. At the k-th iteration, construct the Givens rotation working on rows l and $l+1$ of B_k:

$$P_l = \begin{bmatrix} I_{l-1} & & & \\ & c_l & s_l & \\ & -s_l & c_l & \\ & & & I_{k-l} \end{bmatrix}, \quad l = 1, 2, \ldots, k.$$

So $Q_{k+1} = P_k \cdots P_2 P_1$.

Solve $R_k^T q_k = \bar{\beta}_{k+1} e_k$ for q_k. We have $q_k = (\bar{\beta}_{k+1}/\rho_k)e_k = \phi_k e_k$, with $\phi_k = \bar{\beta}_{k+1}/\rho_k$. Then as in LSMR we perform the QR factorization of $\begin{bmatrix} R_k^T \\ \phi_k e_k^T \end{bmatrix}$,

$$\bar{Q}_{k+1} \begin{bmatrix} R_k^T & \bar{\beta}_1 e_1 \\ \phi_k e_k^T & 0 \end{bmatrix} = \begin{bmatrix} \bar{R}_k & z_k \\ 0 & \bar{\xi}_{k+1} \end{bmatrix} \quad \text{and} \quad \bar{R}_k = \begin{bmatrix} \bar{\rho}_1 & \bar{\theta}_2 & & \\ & \bar{\rho}_2 & \ddots & \\ & & \ddots & \theta_k \\ & & & \bar{\rho}_k \end{bmatrix},$$

where $\bar{Q}_{k+1} = \bar{P}_k \cdots \bar{P}_2 \bar{P}_1$ and $\bar{P}_l$ is defined similarly to P_l. We define $t_k = R_k y$. Then

$$\begin{aligned}
&\min_{y,\alpha} \left\| \bar{\beta}_1 e_1 - \begin{bmatrix} \begin{bmatrix} B_k^T B_k \\ \bar{\beta}_{k+1} e_k^T \end{bmatrix} & \hat{h} \\ 0 & \hat{\delta} \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} \right\|_2 \\
&= \min_{y,\alpha} \left\| \bar{\beta}_1 e_1 - \begin{bmatrix} \begin{bmatrix} R_k^T R_k \\ q_k^T R_k \end{bmatrix} & \hat{h} \\ 0 & \hat{\delta} \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} \right\|_2 \\
&= \min_{y,\alpha} \left\| \bar{\beta}_1 e_1 - \begin{bmatrix} \begin{bmatrix} R_k^T \\ q_k^T \end{bmatrix} & \hat{h} \\ 0 & \hat{\delta} \end{bmatrix} \begin{bmatrix} R_k y \\ \alpha \end{bmatrix} \right\|_2 .
\end{aligned}$$

Denote by

$$\hat{Q}_{k+1} = \begin{bmatrix} \bar{Q}_{k+1} & 0 \\ 0 & 1 \end{bmatrix} .$$

So

$$\hat{Q}_{k+1} \begin{bmatrix} \bar{\beta}_1 e_1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \bar{Q}_{k+1} \begin{bmatrix} \bar{\beta}_1 e_1 \\ 0 \end{bmatrix} \\ 0 \end{bmatrix} = \begin{bmatrix} z_k \\ \bar{\xi}_{k+1} \\ 0 \end{bmatrix} ,$$

and

$$\hat{Q}_{k+1} \begin{bmatrix} \begin{bmatrix} R_k^T \\ \phi_k e_k^T \end{bmatrix} & \hat{h} \\ 0 & \hat{\delta} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \bar{R}_k \\ 0 \end{bmatrix} & \bar{Q}_{k+1} \hat{h} \\ 0 & \hat{\delta} \end{bmatrix} .$$

Because $\hat{Q}_{k+1}$ is orthogonal, the subproblem can be simplified as

$$\min_{t,\alpha} \left\| \begin{bmatrix} z_k \\ \bar{\xi}_{k+1} \\ 0 \end{bmatrix} - \begin{bmatrix} \begin{bmatrix} \bar{R}_k \\ 0 \end{bmatrix} & \bar{Q}_{k+1} \hat{h} \\ 0 & \hat{\delta} \end{bmatrix} \begin{bmatrix} t \\ \alpha \end{bmatrix} \right\|_2 ,$$

where t and y are related by $t = R_k y$. Denote by $\bar{Q}_{k+1}\hat{h} = (\hat{h}_{11}, \hat{h}_{12}, \ldots, \hat{h}_{1,k+1})^T \equiv (\hat{h}_k^T, \hat{h}_{1,k+1})^T$. Now, we form the last Givens rotation $\hat{P}_{k+1}$ to eliminate δ. We have

$$\hat{P}_{k+1} \begin{bmatrix} \bar{Q}_{k+1}\hat{h} \\ \delta \end{bmatrix} = \hat{P}_{k+1} \begin{bmatrix} \hat{h}_k \\ \hat{h}_{1,k+1} \\ \delta \end{bmatrix} = \begin{bmatrix} \hat{h}_k \\ \tilde{h}_{1,k+1} \\ 0 \end{bmatrix}.$$

and

$$\hat{P}_{k+1} \begin{bmatrix} z_k \\ \bar{\xi}_{k+1} \\ 0 \end{bmatrix} = \begin{bmatrix} z_k \\ \hat{\xi}_{k+1} \\ \hat{\xi}_{k+2} \end{bmatrix},$$

Therefore, the subproblem becomes

$$\min_{t,\alpha} \left\| \begin{bmatrix} z_k \\ \hat{\xi}_{k+1} \\ \hat{\xi}_{k+2} \end{bmatrix} - \begin{bmatrix} \bar{R}_k & \hat{h}_k \\ 0 & \tilde{h}_{1,k+1} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} t \\ \alpha \end{bmatrix} \right\|_2.$$

Finally, the subproblem is solved by choosing t and α from

$$\begin{bmatrix} z_k \\ \hat{\xi}_{k+1} \end{bmatrix} = \begin{bmatrix} \bar{R}_k & \hat{h}_k \\ 0 & \tilde{h}_{1,k+1} \end{bmatrix} \begin{bmatrix} t \\ \alpha \end{bmatrix}. \tag{3.4}$$

From the above equation, we find

$$\alpha = \hat{\xi}_{k+1}/\tilde{h}_{1,k+1} \quad \text{and} \quad \bar{R}_k t = z_k - \alpha \hat{h}_k.$$

Denote by t_k the solution of (3.4), and by y_k the corresponding one in $R_k y_k = t_k$. Let W_k and $\bar{W}_k$ be computed by forward substitution from $R_k^T W_k^T = V_k^T$ and $\bar{R}_k^T \bar{W}_k^T = W_k^T$. The k-th solution can be written as

$$\begin{aligned} x_k &= x_0 + V_k y_k + \alpha x_d \\ &= x_0 + W_k R_k y_k + \alpha x_d \\ &= x_0 + W_k t_k + \alpha x_d \\ &= x_0 + \bar{W}_k \bar{R}_k t_k + \alpha x_d \\ &= x_0 + \bar{W}_k z_k - \alpha(\bar{W}_k \hat{h}_k - x_d). \end{aligned}$$

So we can still update solution x_k iteratively as in LSMR. Based on what we have shown, we give the pseudocode of the extended LSMR in Algorithm 3.2. The blue parts in Algorithm 3.2 show the difference from LSMR.

Algorithm 3.2 Pseudo-code of Extended LSMR

Given initial guess x_0 and set $r_0 = b - Ax_0, \beta_1 u_1 = r_0, \alpha_1 v_1 = A^T u_1, \bar{\alpha}_1 = \alpha_1, \bar{\zeta}_1 = \alpha_1\beta_1, \rho_0 = 1, \bar{\rho}_0 = 1, \bar{c}_0 = 1, \bar{s}_0 = 0, h_1 = v_1, \bar{h}_0 = 0$

$g_0 = A^T(Ax_d), t_0 = -x_d, h_{11} = v_1^T g_0, g = g_0 - h_{11}v_1$

for $k = 1, 2, \ldots$ **do**

  $\beta_{k+1}u_{k+1} = Av_k - \alpha_k u_k$

  $\alpha_{k+1}v_{k+1} = A^T u_{k+1} - \beta_k v_k$

  $\hat{h}_{1,k+1} = v_{k+1}^T g_0, g = g - v_{k+1}\hat{h}_{1,k+1}$

  $\rho_k = (\bar{\alpha}_k^2 + \beta_{k+1}^2)^{1/2}$

  $c_k = \bar{\alpha}_k/\rho_k, s_k = \beta_{k+1}/\rho_k$

  $\theta_{k+1} = s_k\alpha_{k+1}, \bar{\alpha}_{k+1} = c_k\alpha_{k+1}$

  $\bar{\theta}_k = \bar{s}_{k-1}\bar{\rho}_k, \bar{\rho}_k = ((\bar{c}_{k-1}\rho_k)^2 + \theta_{k+1}^2)^{1/2}$

  $\bar{c}_k = \bar{c}_{k-1}\rho_k/\bar{\rho}_k, \bar{s}_k = \theta_{k+1}/\bar{\rho}_k$

  $\zeta_k = \bar{c}_k\bar{\zeta}_k, \bar{\zeta}_{k+1} = -\bar{s}_k\bar{\zeta}_k$

  $t_p = \hat{h}_{1k}, \hat{h}_{1k} = \bar{c}_k t_p + \bar{s}_k\hat{h}_{1,k+1}, \hat{h}_{1,k+1} = -\bar{s}_k t_p + \bar{c}_k\hat{h}_{1,k+1}$

  $\bar{h}_k = h_k - (\bar{\theta}_k\rho_k/(\rho_{k-1}\bar{\rho}_{k-1}))\bar{h}_{k-1}$

  $x_k = x_{k-1} + (\zeta_k/(\rho_k\bar{\rho}_k))\bar{h}_k$

  $h_{k+1} = v_{k+1} - (\theta_{k+1}/\rho_k)h_k$

  $t_k = t_{k-1} + (\hat{h}_{1k}/(\rho_k\bar{\rho}_k))\bar{h}_k$

  $\delta = \|g\|_2, t_p = \hat{h}_{1,k+1}, \hat{h}_{1,k+1} = (\delta^2 + \hat{h}_{1,k+1}^2)^{1/2}$

  $c = t_p/\hat{\xi}_{k+1}, \hat{\xi}_{k+1} = c\bar{\xi}_{k+1}, \alpha = \hat{\xi}_{k+1}/\hat{h}_{1,k+1}$

  $x_k = x_k - \alpha t_k$

end for

3.4 Numerical Experiments

In this section, we will discuss numerical experiments in three aspects: reorthogonalization, backward error and comparsion of HBMR and LSMR.

3.4.1 Reorthogonalization

First, let's introduce one error assessment to measure the performance of a method for LS, named normalized residual (NRes) defined as below:

$$\text{NRes} = \frac{\|A^T(Ax-b)\|_2}{\|A\|_1(\|A\|_1\|x\|_2+\|b\|_2)},$$

where using $\|A\|_1$ is for its easiness in computation. NRes will be shown against the number of iterations for different cases in our test.

Theoretically, both V_k and U_k are orthonormal matrices in the Golub-Kahan process. However, the orthogonality of both matrices may not be preserved because of rounding errors. First, we want to test the influence of orthogonality of U_k on convergence of LSMR. We compare the convergence of LSMR in three cases: no reorthogonalization of U_k or V_k, only full reorthogonalizing V_k, and full reorthogonalizing both U_k and V_k. Table 3.1 lists all test coefficient matrices of least squares problems drawing from Matrix Market [1]. In our test, we set the right-hand vector b to be $1/2 * (1, 1, \cdots, 1)^T \in \mathbb{R}^m$ for matrices 1-3. For matrices 4-6, we use their original right-hand sides b. Figure 3.1 shows the results of NRes against iterations for all examples in Table 3.1 with the stopping criteria $||A^T r_k||_2 \leq \texttt{tol} \times ||A||_1 \times ||r_k||_2$. When NRes is less than 10e-12, Table 3.2 lists the number of iterations of three cases: no reorthogonalization (no reorth), only reorthogonalizing V_k (full reorth V_k) and reorthogonalizing both V_k and U_k (full reorth V_k and U_k). From both tables, we can tell two observations:

- The orthogonality of V_k and U_k do effect the speed of convergence, i.e., the convergence of LSMR can be accelerated by reorthogonalization of V_k to some extent.
- The convergence of LSMR by reorthogonalizing both U_k and V_k is the same as just reorthogonalizing V_k. Therefore, we just need to reorthogonalize V_k.

Table 3.1: Testing Matrices for Reorthogonalization

ID	matrix	m	n	nnz	cond(A)
1	abb313	313	176	1557	5.0762e+17
2	ash292	292	2920	22082	1.3610e+18
3	ash85	85	85	523	463.7463
4	e50r0000	236	236	5847	1.5699e+073
5	e20r0100	4241	4241	13143168	9.5925e+10
6	illc1850	1850	712	8636	5.7897e+06

Table 3.2: Number of Iterations for LSMR

matrix	no reorth	full reorth V_k	full reorth V_k & U_k
abb313	137	93	93
ash292	1108	290	290
ash85	192	86	86
e50r0000	187	122	122
e20r0100	53	35	35
illc1850	80	54	54

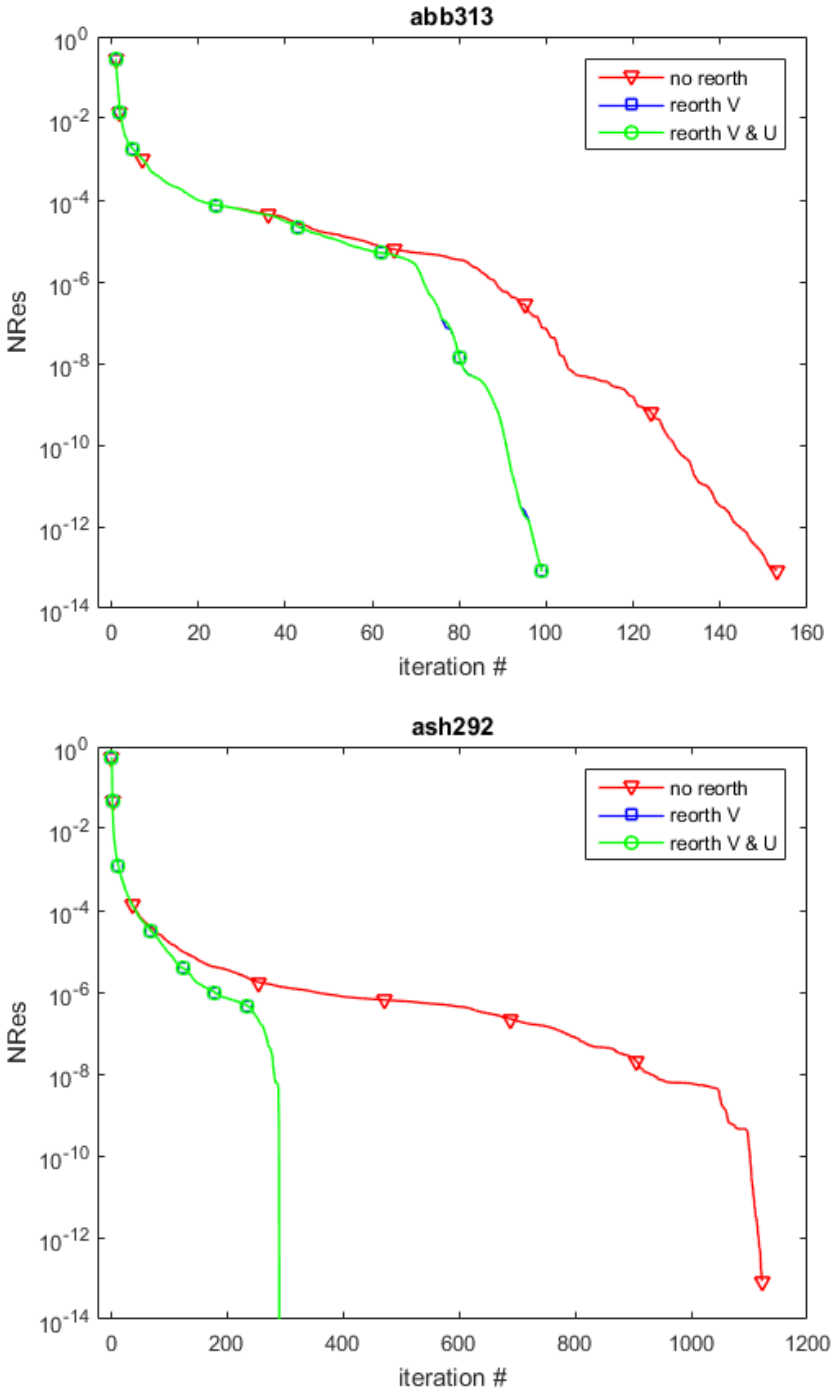

Figure 3.1: NRes *vs.* iteration for `abb313` and `ash292`

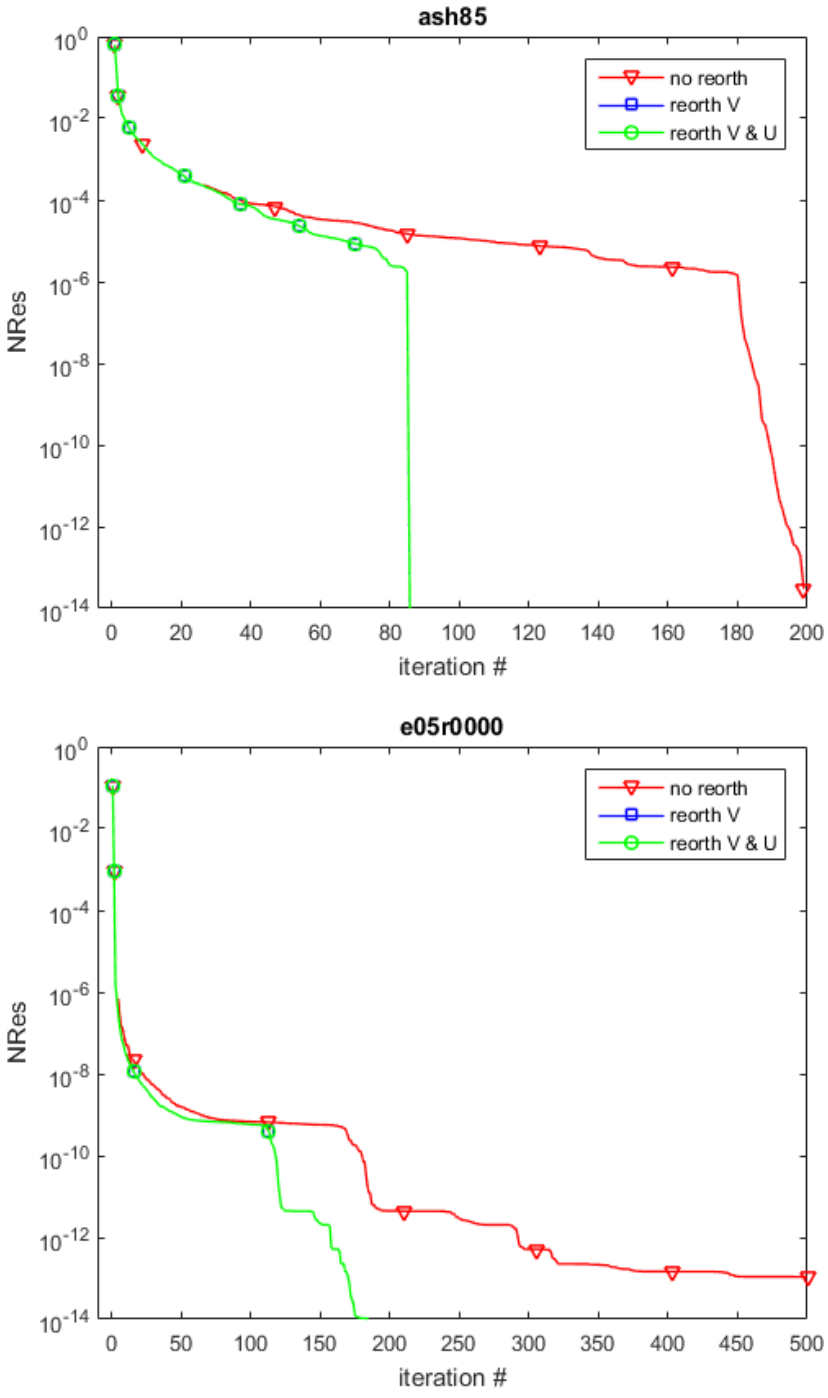

Figure 3.2: NRes *vs.* iteration for `ash85` and `e05r0000`

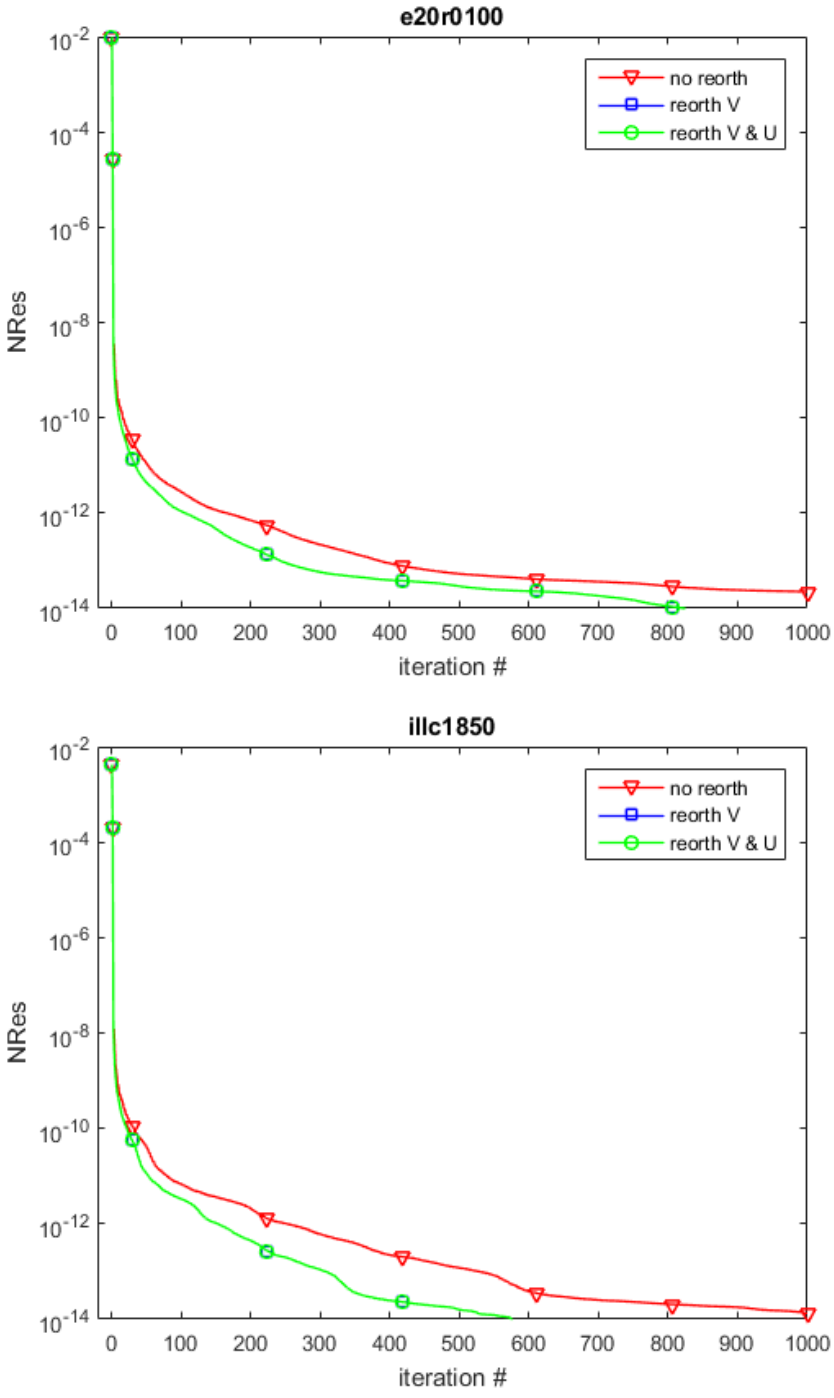

Figure 3.3: NRes *vs.* iteration for `e20r0100` and `illc1850`

3.4.2 Backward Error

The normwise backward error measures the perturbation to A that would make an approximation solution x an exact least squares solution

$$\mu(x) = \min_{\delta A} \|\delta A\|_F \text{ s.t. } x = \arg\min_z \|b - (A + \delta A)z\|_2.$$

$\mu(x)$ can be explicitly expressed by [35, p, 393] [40]:

$$\mu(x) = \min\left\{\frac{\|r\|_2}{\|x\|_2}, \sigma_{\min}[A\ B]\right\},\ B = \frac{\|r\|_2}{\|x\|_2}\left(I - \frac{rr^T}{\|r\|_2^2}\right).$$

Since it's expensive to evaluate $\mu(x)$, we consider the approximate optimal backward error [62, 40] $\tilde{\mu}$:

$$K = \begin{bmatrix} A \\ \frac{||r||_2}{||x||_2} \times I \end{bmatrix}, \quad v = \begin{bmatrix} r \\ 0 \end{bmatrix}, \quad \min_y ||Ky - v||_2, \quad \tilde{\mu}(x) = \frac{||Ky||_2}{||x||_2}.$$

Backward error is an important assessment to measure the accuracy of a method for least squares problems [62, 40, 59]. Often we regard that smaller the backward error is, more accurate the approximate solution is. In addition, the backward error can estimate the run-time stability of one algorithm. In our test, we show the relative approximate optimal backward error E1= $\tilde{\mu}/\|A\|_1$.

3.4.3 Comparison of HBMR and LSMR

The data in Table 3.3 comes from the website of SuiteSparse Matrix Collection [2]. Each comes with their right-hand sides b. We compared LSMR, restarted LSMR and HBMR. The stopping criteria is: if $||A^T r_k||_2 \leq$ `tol` $\times\ ||A||_1 \times ||r_k||_2$, where `tol`=10e-12.

Table 3.3: Testing Matrices

matrix	m	n	nnz	cond(A)
lp_80bau3b	12061	2262	23264	567.2253
lp_brandy	303	220	2202	Inf
lp_pilot_ja	2276	940	14977	2.5307e+08
lp_e226	472	223	2768	9.1322e+03

In order to fairly compare algorithms and make it easy to understand, we use the following testing scenarios:

- We run the restarted LSMR $(k+1)$ (RELSMR) and HBMR (k) in order to best illustrate convergence against the iteration indices. For LSMR $(k/2)$, we reorthogonalize $v_{(k/2+1)}$ with previous $k/2$ v_i's in each iteration when $i > k/2$. There are two considerations for this choice. One is to make sure that for RELSMR and HBMR, all approximate solutions at one cycle are computed from a subspace of dimension $k+1$, i.e., $\text{span}(x_0) + \text{span}(V_{k+1})$ for REFGMRES$(k+1)$, and $\text{span}(x_0)+\text{span}(Z_k)+\text{span}(x_d)$ for HBLSMR(k). Secondly, we want to align the costs per iteration for LSMR$(k/2)$, RELSMR$(k+1)$ and HBMR(k). The x-axis shows the iteration index of LSMR. Since RELSMR and HBMR involves inner iterations, in order to fairly compare all three algorithm, we plot the number of iterations index of LSMR, and total numbers of inner iterations involved in all the cycles of RELSMR and HBMR, respectively.
- The stopping criteria is either NRes is less than or equal to 10e-12 or the number of iterations exceeds 3000.

From Figures 3.4–3.7, we can tell some general observations:

- The relative residuals of all examples shows that LSMR with partial reorthogonalization of V_k converges faster than the one with no reorthogonalization. This observation confirms that the loss of orthogonality of V_k can slow down

the convergence of LSMR and partial reorthogonalization is necessary to speed up the convergence.

- The relative residual is monotonically decreasing for all methods. Figures 3.4–3.7 show that with the comparable computational cost, HBMR converges faster to a smaller relative residual compared to restarted LSMR and LSMR.
- The approximate optimal backward error displays a similar pattern as the normalized residual. HBMR is a stable method according to the results of backward error. Also, we can see that HBMR has the smallest approximate optimal backward error. This means HBMR gives the most accurate solutions among all methods.

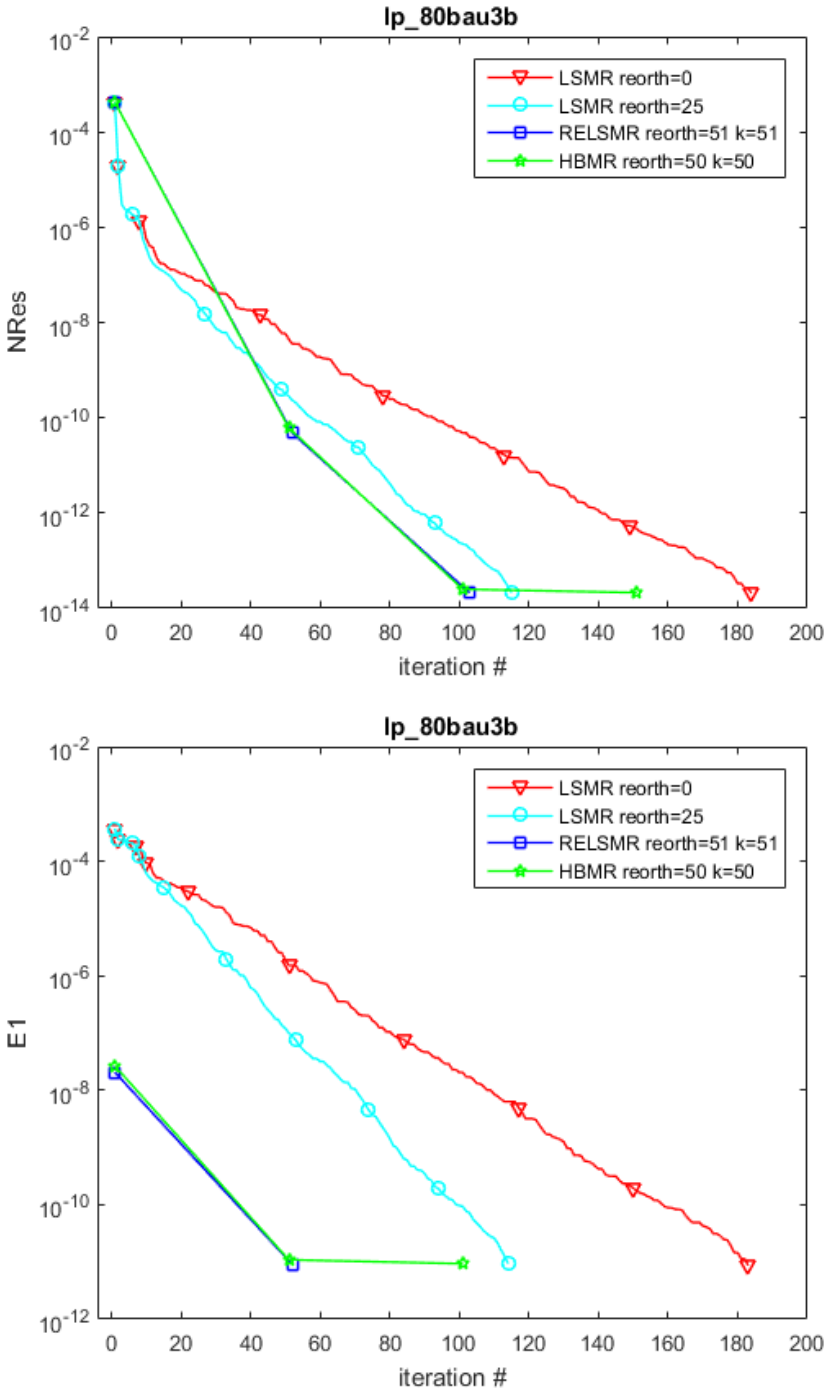

Figure 3.4: Results of `lp_80bau3b`. *Top* : NRes *vs.* iteration; *Bottom* : E1 *vs.* iteration

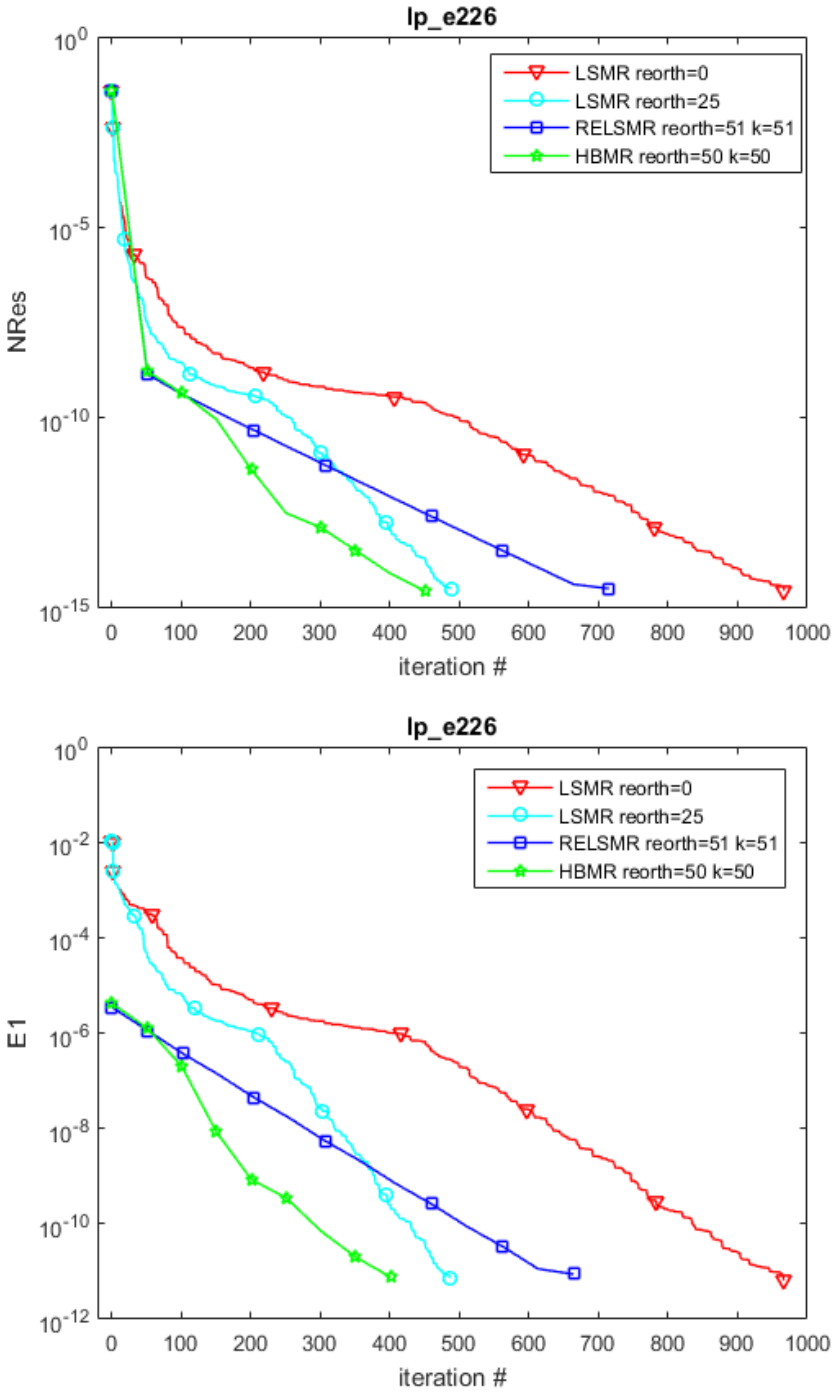

Figure 3.5: Results of `lp_e226`. *Top* : NRes *vs*. iteration; *Bottom* : E1 *vs*. iteration

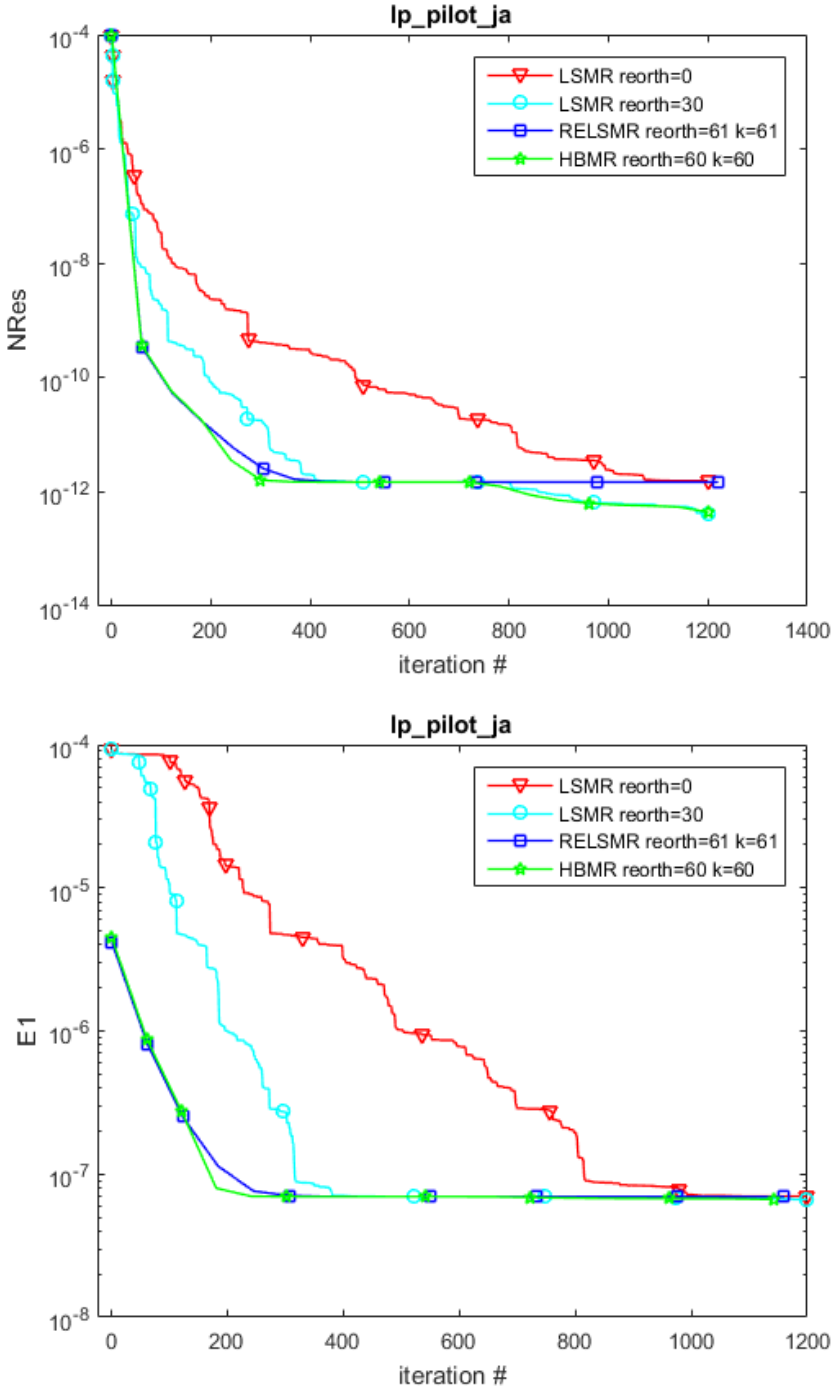

Figure 3.6: Results of `lp_pilot_ja`. *Top* : NRes *vs.* iteration; *Bottom* : E1 *vs.* iteration

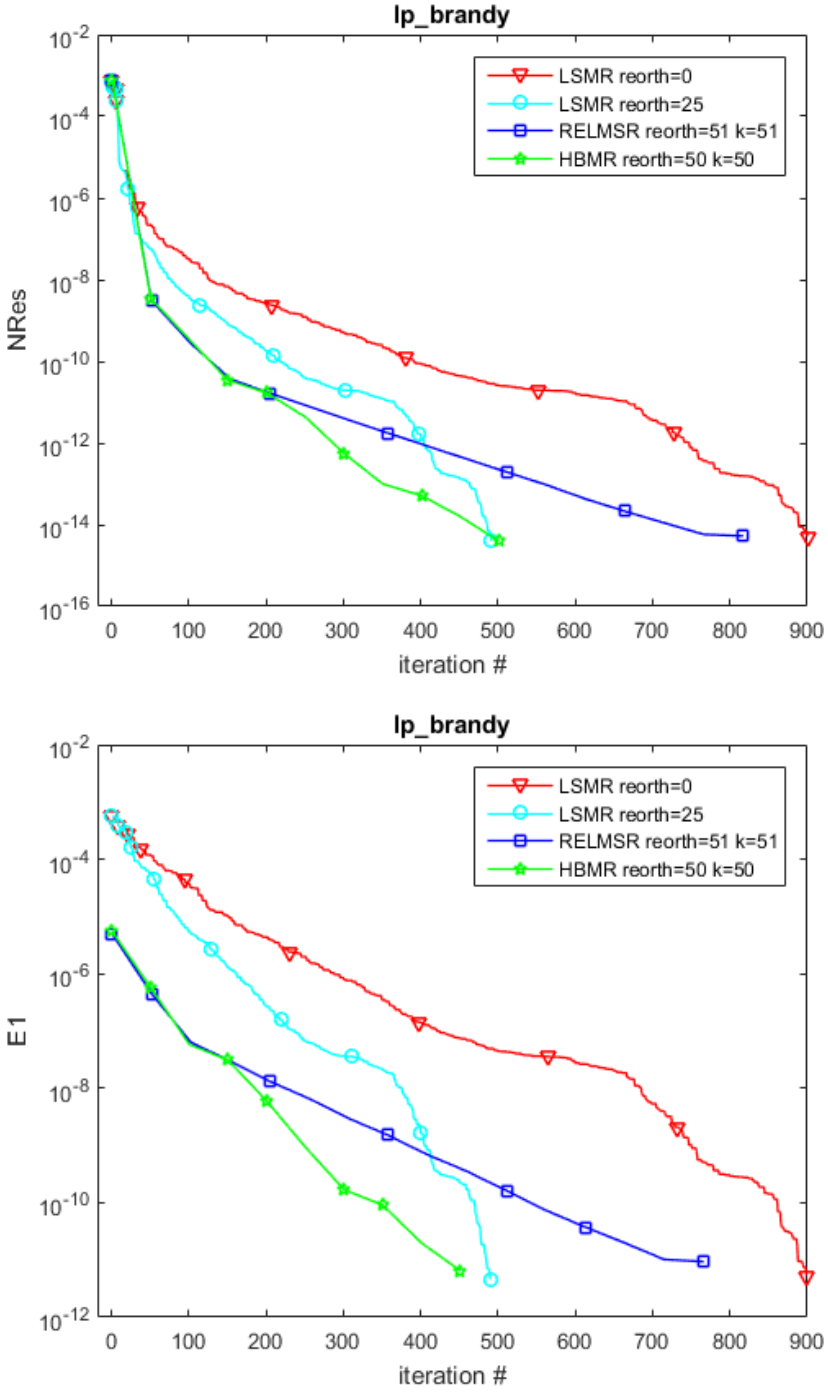

Figure 3.7: Results of `lp_brandy`. *Top* : NRes *vs*. iteration; *Bottom* : E1 *vs*. iteration

CHAPTER 4

FLEXIBLE PRECONDITIONED ITERATIVE METHOD FOR LEAST SQUARES PROBLEMS

In this chapter, we propose a flexible preconditioned iterative method for least squares problems. In this method, an outer-inner iteration is built to constuct changable preconditioners. In section 4.1, we explain our motivation and main idea of our method. In section 4.2, we introduce the right preconditioned least squares problem, factorization-free LSMR (MLSMR) with details, and perform some theoretical analysis. In section 4.3, we integrate our flexible preconditioners with MLSMR to give flexible preconditioned LSMR (FMLSMR). In section 4.4, we show some numerical examples to illustrate the efficiency of our method.

4.1 Motivation and Main Idea

Krylov subspace techniques have been viewed as general-purpose iterative methods, especially since the popularization of preconditioning techniques in 1970s. Based on the Golub-Kahan bidiagonalization process [25], both LSQR [15] and LSMR [23] are iterative methods that seek approximate solutions x_k in the Krylov subspace $\mathcal{K}_k\left(A^TA, A^Tb\right)$. Because of lack of effective preconditioners, LSQR and LSMR can become stagnated as the iterations progress for certain problems. Therefore, applying good preconditioners in LSQR and LSMR is necessary to speed up the convergence. In this chapter, we mainly focus on optimizing LSMR, and the similar techniques can be also applied to LSQR.

There are many ways to construct specific preconditioners for specific problems, such as incomplete LU [54], incomplete QR [45], and preconditioners based on perturbed QR factorizations [5]. But it's not straightforward to determine whether or not a given preconditioner is suitable for the problem at hand. If not, one may attempt other possible preconditioners and switch periodically among them.

It is desirable to be able to switch within the outer iteration instead of restarting. This is the main idea of the flexible GMRES [53] proposed by Saad in 1993. In the flexible GMRES, one linear system needs to be approximately solved in the inner iteration. Different solvers for this inner iteration can result in different preconditioners in each outer iteration. In 2015, Morikuni and Hayami presented an inner-iteration GMRES method [47] for least squares problems. This method involves solving normal-type equations in the form of $A^TAv = p$ by some stationary iterative methods. In this chapter, we combine the above two ideas to form a flexible preconditioner iterative method by using non-stationary methods to solve normal-type equations in the inner iteration that yield different preconditioners in the Golub-Kahan bidiagonalization process.

4.2 Preconditioned Least Squares Problem

The right preconditioned least squares problem is as follows

$$\min_{\hat{x}} \left\| AL^{-1}\hat{x} - b \right\|_2, x = L^{-1}\hat{x}, \tag{4.1}$$

which is equivalent to the split preconditioned normal equation

$$L^{-T}A^TAL^{-1}\hat{x} = L^{-T}A^Tb. \tag{4.2}$$

Recall that the Golub-Kahan process with $t = b$ iteratively transforms $[b \quad A]$ to the upper-bidiagonal form $[\beta_1 e_1 \quad B_k]$, where

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

After the k-th step, we have

$$AV_k = U_{k+1}B_k \quad \text{and} \quad A^T U_{k+1} = V_{k+1}L_{k+1}^T,$$

where $V_{k+1} = [v_1, v_2, \cdots, v_k], U_{k+1} = [u_1, u_2, \cdots, u_k]$, and $L_{k+1} = [B_k \quad \alpha_{k+1}e_{k+1}]$. Both V_{k+1} and U_{k+1} are orthonormal. Applying Algorithm 1.8 to the preconditioned least squares problem (4.1), we have

$$q = AL^{-1}v_k - \alpha_k u_k, \qquad \beta_{k+1} = \|q\|_2, \quad u_{k+1} = q/\beta_{k+1}, \tag{4.3a}$$

$$p = L^{-T}A^T u_{k+1} - \beta_{k+1}v_k, \qquad \alpha_{k+1} = \|p\|_2, \quad v_{k+1} = p/\alpha_{k+1}. \tag{4.3b}$$

Define $M = L^T L$ and $\tilde{v}_{k+1} = L^{-1}v_{k+1}$. Pre-multiplying both sides of (4.3b) by L^{-1}, we have $\alpha_{k+1}\tilde{v}_{k+1} = M^{-1}A^T u_{k+1} - \beta_{k+1}\tilde{v}_k$. We know $\alpha_{k+1} = \langle p, p\rangle^{\frac{1}{2}}$. To compute $\langle p, p\rangle$, we notice

$$\begin{aligned} \langle p, p\rangle &= \langle L^{-T}A^T u_{k+1} - \beta_{k+1}v_k, L^{-T}A^T u_{k+1} - \beta_{k+1}v_k\rangle \\ &= \langle L^{-T}A^T u_{k+1} - \beta_{k+1}v_k, L^{-T}A^T u_{k+1}\rangle - \langle L^{-T}A^T u_{k+1} - \beta_{k+1}v_k, \beta_{k+1}v_k, \rangle \\ &= \langle L^{-T}A^T u_{k+1}, L^{-T}A^T u_{k+1}\rangle - 2\langle \beta_{k+1}v_k, L^{-T}A^T u_{k+1}\rangle + \langle \beta_{k+1}v_k, \beta_{k+1}v_k\rangle. \end{aligned}$$

By the definition of the M-inner product, we have

$$\begin{aligned}
\langle L^{-T}A^T u_{k+1}, L^{-T}A^T u_{k+1}\rangle &= u_{k+1}^T A L^{-1}L^{-T}A^T u_{k+1} \\
&= u_{k+1}^T A M^{-1} A^T u_{k+1} \\
&= \langle A^T u_{k+1}, A^T u_{k+1}\rangle_{M^{-1}} \\
&= \langle M^{-1}A^T u_{k+1}, M^{-1}A^T u_{k+1}\rangle_M, \\
\langle \beta_{k+1} v_k, L^{-T}A^T u_{k+1}\rangle &= \beta_{k+1} u_{k+1}^T A L^{-1} v_k \\
&= \beta_{k+1} u_{k+1}^T A\tilde{v}_k = \beta_{k+1} u_{k+1}^T A M^{-T} M\tilde{v}_k \\
&= \langle M^{-T}A^T u_{k+1}, \beta_{k+1}\tilde{v}_k\rangle_M, \\
\langle \beta_{k+1} v_k, \beta_{k+1} v_k\rangle &= \langle \beta_{k+1} L\tilde{v}_k, \beta_{k+1} L\tilde{v}_k\rangle \\
&= \beta_{k+1}\tilde{v}_k^T L^T L \beta_{k+1}\tilde{v}_k = \beta_{k+1}\tilde{v}_k^T M \beta_{k+1}\tilde{v}_k \\
&= \langle \beta_{k+1}\tilde{v}_k, \beta_{k+1}\tilde{v}_k\rangle_M.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\langle p, p\rangle &= \langle M^{-1}A^T u_{k+1}, M^{-1}A^T u_{k+1}\rangle_M - 2\langle M^{-T}A^T u_{k+1}, \beta_{k+1}\tilde{v}_k\rangle_M + \langle \beta_{k+1}\tilde{v}_k, \beta_{k+1}\tilde{v}_k\rangle_M \\
&= \langle M^{-1}A^T u_{k+1} - \beta_{k+1}\tilde{v}_k, M^{-1}A^T u_{k+1} - \beta_{k+1}\tilde{v}_k\rangle_M.
\end{aligned}$$

Denote by $s = M^{-1}A^T u_{k+1} - \beta_{k+1}\tilde{v}_k$. So (4.3) can be written as

$$\begin{aligned}
\beta_{k+1} u_{k+1} &= A\tilde{v}_k - \alpha_k u_k, \\
s &= M^{-1}A^T u_{k+1} - \beta_{k+1}\tilde{v}_k, \\
\alpha_{k+1} &= \langle s, s\rangle_M^{\frac{1}{2}}, \\
\tilde{v}_{k+1} &= s/\alpha_{k+1}.
\end{aligned}$$

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

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

s can be written as $s = M^{-1}\tilde{s}$ and $\alpha_{k+1} = \langle s, s\rangle_M^{\frac{1}{2}} = sMs = sMM^{-1}\tilde{s} = \langle s, \tilde{s}\rangle^{\frac{1}{2}}$.

Then the k-th step of the Golub-Kahan process is

$$\begin{aligned} \beta_{k+1}u_{k+1} &= A\tilde{v}_k - \alpha_k u_k, \\ \tilde{s} &= A^T u_{k+1} - \beta_{k+1}\tilde{p}, \\ s &= M^{-1}\tilde{s}, \\ \alpha_{k+1} &= \langle s, \tilde{s}\rangle^{\frac{1}{2}}, \\ \tilde{v}_{k+1} &= s/\alpha_{k+1}. \end{aligned} \tag{4.4}$$

In the $(k+1)$-th step, we define $\tilde{p} = M\tilde{v}_{k+1}$ similarly as in the k-th step. Since $\tilde{v}_{k+1} = s/\alpha_{k+1}$ and $s = M^{-1}\tilde{s}$, we have

$$\begin{aligned} \tilde{p} = M\tilde{v}_{k+1} &= Ms/\alpha_{k+1} \\ &= MM^{-1}\tilde{s}/\alpha_{k+1} \\ &= \tilde{s}/\alpha_{k+1} \end{aligned}$$

Use $\tilde{p}$ and $\tilde{v}_{k+1}$ to replace $\tilde{s}$ and s, respectively. Therefore, an computational cost saving version of preconditioned Golub-Kahan bidiagonalization process can be summarized as

$$\begin{aligned} \beta_{k+1}u_{k+1} &= A\tilde{v}_k - \alpha_k u_k, \\ \tilde{p} &= A^T u_{k+1} - \beta_{k+1}\tilde{p}, \\ \tilde{v}_{k+1} &= M^{-1}\tilde{p}, \\ \alpha_{k+1} &= \langle \tilde{v}_{k+1}, \tilde{p}\rangle^{\frac{1}{2}}, \\ \tilde{v}_{k+1} &= \tilde{v}_{k+1}/\alpha_{k+1}, \\ \tilde{p} &= \tilde{p}/\alpha_{k+1}. \end{aligned} \tag{4.5}$$

As we can see in (4.3), for a given preconditioner L, we need to solve two linear systems of the form $Lv = u$ and $L^T v = u$. However, in (4.5), we only need to deal

with one inner linear system $Ms = \tilde{s}$ and don't need to factorize the preconditioner M. This is the so-called factorization-free preconditioner [4].

After the k-step preconditioned Golub-Kahan process, instead of having

$$AL^{-1}V_k = U_{k+1}B_k, \tag{4.6}$$

$$L^{-T}A^TU_{k+1} = V_kB_k^T + \alpha_{k+1}v_{k+1}e_{k+1}^T, \tag{4.7}$$

we have

$$A\tilde{V}_k = U_{k+1}B_k, \tag{4.8}$$

$$M^{-1}A^TU_{k+1} = \tilde{V}_kB_k^T + \alpha_{k+1}\tilde{v}_{k+1}e_{k+1}^T. \tag{4.9}$$

It's easy to see that U_{k+1} is orthonormal, while $\tilde{V}_{k+1}$ isn't. For the preconditioned least squares problem (4.1), LSMR seeks the approximation solutions $\hat{x}_k$ in the subspace span(V_k), i.e. $\hat{x}_k = V_k\hat{y}_k$, where $\hat{y}_k$ is the vector we need to find. Because of (4.8), the residual of the preconditioned normal equation (4.2) can be written as

$$\begin{aligned}
L^{-T}A^Tr_k &= L^{-T}A^T(b - AL^{-1}\hat{x}_k) = L^{-T}A^T(b - AL^{-1}V_k\hat{y}_k) \\
&= L^{-T}A^T(b - U_{k+1}B_k\hat{y}_k) = L^{-T}A^Tb - L^{-T}A^TU_{k+1}B_k\hat{y}_k \\
&= \bar{\beta}_1v_1 - V_{k+1}\begin{bmatrix} B_k^TB_k \\ \bar{\beta}_{k+1}e_k^T \end{bmatrix}\hat{y}_k \\
&= V_{k+1}\left(\bar{\beta}_1e_1 - \begin{bmatrix} B_k^TB_k \\ \bar{\beta}_{k+1}e_k^T \end{bmatrix}\hat{y}_k\right),
\end{aligned}$$

where $\bar{\beta}_1 = \alpha_1\beta_1$ and $\bar{\beta}_k = \alpha_k\beta_k$. So

$$\min_{\hat{x}} \left\|L^{-T}A^Tr\right\|_2 = \min_{\hat{y}} \left\|\bar{\beta}_1e_1 - \begin{bmatrix} B_k^TB_k \\ \bar{\beta}_{k+1}e_k^T \end{bmatrix}\hat{y}\right\|_2.$$

The last equality holds because of V_{k+1} is orthonormal. The doubl QR factorization in LSMR can still be used to solve the following subproblem

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

Therefore solution of the original least squares problem can be approximated by

$$x_k = L^{-1}\hat{x}_k = L^{-1}V_k\hat{y}_k = \tilde{V}_k\hat{y}_k$$

which means we can avoid solving $x_k = L^{-1}\tilde{x}_k$ to get the solution of the preconditioned least squares problem. This technique and the preconditioned Golub-Kahan process (4.5) give rise to the factorization-free LSMR (MLSMR) method in Algorithm 4.1

Algorithm 4.1 factorization-free preconditioned LSMR (MLSMR)

Initialization:

$\beta_1 u_1 = b, \tilde{p} = A^T u_1, \tilde{v}_1 = M^{-1}\tilde{p}, \alpha_1 = \langle \tilde{v}_1, \tilde{p} \rangle^{1/2}, \tilde{p} = \tilde{p}/\alpha_1, \tilde{v}_1 = \tilde{v}_1/\alpha_1$

$\bar{\alpha}_1 = \alpha_1$, $\bar{\xi}_1 = \alpha_1\beta_1$, $\rho_0 = \bar{\rho}_0 = \bar{c}_0 = 1$, $\bar{s}_0 = 0, h_1 = \tilde{v}_1$, $\bar{h}_0 = 0$, $x_0 = 0$

for $k = 1, 2, \ldots,$ **do**

  $\beta_{k+1} u_{k+1} = A\tilde{v}_k - \alpha_k u_k$

  $\tilde{p} = A^T u_{k+1} - \beta_{k+1}\tilde{p}$

  $\tilde{v}_{k+1} = M^{-1}\tilde{p}$, $\alpha_{k+1} = \langle \tilde{v}_{k+1}, \tilde{p} \rangle^{1/2}$

  $\tilde{p} = \tilde{p}/\alpha_{k+1}$

  $\tilde{v}_{k+1} = \tilde{v}_{k+1}/\alpha_{k+1}$

  $\rho_k = (\bar{\alpha}_k^2 + \beta_{k+1}^2)^{1/2}$

  $c_k = \bar{\alpha}_k/\rho_k$

  $s_k = \beta_{k+1}/\rho_k$, $\theta_{k+1} = s_k\alpha_k$

  $\bar{\alpha}_{k+1} = c_k\alpha_{k+1}$

  $\bar{\theta}_k = \bar{s}_{k-1}\rho_k$

  $\bar{\rho}_k = ((\bar{c}_{k+1}\rho_k)^2 + \theta_{k+1}^2)^{1/2}$

  $\bar{c}_k = \bar{c}_{k-1}\rho_k/\bar{\rho}_k$

  $\bar{s}_k = \theta_{k+1}/\bar{\rho}_k$

  $\xi_k = \bar{c}_k\xi_k$, $\bar{\xi}_k = -\bar{s}_k\bar{\xi}_k$

  $\bar{h}_k = h_k - (\bar{\theta}_k\rho_k/(\rho_{k-1}\bar{\rho}_{k-1}))\bar{h}_{k-1}$

  $x_k = x_{k-1} + (\xi_k/(\rho_k\bar{\rho}_k))\bar{h}_k$

  $h_{k+1} = \tilde{v}_{k+1} - (\theta_{k+1}/\rho_k)h_k$

  if $\|A^T r_k\|_2 \leq \texttt{tol}\|A\|_1(\|b\|_2 + \|A\|_1\|x_k\|_2)$ **then**

    break

  end if

end for

Theorem 4.2.1. *At the k-th step of MLSMR,*

$$x_k \in \mathcal{K}_k(M^{-1}A^TA, M^{-1}A^Tb) = L^{-1}\mathcal{K}_k(L^{-T}A^TAL^{-1}, L^{-T}A^Tb).$$

Proof *Applying LSMR to the split preconditioned normal equation* (4.2)*, we find the approximate solution* x_k *satisfies*

$$x_k \in \text{span}(L^{-1}V_k) = L^{-1}\mathcal{K}_k(L^{-T}A^TAL^{-1}, L^{-T}A^Tb).$$

According to the preconditioned Golub-Kahan bidiagonalization process (4.5)*, we have*

$$\beta_1 u_1 = b, \quad \alpha_1 \tilde{v}_1 = M^{-1}A^T u_1,$$
$$\beta_{k+1}u_{k+1} = A\tilde{v}_k - \alpha_k u_k, \quad \alpha_{k+1}\tilde{v}_{k+1} = M^{-1}A^T u_{k+1}\tilde{v}_k - \beta_{k+1}\tilde{v}_k.$$

This means $x_k = \text{span}(\tilde{V}_k) = \mathcal{K}_k(M^{-1}A^TA, M^{-1}A^Tb)$*. Therefore*

$$x_k \in L^{-1}\mathcal{K}_k(L^{-T}A^TAL^{-1}, L^{-T}A^Tb) = \mathcal{K}_k(M^{-1}A^TA, M^{-1}A^Tb),$$

as was to be shown.

Theorem 4.2.2. $\tilde{V}_{k+1}$ *is M-orthonormal.*

Proof *Based on the Golub-Kahan bidiagonalization process, we know both* U_{k+1} *and* V_{k+1} *are orthonormal, that is* $U_{k+1}^TU_{k+1} = I_{k+1}$ *and* $V_{k+1}^TV_{k+1} = I_{k+1}$*. We know* $\tilde{V}_{k+1} = L^{-1}V_{k+1}$*, which implies*

$$I_{k+1} = V_{k+1}^TV_{k+1} = \tilde{V}_{k+1}^TL^TL\tilde{V}_{k+1} = \tilde{V}_{k+1}^TM\tilde{V}_{k+1} = \langle \tilde{V}_{k+1}, \tilde{V}_{k+1}\rangle_M,$$

i.e., $\tilde{V}_{k+1}$ *is M-orthonormal.*

4.3 Flexible Preconditioned Iterative Method

In MLSMR, if $M = A^TA$ and M^{-1} can be exactly calculated with no rounding error, x_1 is the exact solution of normal equation (1.10). But it's impractical to

exactly compute $(A^TA)^{-1}$, therefore an approximate inverse of A^TA has to be used. This means we can solve

$$A^TA\tilde{v}_k = \tilde{p} \tag{4.10}$$

to get M_k^{-1} as an approximate of $(A^TA)^{-1}$ in inner iterations. If choosing stationary methods such as Jacobi and SOR-type methods to solve (4.10), we can actually obtain an fix preconditioner M as mentioned in section 4.1. However, applying non-stationary methods like CG and GMRES can give us M_k^{-1}, i.e., changeable preconditioners in each iteration. For any given non-stationary inner solver, as $\tilde{v}_k$ and the right-hand side $\tilde{p}$ change in every iteration, M_k^{-1} changes. Algorithm 4.2 outlines the flexible preconditioned LSMR method (FMLSMR). In FMLSMR, the

Algorithm 4.2 Flexible MLSMR

1: Initialization:

2: $\beta_1 u_1 = b, \tilde{p} = A^T u_1, \tilde{v}_1 = M_1^{-1}\tilde{p}, \alpha_1 = \langle \tilde{v}_1, \tilde{p}\rangle^{1/2}, \tilde{p} = \tilde{p}/\alpha_1, \tilde{v}_1 = \tilde{v}_1/\alpha_1$

3: $\bar{\alpha}_1 = \alpha_1$, $\bar{\xi}_1 = \alpha_1\beta_1$, $\rho_0 = \bar{\rho}_0 = \bar{c}_0 = 1$, $\bar{s}_0 = 0, h_1 = \tilde{v}_1$, $\bar{h}_0 = 0$, $x_0 = 0$

4: **for** $k = 1, 2, \ldots,$ **do**

5: $\quad \beta_{k+1}u_{k+1} = A\tilde{v}_k - \alpha_k u_k$

6: $\quad \tilde{p} = A^T u_{k+1} - \beta_{k+1}\tilde{p}$

7: $\quad \tilde{v}_{k+1} = M_{k+1}^{-1}\tilde{p}$, $\alpha_{k+1} = \langle \tilde{v}_{k+1}, \tilde{p}\rangle^{1/2}$

8: $\quad \tilde{p} = \tilde{p}/\alpha_{k+1}$

9: $\quad \tilde{v}_{k+1} = \tilde{v}_{k+1}/\alpha_{k+1}$

10: $\quad$ Steps 10-24 as in Algorithm 4.1

11: **end for**

approximate solution x_k is sought in $\mathrm{span}(\tilde{V}_k)$, not in $\mathcal{K}_k(M^{-1}A^TA, M^{-1}A^Tb)$ any more. After the k-th step, we have

$$
\begin{aligned}
A\tilde{V}_k &= U_{k+1}B_k, \\
A^TU_{k+1} &= [M_1\tilde{v}_1, M_2\tilde{v}_2, \cdots, M_{k+1}\tilde{v}_{k+1}]
\begin{bmatrix}
\alpha_1 & \beta_2 & & & \\
 & \alpha_2 & \beta_3 & & \\
 & & \ddots & \ddots & \\
 & & & \ddots & \beta_{k+1} \\
 & & & & \alpha_{k+1}
\end{bmatrix} \\
&= [M_1\tilde{v}_1, M_2\tilde{v}_2, \cdots, M_{k+1}\tilde{v}_{k+1}]\, L_{k+1}^T.
\end{aligned}
$$

Let $x = \tilde{V}_k\tilde{y}$. $\|A^Tr\|_2$ can be expressed as follows,

$$
\begin{aligned}
\left\|A^Tr\right\|_2 &= \left\|A^Tb - A^TAx\right\|_2 = \left\|A^Tb - A^TU_{k+1}B_k\tilde{y}\right\|_2 \\
&= \left\| A^Tb - [M_1\tilde{v}_1, \cdots, M_{k+1}\tilde{v}_{k+1}] \begin{bmatrix} B_k^TB_k \\ \alpha_{k+1}\beta_{k+1}e_k^T \end{bmatrix} \tilde{y} \right\|_2 \\
&= \left\| [M_1\tilde{v}_1, \cdots, M_{k+1}\tilde{v}_{k+1}]\, (\bar{\beta}_1e_1 - \begin{bmatrix} B_k^TB_k \\ \bar{\beta}_{k+1}e_k^T \end{bmatrix} \tilde{y}) \right\|_2 \\
&\leq \|[M_1\tilde{v}_1, \cdots, M_{k+1}\tilde{v}_{k+1}]\|_2 \left\| \bar{\beta}_1e_1 - \begin{bmatrix} B_k^TB_k \\ \bar{\beta}_{k+1}e_k^T \end{bmatrix} \tilde{y} \right\|_2 .
\end{aligned}
$$

In our flexible preconditioned LSMR (FMLSMR) method, we solve the sub-problem as below

$$
\min_{\tilde{y}} \left\| \bar{\beta}_1e_1 - \begin{bmatrix} B_k^TB_k \\ \bar{\beta}_{k+1}e_k^T \end{bmatrix} \tilde{y} \right\|_2 .
$$

As we can see, the approximate solution $x_k = \tilde{V}_k \tilde{y}_k$ formed by the minimizer $\tilde{y}_k$ of the above subproblem is also a good approximate solution of the original least squares problem (1.2) when $\| [M_1 \tilde{v}_1, \cdots, M_{k+1} \tilde{v}_{k+1}] \|_2$ is not too big.

Similarly, we can have flexible factorization-free LSQR with the following inequality

$$\begin{aligned} \|r\|_2 = \|b - Ax\|_2 = \left\| b - A\tilde{V}_k \tilde{y} \right\|_2 = \|b - U_{k+1} B_k \tilde{y}\|_2 \\ = \|U_{k+1}(\beta_1 e_1 - B_k \tilde{y}_k)\|_2 \leq \|U_{k+1}\|_2 \, \|\beta_1 e_1 - B_k \tilde{y}_k\|_2 . \end{aligned}$$

When $\|U_{k+1}\|_2$ is not too big, $\tilde{v}_{k+1} \tilde{y}_k$ is a good approximate solution of (1.2).

4.4 Numerical Experiments

Table 4.1 lists the number of flops for LSMR and FMSLMR with the l-step GMRES as the inner solver, where (MV) is the number of flops by one matrix-vector multiplication with A or A^T. We take it to be twice the number of nonzero entries in A. For simplicity, we only keep the leading terms in flops by the two major actions in each iteration: matrix-vector multiplications and solutions of the inner linear systems. Table 4.2 shows the information of all test matrices with right-hand vectors which are drawn from SuiteSparse Matrix Collection [2] .

Table 4.1: Flops for LSMR Variants

per iteration of LSMR	2(MV)
per iteration of FMLSMR(GMRES(l))	$(2l+3)$(MV)$+2l^2n + 4l^2$

We present several numerical experiments to compare LSMR and FMLSMR with GMRES as the inner solver. Comparisons will be done in two aspects:

Table 4.2: Testing Matrices

ID	matrix	m	n	nnz	sparsity
1	lp_cre_a	7248	3516	18168	7.1291e-04
2	lp_cre_b	77137	9648	260785	3.5041e-04
3	lp_cre_c	6411	3068	15977	8.1230e-04
4	lp_greenbea	5598	2392	31070	0.0023
5	lp_ken_11	21349	14694	49058	1.5638e-04
6	lp_maros	1966	846	10137	0.0061
7	lp_pilot	4860	1441	44375	0.0063
8	lp_osa_07	25067	1118	144812	0.0052

- Normalized residual (NRes)

$$\text{NRes} = \frac{\|A^T(Ax-b)\|_2}{\|A\|_1(\|A\|_1\|x\|_2 + \|b\|_2)}$$

  against the number of iterations for each methods in Table 4.2, where using $\|A\|_1$ is for its easiness in computation.

- Approximate backward error. We use the approximate backward error [59]

$$\hat{E} = -\frac{rr^TA}{\|r\|_2^2}, \quad \|\hat{E}\|_2 = \frac{\|A^Tr\|_2}{\|r\|_2}.$$

  In our test, we show the relative approximate backward error $\|\hat{E}\|/\|A\|_1$, denoted by nrmAr/(nrmr*nrmA) in figures. As we mentioned in chapter 3, the backward error can estimate the accuracy and stability of a method for least squares problem. Often we regard that smaller the backward error is, more accurate the approximate solution is.

In our numerical experiments, the inner iteration (4.10) is solved by the l-step GMRES. For each matrix, we ran different l-step GMRES. Since $\tilde{V}_k$ is not orthonormal in FMLSMR, we don't have to reorthogonalize $\tilde{V}_k$. Table 4.3 lists The number of iterations needed by both algorithms when NRes $\leq$ 10e-12, except for

those marked by "-" which means that the maximum number 5000 of iterations is exceeded without satisfying the stopping criteria.

Table 4.3: Number of Iterations

ID	matrix	LSMR	FMLSMR
1	lp_cre_a	-	-
2	lp_cre_b	-	666
3	lp_cre_c	-	2149
4	lp_greenbea	2471	451
5	lp_ken_11	537	46
6	lp_maros	3316	217
7	lp_pilot	2581	450
8	lp_osa_07	68	9

We display all the figures to show how NRes and relative approximate backward errors in the computed solutions move against the iteration index. Here are some observations we got from the figures and Table 4.3.

- Figures 4.1–4.4 show that the relative residual decreases for both LSMR and FMLSMR. Together with Table 4.3, we can see FMLSMR converges faster than LSMR for all examples. For lp_cre_a, both LSMR and FLSMRM cannot reach the stopping criteria, but FMLSMR has a smaller NRes than LSMR at every iteration. For lp_cre_b and lp_cre_c, FMLSMR converges to 10e-12 at certain iterations, while LSMR doesn't. For Examples 4-8, the number of iterations needed to get NRes$\leq$ 10e-12 by FMLSMR is much smaller than by LSMR.
- Figures 4.5–4.8 show the relative approximate backward errors of all examples. The results have almost similar patterns as relative residuals. We can see that $\|\hat{E}^{FMLSMR}\| \leq \|\hat{E}^{LSMR}\|$ almost always, which means for the same numbers of iterations, FMLSMR computes more accurate solutions than LSMR.

- We show part of the enlarged view results for `lp_cre_b` in Figure 4.9. For FMLSMR, NRes and $\|\hat{E}^{FMLSMR}\|$ decreases not monotonically but oscillatingly. For the rest of examples, we have the similar discovery.
- In Figure 4.10, we show the results for `lp_pilot` with different l for inner solver GMRES. As we can see, as l gets larger, the iterations needed for FMLSMR is getting smaller. Similar observation can be seen in other examples. This can be explained by the fact that the greater l is, the better $(A^TA)^{-1}$ is approximated, which returns the more accurate solution x_k.

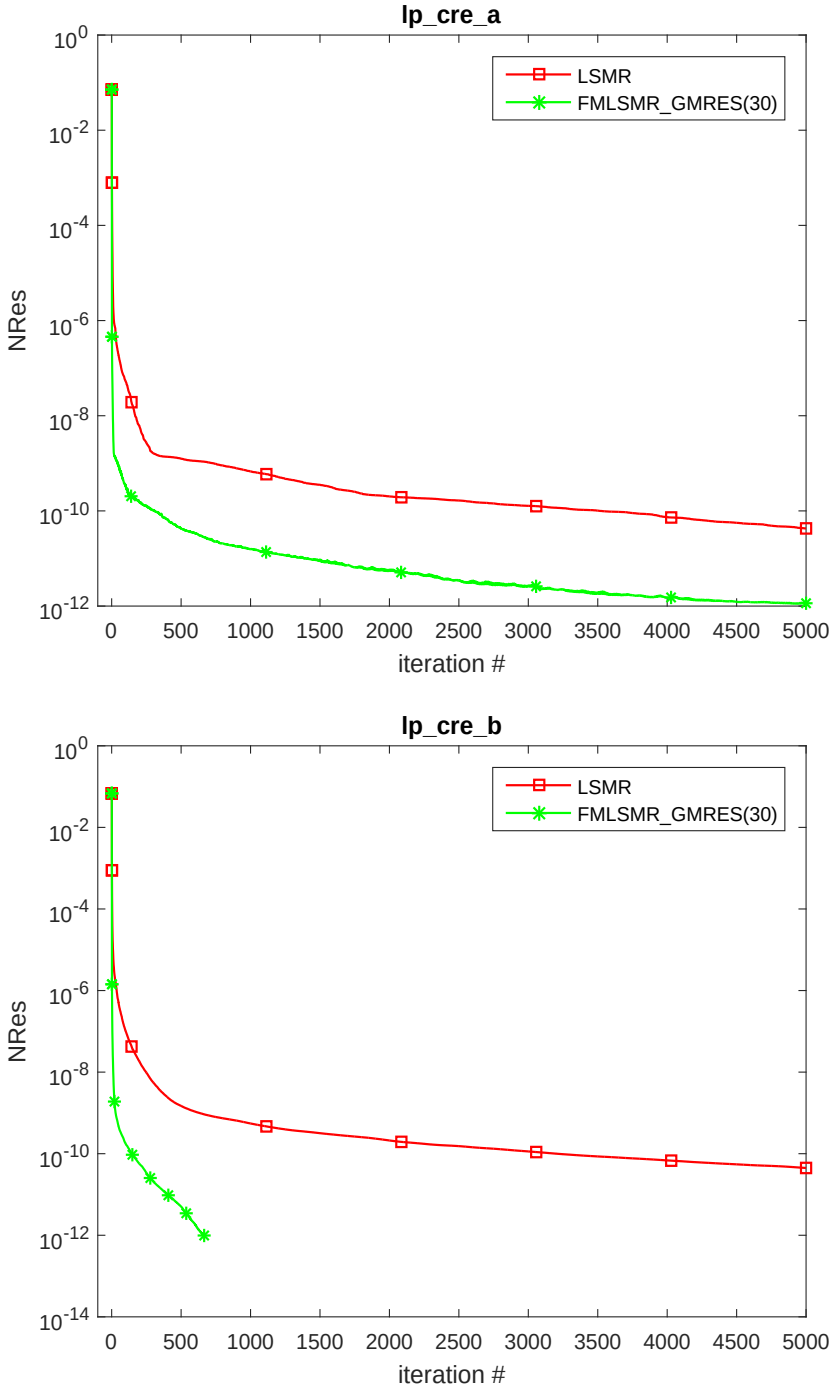

Figure 4.1: NRes *vs.* iteration for `lp_cre_a` and `lp_cre_b`

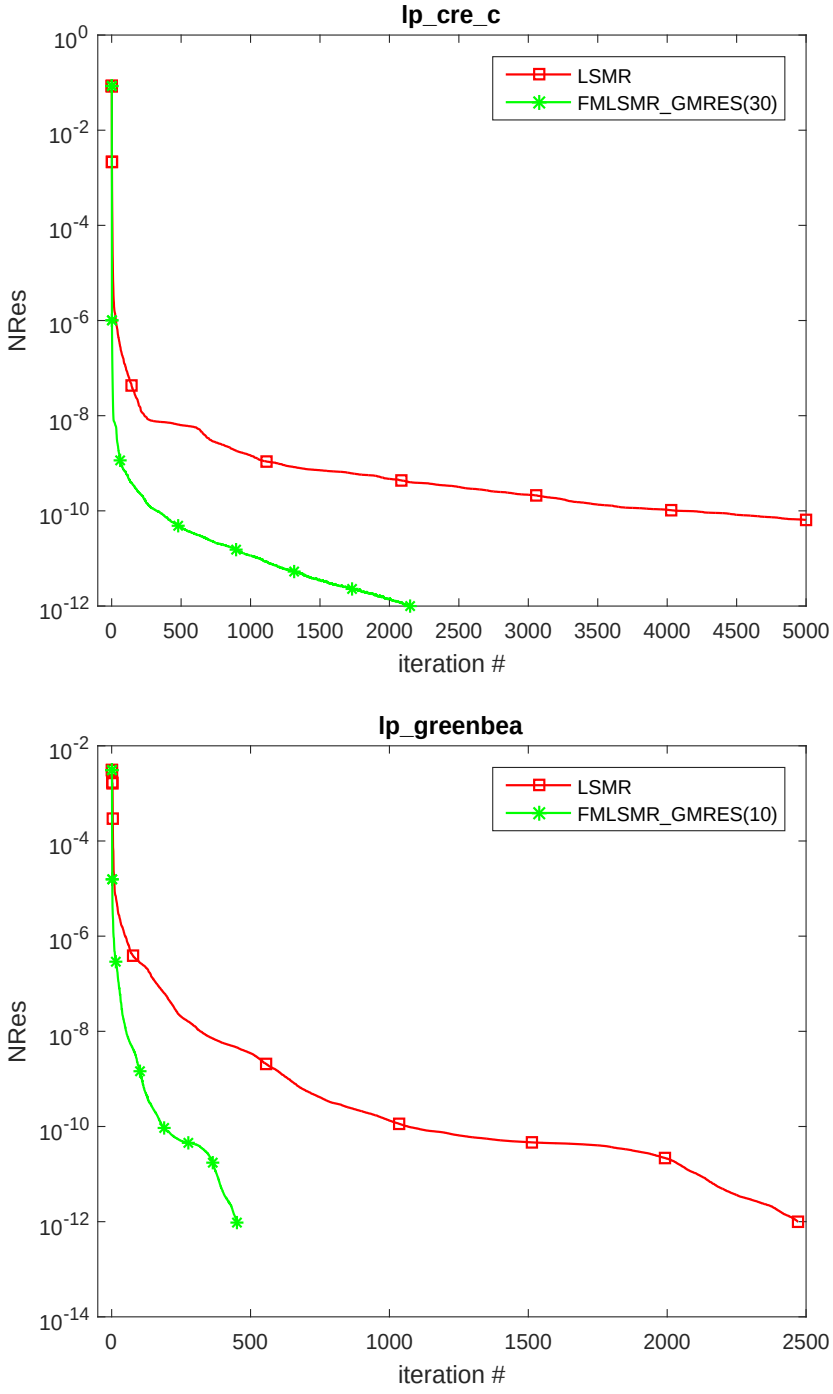

Figure 4.2: NRes *vs*. iteration for `lp_cre_c` and `lp_greenbea`

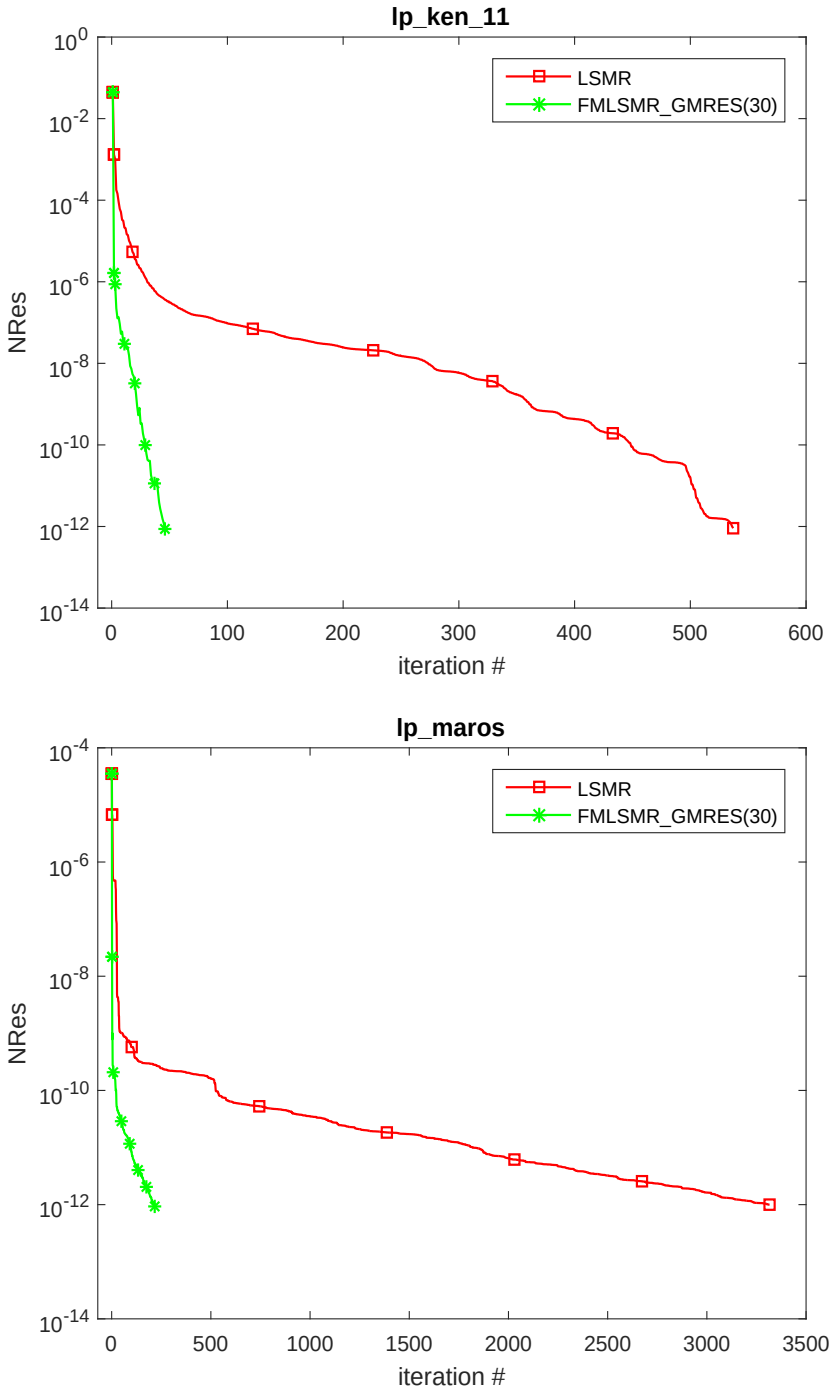

Figure 4.3: NRes *vs.* iteration for `lp_ken_11` and `lp_maros`

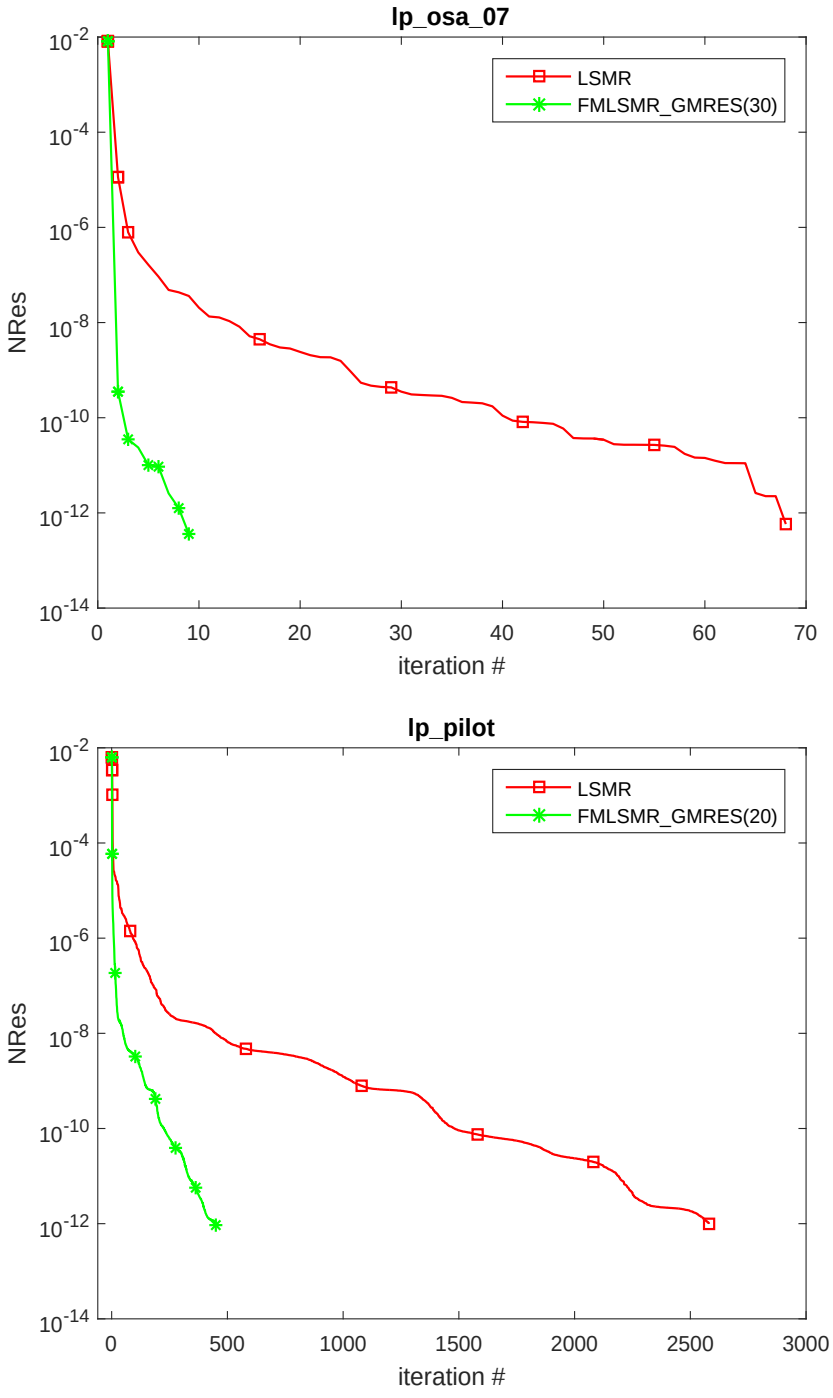

Figure 4.4: NRes *vs.* iteration for `lp_pilot` and `lp_osa_07`

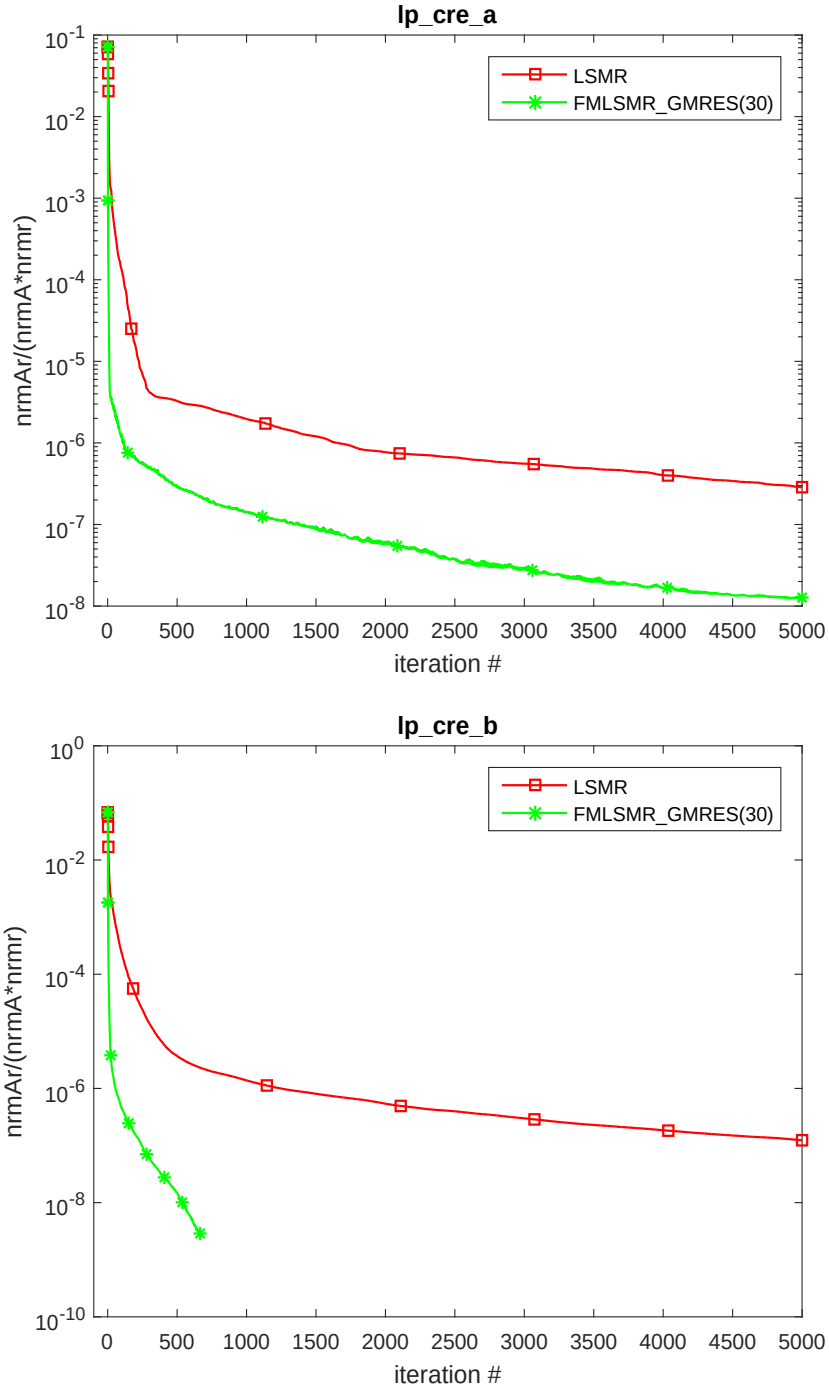

Figure 4.5: Relative Approximate Backward Error *vs.* iteration for `lp_cre_a` and `lp_cre_b`

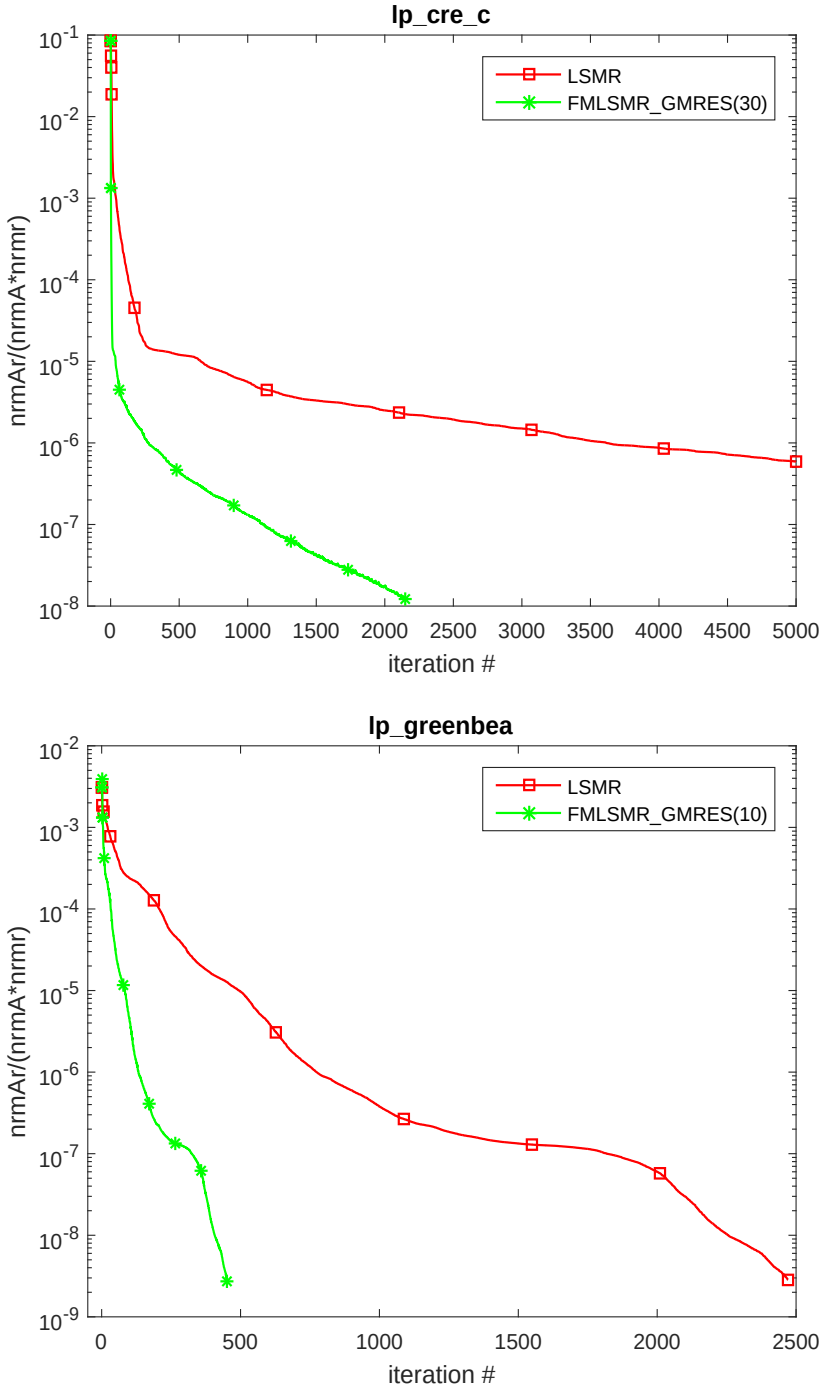

Figure 4.6: Relative Approximate Backward Error *vs.* iteration for `lp_cre_c` and `lp_greenbea`

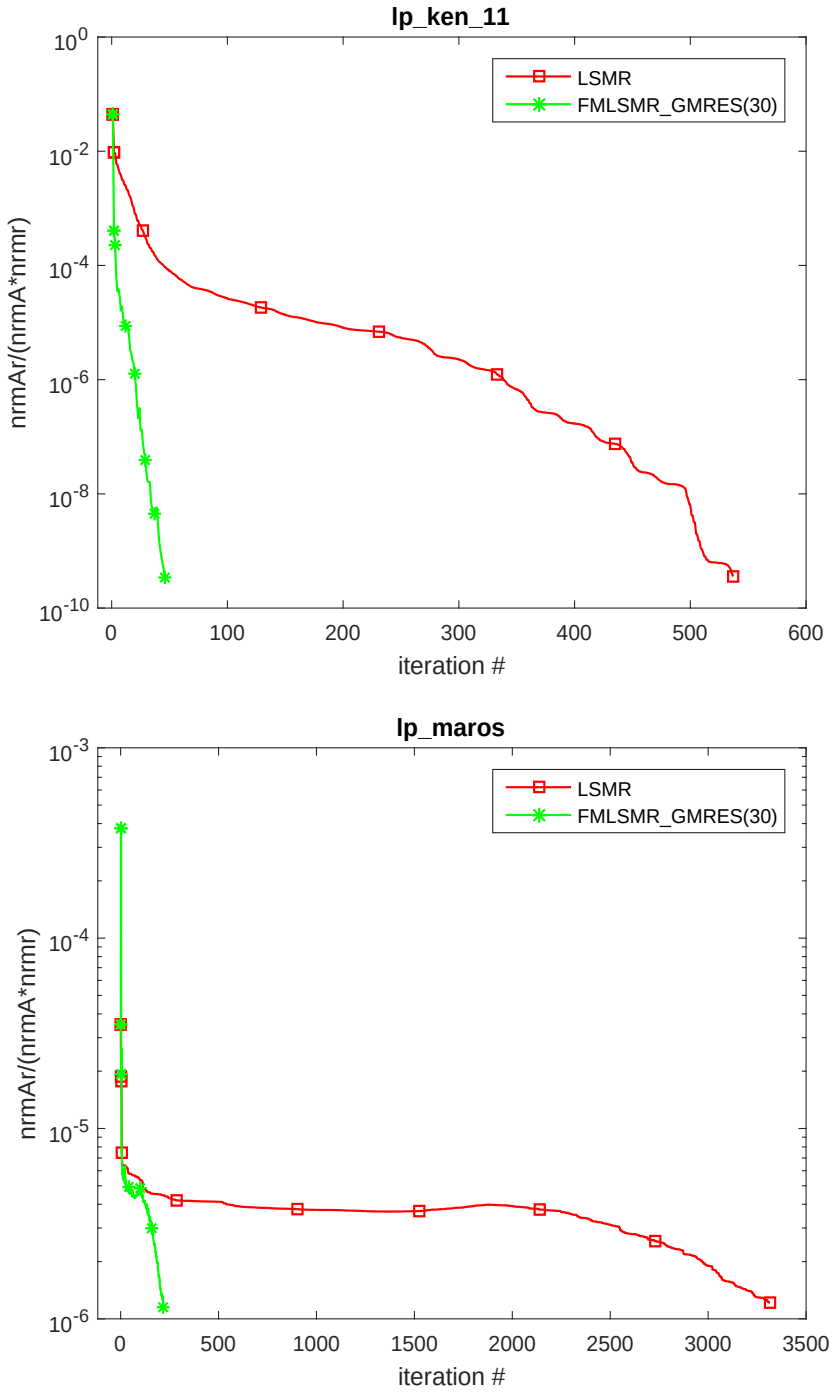

Figure 4.7: Relative Approximate Backward Error *vs.* iteration for `lp_ken_11` and `lp_maros`

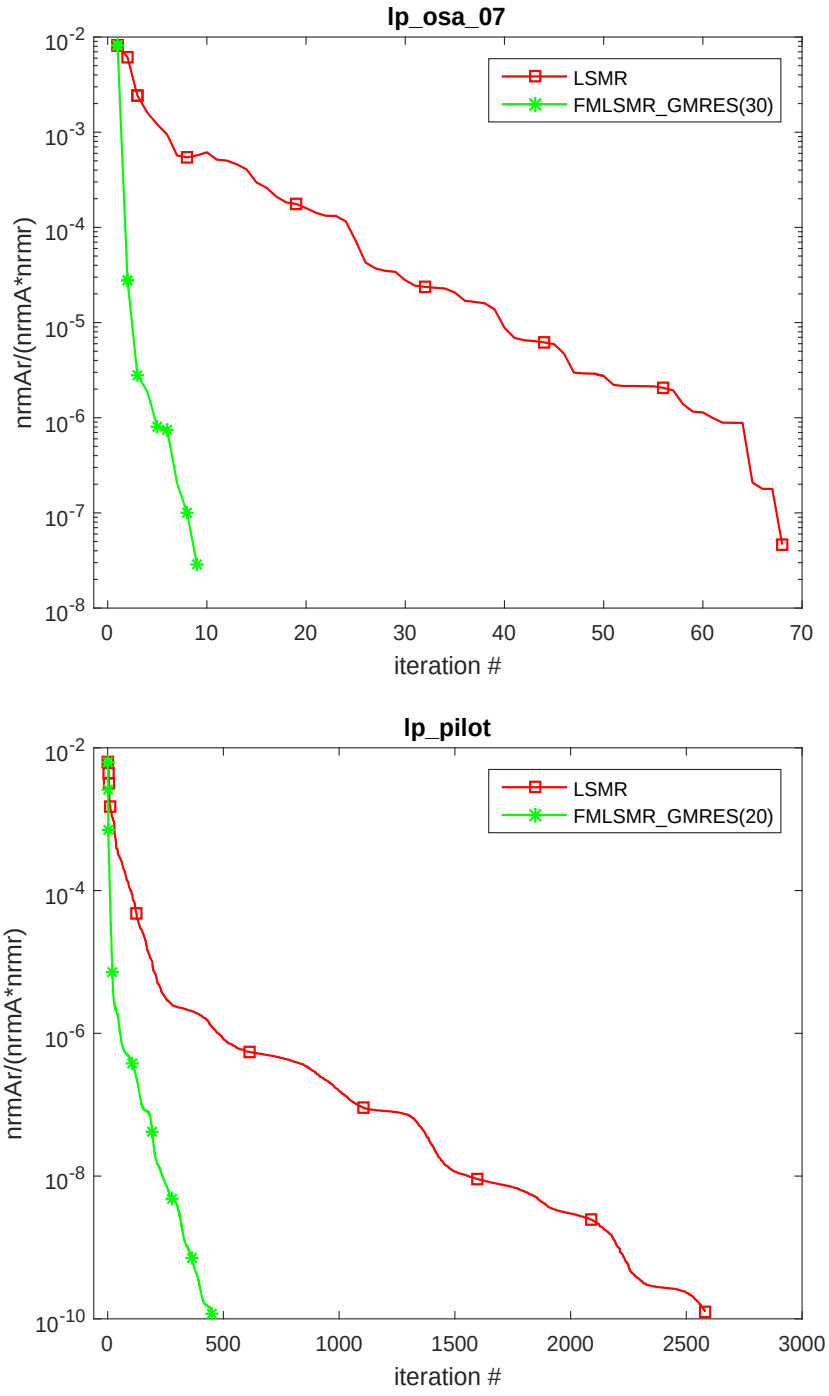

Figure 4.8: Relative Approximate Backward Error *vs.* iteration for `lp_osa_07` and `lp_pilot`

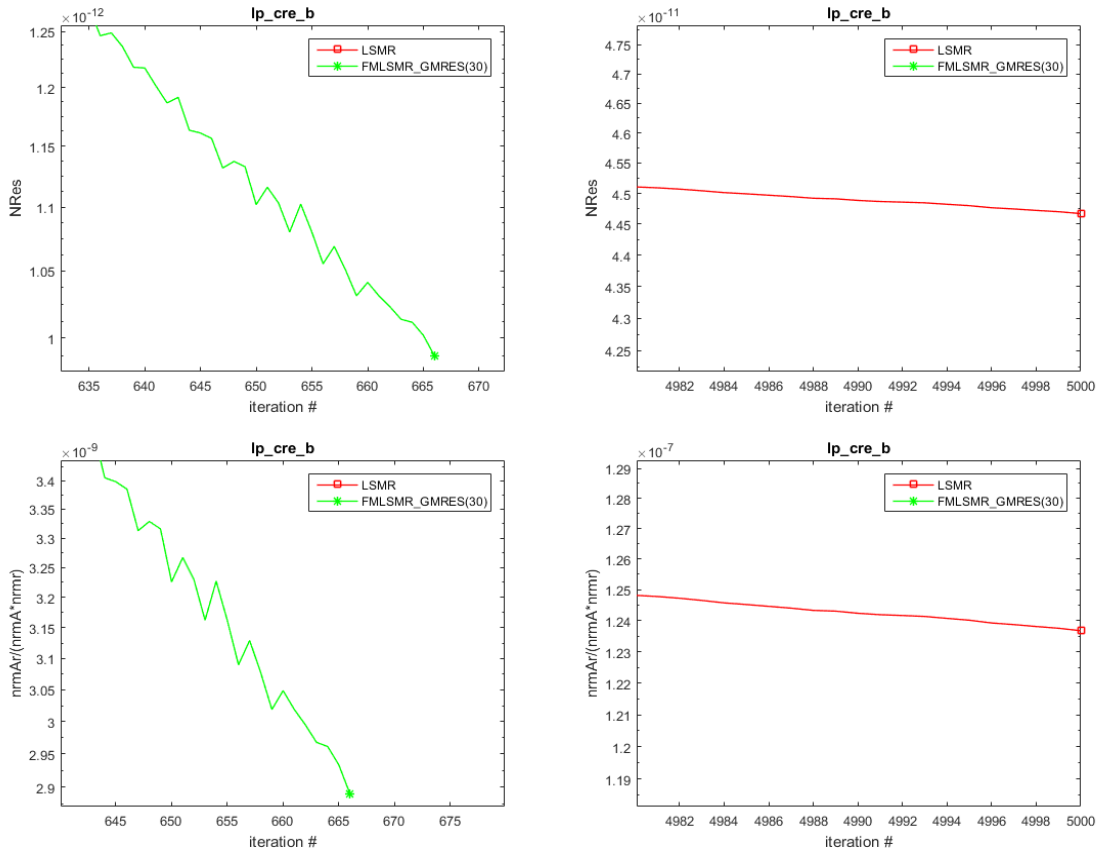

Figure 4.9: Enlarged view of results for `lp_cre_b`. *Top* : NRes *vs.* iteration; *Bottom* : Relative Approximate Backward Error *vs.* iteration

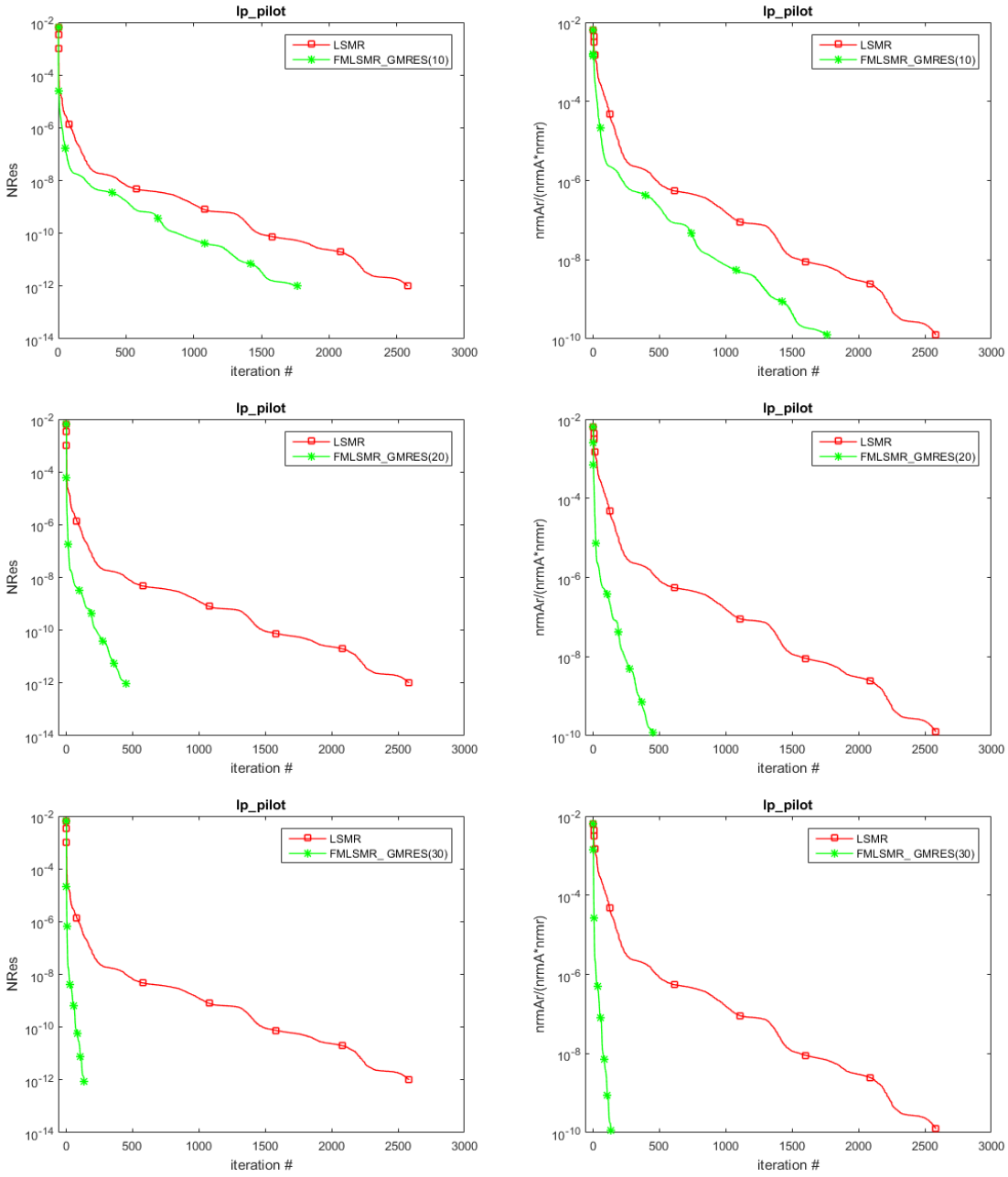

Figure 4.10: Results of different l-step GMRES solving inner iteration for `lp_pilot`. $Left$: NRes *vs.* iteration; $Right$: Relative Approximate Backward Error *vs.* iteration

CHAPTER 5

SUMMARY

5.1 Contributions

In this thesis, we show different techniques to optimize certain Krylov subspace methods for linear systems and least squares problems. We summary our contributions as follows:

- In chapter 2, we presented a heavy ball flexible GMRES method to solve non-symmetric linear systems. This method is a combination of the heavy ball method and the flexible GMRES method to save memory storage and orthogonalization cost as well as making up slow convergence.
- In chapter 3, we proposed a heavy ball minimal residual method to solve least squares problems. This method combines the heavy ball method and LSMR. Our numerical results show that HBMR works better for some examples compared to restarted LSMR and LSMR.
- In chapter 4, we reviewed MLSMR and presented some theoretical analysis for MLSMR. By building an inner-outer iteration in MLSMR, we proposed a flexible preconditioned iterative method (FMLSMR) for least squares problems. The numerical results show faster convergence and more accurate solutions of FMLSMR than LSMR.

5.2 Future Work

We summarize several potential ideas for the future research.

- In chapters 3 and 4, we introduced the approximate optimal backward error and the approximate backward error to estimate the backward error, repectively. In our numerical tests, we found that the approximate backward error is not monotonically decreasing for some examples, while the approximate optimal backward error has the same pattern as backward error for all the testing examples. So how to distinguish which one we need to use is a question. Another question is how to find an efficient way to accurately compute backward error instead of using some approximate backward error.
- In chapter 4, we introduce an inner iteration to the Golub-Kahan process by solving an normal-type equation $A^TAv = p$. Numerical experiments show that this technique works well in the flexible factorization-free LSMR. So we are wondering is that possible to apply this techique to some methods for underdetermine or regularized least squares problems? Or can we try to apply FMLSMR to saddle point problems [39] or other types of least squares problems like total least squares and constrained least squares problems [50]?

REFERENCES

[1] *Matrix market*, https://doi.org/https://math.nist.gov/MatrixMarket/.

[2] *Suitesparse matrix collection*, https://doi.org/https://sparse.tamu.edu/.

[3] W. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.

[4] S. R. Arridge, M. M. Betcke, and L. Harhanen, *Iterated preconditioned LSQR method for inverse problems on unstructured grids*, Inverse problem, 36 (2015), pp. 225–250.

[5] H. Avron, E. NG, and S. Toledo, *Using perturbed QR factorizations to solve linear least squares problems*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 674–963.

[6] O. Axelsson, *Conjugate gradient type-method for unsymmetric and inconsisitent systems*, Linear Algebra Appl., 29 (1980), pp. 1–16.

[7] O. Axelsson and P. Vassilevski, *Algebraic multilevel preconditioning methods, I,*, Numer. Math., 56 (1989), pp. 157–177.

[8] O. Axelsson and P. Vassilevski, *Algebraic multilevel preconditioning methods, II,*, SIAM J. Numer. Anal., 27 (1990), pp. 1569–1590.

[9] M. W. Benson and P. O. Erederickson, *Iterative solution of large sparse linear systems arising in certain multimensional approximation problems*, Utilitas Math., 22 (1982), pp. 127–140.

[10] M. Benzi, *Preconditioning techniques for leage linear systems: a survey*, Journal of Computational Physics, 182 (2002), pp. 418–477.

[11] M. Benzi, J. C. Haws, and M. Tuma, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353.

[12] M. Benzi and M. Tůma, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30 (1999), pp. 305–340.

[13] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.

[14] P. Concus, G. H. Golub, and G. Meurant, *Block preconditioning for the conjugate gradient method*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 220–252.

[15] C. C.Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.

[16] J. W. Demmel, *Applied Nmerical Linear Algebra*, SIAM, Philadephia, 1997.

[17] H. A. V. der Vorst and C. Vuik, *GMRESR: a family of nested GMRES methods*, Numer. Linear Algebra Appl., (1994), pp. 369–386.

[18] S. C. Eisenstat, H. C. Elman, and H. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.

[19] D. J. Evans, *The use of pre-conditioning in iterative methods for solving linear equations with symmetric positive definite matrices*, J. Inst. Math. Appl., 4.

[20] D. K. Faddeev and V. N. Faddeeva, *Computational mothods for linear algebra*, Freeman and Company, San Fransisco, 1956.

[21] R. Fletcher, *Conjugate gradient methods for indefinite systems*, Journal of Research of the National Bureau of Standards, 49 (1952), pp. 409–436.

[22] D. C. Fong, *Minimum-Rsidual Methods for Sparse Least-Squares Using Golub-Kahan Bidiagonalization*, PhD thesis, Stanford University, Stanford, MA, 2011.

[23] D. C. Fong and M. A. Saunders, *LSMR: an iterative algorithm for sparse least-squares problems*, SIAM J. Sci. Comput., 33 (2011), pp. 2950–2971.

[24] R. W. Freund and N. M. Nachtigal, *QMR: a quasi minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[25] G.Golub and W.Kahan, *Calculating the singular values and pseduo-inverse of a matrix*, SIAM J. Numer. Anal. Ser. B, 8 (1965), pp. 205–224.

[26] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials*, SIAM, Philadephia, 2009.

[27] G. H. Golub and C. F. V. Loan, *Matrix Computations*, Johns Hopkins, Baltimore, 4th ed., 2013.

[28] J. Grcar, *How ordinary elimination became gaussian elimination*, Historica Mathematica, 38 (2011), pp. 163–218.

[29] J. Grcar, *How ordinary elimination became gaussian elimination*, Notice of the AMS, 58 (2011), pp. 782–792.

[30] A. Greenbaum, *Iterative methods for solving linear systems*, SIAM, Philadephia, 1997.

[31] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverse*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[32] I. Gustafsson, *A class of first order factorization methods*, BIT, 18 (1978), pp. 142–156.

[33] M. R. Hestenes, *The conjugate-gradient method for solving linear systems*, in: Sympos. Appl. Math., Numerical Analysis, Vol. VI, McGraw-Hill, New York, 1956, pp. 83–102.

[34] M. R. Hestenes and E. Stiefe, *Methods of conjugate gradients for solving linear systemsl*, Journal of Research of the National Bureau of Standards, 49 (1952), pp. 409–436.

[35] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadephia, 2nd ed., 2002.

[36] A. S. Householder, *Unitary triangularization of a nonsymmetric matrix*, Journal of the ACM, 5 (1958), pp. 339–342.

[37] A. Imakura, R. C. Li, and S. L. Zhang, *Locally optimal and heavy ball GMRES methods*, Japan J. Indust. Appl. Math., 33 (2016), pp. 471–499.

[38] K. C. Jea and D. M. Young, *Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra Appl., 20 (1980), pp. 159–194.

[39] G. Karaduman, *Numerical Solution of Saddle Point Problems by Projection*, PhD thesis, University of Texas at Arlington, Arlington, TX, 2017.

[40] R. Karlson and B. Waldén, *Estimation of optimal backward pertubation bounds for the linear least squares problem*, BIT, 34 (1997), pp. 862–869.

[41] L. Y. Kolotilina and A. Y. Yeremin, *On a family of two-level preconditionings of the incomplete block factorization type*, Soviet J. Numer. Anal. Math. Modelling, 1 (1986), pp. 293–320.

[42] L. Y. Kolotilina and A. Y. Yeremin, *Factorized sparse approximate inverse preconditionings I, Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.

[43] C. Lanczos, *Chebyshev polynomials in the solution of large-scale linear systems*, Toronto Symposium on Computing Techniques, (1952), pp. 124–133.

[44] C. Lanczos, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 33–53.

[45] N. Li and Y. Saad, *MIQR: A mutilevel incomplete QR preconditioner for large sparse least-squares problems*, SIAM J. Matrix Anal. & Appl., 28 (2006), pp. 524–550.

[46] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[47] K. Morikuni and K. Hayami, *Convergence of inner-iteration GMRES methods for rank-deficient least squares problems*, SIAM J. Sci. Comput., 36 (2015), pp. 225–250.

[48] C. C. Paige and M. A. Saunders, *Solutions of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[49] B. T. Polyak, *Introduction to optimization*, Optimization Software, New York, 1987.

[50] Å. Björck, *Numerical methods for least squares problems*, SIAM, Philadephia, 1996.

[51] Y. Saad, *Krylov subspace methods for solving large unsymmetric linear systems*, Mathematics of Computation, 37 (1981), pp. 105–126.

[52] Y. Saad, *Preconditioning techniques for indefinite and nonsymmetric linear systems*, J. Comput. Appl. Math., 24 (1988), pp. 89–105.

[53] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[54] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadephia, 2nd ed., 2003.

[55] Y. Saad and M. H. Schultz, *GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. and Statist. Comput., 7 (1986), pp. 856–869.

[56] Y. Saad and H. A. van der Vorst, *Iterative solution of linear systems in the 20th century*, J. Comput. and Appl. Math., 123 (2000), pp. 1–33.

[57] G. L. G. Sleijpen and D. R. Fokkema, *BICGSTAB(ℓ) for linear equations involving unsymmetric matrices with complex spectrum*, ETAN, 1 (1993), pp. 11–32.

[58] P. Sonnoveld, *CGS: a fast Lanczos-type olver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.

[59] G. W. Stewart, *Research development and LINPACL*, in Mathematical Software III, J. R. Rice, ed., Academic Press, 1977, pp. 1–14.

[60] E. Stiefel, *Relaxationsmethoden bester strategie zur lösung linearer*, Journal of Research of the National Bureau of Standards, 49 (1952), pp. 409–436.

[61] E. L. Stiefel, *Kernel polynomials in linear algebra and their applications*, U. S. Nat. Bur. Standards Appl. Math. Ser., 49 (1958), pp. 1–24.

[62] Z. Su, *Computational Methods for Least Squares Problems and Clinical Trials*, PhD thesis, Stanford University, Stanford, CA, 2005.

[63] H. A. van der Vorst, *Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[64] P. K. W. Vinsome, *ORTHOMIN: an iterative method solving sparse sets of simultaneous linear equations*, Proceedings of the Fourth Symposium on Reservoir Simulation, Society of Petroleum Enjineers of AIME, 1976, pp. 149–159.

[65] C. Vuik, *New insights in GMRES-like methods with variable preconditioners*, Journal of Computational and Applied Mathematics, 61 (1999), pp. 189–204.

[66] J. W. Watts, *A conjugate gradient truncated direct method for the iterative solution of the reservoir simulation pressure equation*, Soc. Petroleum Eng. J., 21 (1981), pp. 345–353.

[67] D. M. Young, *Iterative Methods for Solving Partial Difference Equations of Elliptic Type*, PhD thesis, Harvard University, Cambridge, MA, 1950.

BIOGRAPHICAL STATEMENT

Mei Yang was born in Xingping, China, in 1985. She received her Bachelor of Science degree in Mathematics from China University of Mining and Technology, Master of Science degree in Mathematics from Xi'an Jiaotong University in 2008 and 2012 respectively. She joined the Ph.D program in Mathematics at the University of Texas at Arlington in the fall of 2012.

Mei's research interest includes numerical linear algebra, numerical analysis, data mining, and statistics.