Japan J. Indust. Appl. Math.
DOI 10.1007/s13160-016-0220-1

ORIGINAL PAPER

Area 2

Locally optimal and heavy ball GMRES methods

Akira Imakura[1] · **Ren-Cang Li**[2,3] · **Shao-Liang Zhang**[4,5]

Received: 25 November 2015 / Revised: 18 May 2016

Abstract The restarted GMRES (REGMRES) is one of the well used Krylov subspace methods for solving linear systems. However, the price to pay for the restart usually is slower speed of convergence. In this paper, we draw inspirations from the locally optimal CG and the heavy ball methods in optimization to propose two variants of the restarted GMRES that can overcome the slow convergence. Compared to various existing hybrid GMRES which are also designed to speed up REGMRES and which usually require eigen-region estimations, our variants preserve the appealing feature of

Supported in part by Strategic Programs for Innovative Research (SPIRE) Field 5 "The origin of matter and the universe", JSPS Grants-in-Aid for Scientific Research (Nos. 25870099, 25286097), MEXT Grants-in-Aid for Scientific Research (No. 22104004), NSF grants DMS-1115834 and DMS-1317330, NSF CCF-1527104 a Research Gift Grant from Intel Corporation, and NSFC Grant 11428104.

✉ Akira Imakura
imakura@cs.tsukuba.ac.jp

Ren-Cang Li
rcli@uta.edu

Shao-Liang Zhang
zhang@na.cse.nagoya-u.ac.jp

[1] Faculty of Engineering, Information and Systems, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan

[2] School of Mathematical Science, Xiamen University, Xiamen, People's Republic of China

[3] Department of Mathematics, University of Texas at Arlington, P.O. Box 19408, Arlington, TX 76019-0408, USA

[4] Department of Computational Science and Engineering, Graduate School of Engineering, Nagoya University, Furo-Cho, Chikusa-Ku, Nagoya 464-8603, Japan

[5] CREST, JST, 5 Sanbancho, Chiyoda-ku, Tokyo 102-0075, Japan

Published online: 09 June 2016

GMRES and REGMRES—their simplicity. Numerical tests on real data are presented to demonstrate the superiority of the new methods over REGMRES and its variants.

Keywords GMRES · Restarted GMRES · Locally optimal GMRES · Heavy ball GMRES · Linear system

Mathematics Subject Classification 65F10

1 Introduction

The generalized minimal residual method (GMRES) [29] is often used to solve a (nonsymmetric) linear system $Ax = b$. The basic idea is to seek approximate solutions, optimal in a certain sense, from the so-called Krylov subspaces. Given an initial approximation x_0, the kth approximation x_k is sought so that the kth residual $r_k = b - Ax_k$ satisfies

$$\|r_k\|_2 = \min_{z \in \mathcal{K}_k(A, r_0)} \|b - A(x_0 + z)\|_2, \tag{1.1}$$

where the kth Krylov subspace of A on r_0 is defined as

$$\mathcal{K}_k(A, r_0) = \operatorname{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}. \tag{1.2}$$

For implementation, it is realized via the k-step Arnoldi process that produces an orthonormal basis of $\mathcal{K}_{k+1}(A, r_0)$.

For large scale linear systems, GMRES can be very expensive for large k. The need to store all the basis vectors in the Arnoldi process places a heavy burden on memory, and the orthogonalization cost in computing the orthonormal basis vectors grows quadratically with k. Hence, for a difficult system that requires large k to converge, GMRES may become too expensive in both memory and arithmetic operations to use, with high memory demand, in particular, that may make it impractical. For this consideration, the so-called *restarted GMRES* [29] is naturally developed. The simplest version of the restarted GMRES is: fix k and repeatedly iterate the k-step GMRES with the current initial guess being the very previous k-step GMRES solution. Such a restarted GMRES is often denoted by GMRES(k) [11,29], but for notational consistency with its two variants later, we will denote it by REGMRES(k) or simply REGMRES when the parameter k is either clear from the context or not particularly important for the discussion. We will also call each k-step GMRES run in REGMRES(k) a GMRES *cycle*.

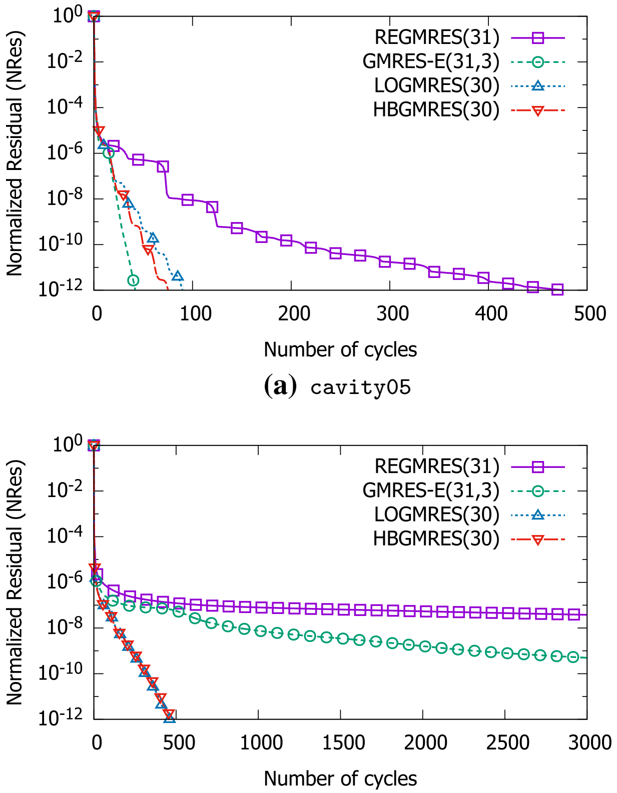
REGMRES successfully alleviates possibly heavy memory burden and orthogonalization cost by limiting the number of Arnoldi steps in each GMRES cycle, but not without any tradeoff. In particular, there is possibly severe degradation in the convergence behavior. In fact, the success of the k-step GMRES is closely related to polynomial approximations [5,11,18,27] by polynomials of degree k or less. Experience shows that sometimes it may need a big enough k to achieve any noticeable reduction in the residual norm $\|r_k\|_2$ and the reduction usually accelerates as k gets bigger. In the case of REGMRES(k), k is fixed (or limited) and thus each GMRES cycle does not get the benefit in residual reduction from increasing k as in the usual GMRES. That is the most significant shortcoming of REGMRES(k) compared to GMRES. How to improve on this defect so as to make REGMRES(k) converge faster becomes an imminent and practically important question.

For REGMRES(k), each GMRES cycle builds a Krylov subspace of dimension k for computing the approximate solution of the cycle and the approximation is used as the initial guess for the next cycle. As soon as the approximation is computed, the Krylov subspace is thrown away for the purpose of limiting memory usage and controlling orthogonalization cost. Indeed, this goal of saving is successfully achieved, but is there any thing one could do to salvage the speed of convergence lost by the throwing-away? In this article, we will propose two variations of REGMRES, inspired by two optimization techniques: one from locally optimal conjugate gradient methods (LOCG) [24,31] and the other from the heavy ball method [24, p. 65]. The two respective variations are therefore called the *locally optimal GMRES* (LOGMRES) and the *heavy ball GMRES* (HBGMRESS). Both methods keep the benefit of REGMRES in limiting memory usage, controlling orthogonalization cost, and simplicity in implementation. Numerical tests show that they are also able to salvage the lost convergence speed in REGMRES. As a demonstration, Fig. 1 plots normalized residual norm vs. cycle for REGMRES, LOGMRES, HBGMRES and the deflated GMRES (GMRES-E) [21], which is one of the most succesful improvements of GMRES, for `cavity05` and `chipcool0` from the University of Florida sparse matrix collection [4], where the numbers (31) and (30) attached to `xx`GMRES tell the orders of the Krylov subspaces used in a cycle and they are chosen to make the cost of a cycle for each method roughly the same. We see that the improvement is nothing short of being spectacular.

There are many other hybrid GMRES methods that were designed to improve GMRES, too. Generally speaking, these hybrid methods follow the paradigm: (1) perform an initial GMRES run to estimate the eigenvalue region of A, (2) construct reducing polynomials $p_k(t)$ to aim at making $\|p_k(A)\|$ small in a sense, and (3) use Richardson/Chebyshev iterations which don't need orthogonalization. Nachtigal, Reichel, and Trefethen [23] briefly surveyed much of the development along this line (see, e.g., [7,8,19,26,30]) before 1992, explained pros and cons of using spectral information, and proposed an easy-to-use hybrid GMRES.

As a direct modification to REGMRES, Morgan [21,22] proposed a deflated restarting strategy to include eigenvectors associated with the few smallest eigenvalues (in magnitude) into the Krylov subspaces of GMRES cycles. These eigenvectors are usually not available a priori and thus have to be estimated also in previous GMRES cycles. Morgan [21, section 5] discussed various delicate practical issues that must be dealt with.

The most appealing feature of GMRES and REGMRES that makes them popular is their simplicity in implementation. They may well be picked up by a practitioner from a numerical recipe book to quickly embed it, without much effort, into their programs, as opposed to methods that require some eigen-estimation phase that generally do not enjoy such easiness in use. They include all existing developments mentioned above, except the method of Nachtigal, Reichel, and Trefethen [23] who proposed to simply cyclically reuse the GMRES residual polynomial from the initial GMRES run



(a) `cavity05`

(b) `chipcool0`

Fig. 1 Normalized residual norm vs. cycle for `cavity05` and `chipcool0` [4]. Roughly the cost of a cycle for REGMRES and proposed methods: LOHBGMRES and HBGMRES are about the same. To reach about 10^{-12} in normalized residual norm, on `cavity05` REGMRES takes 477 cycles while new LOGMRES and HBGMRES take 90 and 76 cycles, representing speedups of 5.3 and 6.3, respectively. Even more dramatically on `chipcool0`, LOGMRES and HBGMRES take 464 and 491, respectively, while REGMRES and GMRES-E are not going to get there anytime soon. Both LOGMRES and HBGMRES are just as easy to implement as REGMRES, an appealing feature not shared by almost all other variations of GMRES

until convergence. The needs for eigen-estimations complicate implementations due to practical issues that require expertly care. These issues include which and how many estimated eigenvalues and eigenvectors are considered essential in Morgan's approach and how good the estimated regions should be in order to achieve respectable convergence in later Richardson/Chebyshev iterations. Unfortunately, there is no definitive way to handle these issues and in fact they differ from one implementation to another, resulting in wide differences in performance. Compared with existing approaches, our methods share the same simplicity as GMRES and REGMRES in practical uses; so does the method of Nachtigal, Reichel, and Trefethen [23] who reported mostly superior performance to (RE)GMRES on three banded Toeplitz matrices, one artificial bi-diagonal and one artificial diagonal matrices. But in our experiments, their method has hard time solving real life matrices, including `cavity05` and `chipcool0` [4]. More detail is in Sect. 5.

The rest of this paper is organized as follows. Section 2 reviews GMRES and REGMRES and briefly discuss why REGMRES loses out to GMRES in residual norm reduction. Section 3 presents the algorithmic framework of LOGMRES and HBGMRESS whose implementation details are discussed in Sect. 4. Section 5 gives our numerical results on two experiments: one is for the five testing matrices in [23], and the other is for 10 real life problems from the University of Florida sparse matrix collection. Finally, we give our concluding remarks in Sect. 6.

Notation Throughout this paper, $\mathbb{C}^{n\times m}$ is the set of all $n \times m$ complex matrices, $\mathbb{C}^n = \mathbb{C}^{n\times 1}$, and $\mathbb{C} = \mathbb{C}^1$. I_n (or simply I if its dimension is clear from the context) is the $n \times n$ identity matrix, and e_j is its jth column. The superscript "$\cdot^{\mathrm{T}}$" and "$\cdot^{\mathrm{H}}$" takes the transpose and the complex conjugate transpose of a matrix or vector, respectively. We shall also adopt MATLAB-like convention to access the entries of vectors and matrices. Let $i{:}j$ be the set of integers from i to j inclusive. For a vector u and a matrix X, $u_{(j)}$ is u's jth entry, $X_{(i,j)}$ is X's (i, j)th entry; X's submatrices $X_{(k:\ell,i:j)}$, $X_{(k:\ell,:)}$, and $X_{(:,i:j)}$ consist of intersections of row k to row ℓ and column i to column j, row k to row ℓ, and column i to column j, respectively. Notation $\|\cdot\|_2$ is either the matrix spectral norm or the Euclidean vector norm, depending on its arguments:

$$\|x\|_2 = \sqrt{\sum_i |x_{(i)}|^2}, \quad \|A\|_2 := \max_{x\neq\mathbf{0}} \frac{\|Ax\|_2}{\|x\|_2}.$$

2 Restarted GMRES

2.1 GMRES

GMRES was proposed by Saad and Schultz [29]. We outline it below as the basis for our later development. Recall (1.2). Assume, for the moment, that

$$\dim \mathcal{K}_{k+1}(A, r_0) = k + 1, \tag{2.1}$$

and let $\{v_1, v_2, \ldots, v_{k+1}\}$ be the basis of $\mathcal{K}_{k+1}(A, r_0)$ generated by the Arnoldi process as given by Lines 1-13 in Algorithm 1. The assumption (2.1) ensures that the test at Line 7 turns out true for all $1 \le i \le k$. Denote by

$$V_j = [v_1, v_2, \ldots, v_j]. \tag{2.2}$$

Compactly, the process can be expressed as

$$AV_k = V_k H_k + v_{k+1} h_{k+1\,k} e_k^{\mathrm{T}} =: V_{k+1} \check{H}_k, \tag{2.3}$$

where

$$H_k = V_k^{\mathrm{H}} A V_k \in \mathbb{C}^{k\times k}, \quad \check{H}_k = \begin{bmatrix} H_k \\ h_{k+1\,k} e_k^{\mathrm{T}} \end{bmatrix} \in \mathbb{C}^{(k+1)\times k}. \tag{2.4}$$

Algorithm 1 (k-step) GMRES

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

```
$r_0 = b - Ax_0$, $\beta = \|r_0\|_2$;
$V_{(:,1)} = r_0/\beta$, $\check{H} = 0_{(k+1)\times k}$ (the $(k+1) \times k$ zero matrix);
For $i = 1, 2, \dots, k$ Do:
    $f = AV_{(:,i)}$;
    $\check{H}_{(1:i,i)} = V^{\mathrm{H}}_{(:,1:i)} f$, $f = f - V_{(:,1:i)} \check{H}_{(1:i,i)}$;
    $\check{H}_{(i+1,i)} = \|f\|_2$;
    If $\check{H}_{(i+1,i)} > 0$ Then
        $V_{(:,i+1)} = f/\check{H}_{(i+1,i)}$;
    else
        reset $k = i$, $\check{H} = \check{H}_{(1:i,1:i)}$;
        BREAK;
    End If
End For
$y = \arg\min_y \|\check{H}y - \beta e_1\|_2$;
return $x_k = x_0 + Vy$ as an approximate solution to $Ax = b$.
```

Both H_k and $\check{H}_k$ are upper-Hessenberg.[1] In particular, $v_1 = r_0/\|r_0\|_2$.

In GMRES, we have to solve (1.1). Write $x = x_0 + z$. Any $z \in \mathcal{K}_k(A, r_0)$ can be expressed as $z = V_k y$ for some $y \in \mathbb{C}^k$, and thus

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

Therefore

$$\min_{z \in \mathcal{K}_k(A,r_0)} \|b - A(x_0 + z)\|_2 = \min_{z \in \mathcal{K}_k(A,r_0)} \|Az - r_0\|_2 = \min_{y \in \mathbb{C}^k} \|\check{H}_k y - \beta e_1\|_2, \tag{2.5}$$

where $\beta = \|r_0\|_2$. Solving the last problem in (2.5) 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 [11,29]

$$x_k = x_0 + z_{\text{opt}} = x_0 + V_k y_{\text{opt}}. \tag{2.6}$$

In the case when

$$\dim \mathcal{K}_{k+1}(A, r_0) = j \leq k, \tag{2.7}$$

instead of (2.3) we have $AV_j = V_j H_j$, where $H_j = V_j^{\mathrm{H}} AV_j$. Correspondingly

$$\min_{z \in \mathcal{K}_k(A,r_0)} \|b - A(x_0 + z)\|_2 = \min_{z \in \mathcal{K}_j(A,r_0)} \|Az - r_0\|_2 = \min_{y \in \mathbb{C}^j} \|H_j y - \beta e_1\|_2 = 0$$

because H_j is nonsingular. Finally $x = V_j H_j^{-1}(\beta e_1)$ is the exact solution. This is true in the absence of roundoff errors. As we know roundoff errors are unavoidable, but we hope that the algorithm is implemented robust enough, an acceptable solution is computed in such a case.

[1] A (square or rectangular) matrix X is upper-Hessenberg if $X_{(i,j)} = 0$ for all $i > j + 1$.

We summarize GMRES as in Algorithm 1 which can handle both (2.1) and (2.7). In the latter, the test at Line 7 will fail, i.e., $\check{H}_{(i+1,i)} = 0$ for some i and consequently the ensuing $\check{H}$ at Line 14 is a square and nonsingular matrix and the corresponding least squares problem $\min_y \|\check{H}y - \beta e_1\|_2$ becomes a usual nonsingular linear system.

Remark 1 GMRES in Algorithm 1 is not stated in the most sophisticated way as far as its implementation is concerned.

1. At Line 5, it simply uses the classical Gram–Schmidt (CGS) orthogonalization to orthogonalize $AV_{(:,i)}$ against already computed columns of V. This is faster computationally in actual execution but may produce less orthogonal vectors than the modified Gram–Schmidt (MGS) orthogonalization:

```
for $j = 1, 2, \ldots, i$ do
    $\check{H}_{(j,i)} = V_{(:,j)}^{\mathrm{H}} f$, $f = f - V_{(:,j)}\check{H}_{(j,i)}$;        (2.8)
end for
```

   In practice, some form of re-orthogonalization may be needed. This is a subtle issue and can affect numerical performance. We will discuss it later in the numerical example section.
2. At Line 7, the test should be implemented as "if $\check{H}_{(i+1,i)} > \mathtt{tol} \times \|AV_{(:,i)}\|_2$" for some `tol` in the order of the machine roundoff $\mathfrak{u}$ (which is 2^{-24} for the IEEE single precision or 2^{-53} for the IEEE double precision). Often a good choice would be $n\mathfrak{u}$ or $\sqrt{n}\,\mathfrak{u}$.

2.2 Restarted GMRES

For large scale linear systems, GMRES can be very expensive for large k. The need to store all the basis vectors v_j places a heavy demand on memory, and the orthogonalization cost in computing v_j increases quadratically in k. The so-called restarted GMRES is naturally born in an attempt to control memory usage and orthogonalization cost by fixing k and repeatedly iterate the k-step GMRES with the current initial guess $x_0^{(\ell)}$ being the very previous k-step GMRES solution, where it is assumed that the initial guess $x_0^{(1)}$ for the first k-step GMRES, is given. We will denote it by REGMRES(k) whose framework is sketched in Algorithm 2 which has inner iterations at Line 6 and outer iterations indexed by ℓ. To distinguish the two types of iterations, in what follows, we will refer to each outer iteration—a call to Algorithm 1, by a GMRES cycle or simple a cycle. This applies to the algorithms in the next section, too.

While REGMRES(k) successfully alleviates possibly heavy memory burden and orthogonalization cost by limiting the number k of Arnoldi steps, it does so with tradeoff in possibly severe degradation in convergence behavior. In Algorithm 2, it can be seen that

$$r_0^{(\ell)} = b - Ax_0^{(\ell)} \in \mathcal{K}_{k+1}(A, r_0^{(\ell-1)}).$$

Algorithm 2 REGMRES(k)

Given any initial guess $x_0^{(1)} \in \mathbb{C}^n$ 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** $\|r_0^{(\ell)}\|_2 \leq \texttt{tol} \times (\|A\|_2 \|x_0^{(\ell)}\|_2 + \|b\|_2)$ **Then**
        **BREAK**;
    **else**
        call Algorithm 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 $Ax = b$.
```

Therefore for $\ell \geq 2$

$$x_k^{(\ell)} \in x_0^{(\ell)} + \mathcal{K}_k(A, r_0^{(\ell)}) \in x_k^{(\ell-1)} + \mathcal{K}_{2k}(A, r_0^{(\ell-1)}) \in \cdots \in x_0^{(1)} + \mathcal{K}_{\ell k}(A, r_0^{(1)}),$$

i.e., $x_k^{(\ell)} = x_0^{(1)} + z$ with $z \in \mathcal{K}_{\ell k}(A, r_0^{(1)})$. Naturally, we wonder how good this $x_k^{(\ell)}$ is, compared to the (ℓk)-step GMRES solution $x_{\ell k}^{(1)}$, i.e., the one returned by Algorithm 1 with inputs $x_0^{(1)}$ and the integer ℓk. We know that $x_k^{(\ell)}$ is always no better than the latter in the sense that

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

but how much worse could it be? Unfortunately, there are no shortage of examples for which

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

The question is if there is anything one could do to narrow down the gap in magnitude between the two residuals $\|b - Ax_k^{(\ell)}\|_2$ and $\|b - Ax_{\ell k}^{(1)}\|_2$, without adding too much complexity in cost (or better yet no extra cost to Algorithm 2) and preserving the easiness in implementation.

In Algorithm 2, the order k of the Krylov subspaces in each cycle is stated as an input for simplicity. In practice, it can also be determined computationally in the initial cycle, much like the strategy in [23, (6.5)], so that the ratio $\|r_k^{(1)}\|_2 / \|r_k^{(0)}\|_2$ falls below a preset tolerance, say 10^{-1}.

3 LOGMRES

Restarted GMRES—Algorithm 2—controls memory usage and orthogonalization cost by limiting the number of the Arnoldi steps per GMRES call (to Algorithm 1). The initial guess for the current GMRES cycle is the approximate GMRES solution of the very previous cycle. Other than that, all the Krylov subspaces built in the previous cycles are completely ignored for the purpose of cost control in memory and flops.

If, somehow, we could use more information concealed in these Krylov subspaces other than just in the initial guess part to speed up convergence and at the same time without adding significant cost and complication in implementation, it would represent a significant improvement. Using these Krylov subspaces in a straightforward way like seeking the best approximation in the sum of these subspaces is clearly not an option because it would mean to keep and orthogonalize all the basis vectors, the very thing that Algorithm 2 was developed to avoid.

Our idea for the locally optimal restarted generalized minimal residual method (LOGMRES) is inspired by locally optimal conjugate gradient methods (LOCG) in optimization [24,31]. It brings in more information in the previous cycles by just including the approximation before the last. We shall now outline the idea.

The steepest descent method (SD) is perhaps the most primitive method in minimizing a differentiable function $f(x)$:

$$\min_x f(x).$$

For each step, SD starts at an approximation $\boldsymbol{x}^{(\ell)}$ to an optimum and seeks the next approximation by the line-search:

$$t_{\text{opt}} = \min_t f(\boldsymbol{x}^{(\ell)} + t\nabla f(\boldsymbol{x}^{(\ell)})), \quad \boldsymbol{x}^{(\ell+1)} = \boldsymbol{x}^{(\ell)} + t_{\text{opt}}\nabla f(\boldsymbol{x}^{(\ell)}), \tag{3.1}$$

where ∇f is the gradient of the function f. In doing so, $\boldsymbol{x}^{(\ell+1)}$ is determined only by the information at $\boldsymbol{x}^{(\ell)}$, completely ignoring approximations before $\boldsymbol{x}^{(\ell)}$ that lead to $\boldsymbol{x}^{(\ell)}$. This makes SD very friendly to implement, but at the same time converge slowly sometimes. An LOCG improves SD through incorporating the very previous approximation $\boldsymbol{x}^{(\ell-1)}$:

$$\boldsymbol{x}^{(\ell+1)} = \arg \min_{x\in\text{span}\{\boldsymbol{x}^{(\ell)},\nabla f(\boldsymbol{x}^{(\ell)}),\boldsymbol{x}^{(\ell-1)}\}} f(x) \tag{3.2}$$

which falls into the category of the so-called *multistep* methods [24, p. 65]. This LOCG, simply using a larger searching space than the line-search (3.1), always yields a better approximation $\boldsymbol{x}^{(\ell+1)}$ than the one by SD. However, this is not the key advantage that makes LOCG a much better choice than SD. The key advantage is its often overall fast convergence towards the optimum. Having said that, we must point out that the sub-minimization problem (3.2) could be much harder to solve than the line-search (3.1) for some nonlinear objective functions $f(x)$. When that is the case, LOCG may not be very appealing.

Recently, we have seen a surge in applying LOCG and its variants (including block, preconditioning, and extended subspace versions [14]) to solve large scale eigenvalue problems that admit certain optimization principles [1–3,13,16,20,25]. A major reason for this surge is due to not only in general much preferable speeds of convergence of LOCG methods but also in large part the easiness in solving related subspace-search problems like (3.2). However, precise (or accurate) estimates on the speeds of convergence of LOCG methods are still lacking in the literature. An intuitive argument

Algorithm 3 LOGMRES(k)

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

```
1: $r_0^{(1)} = b - Ax_0^{(1)}$, $x_0^{(-1)} = 0$;
2: For $\ell = 1, 2, \dots,$ Do:
3:   If $\|r_0^{(\ell)}\|_2 \leq \mathtt{tol} \times (\|A\|_2 \|x_0^{(\ell)}\|_2 + \|b\|_2)$ Then
4:     BREAK;
5:   else
6:     compute
         $x_k^{(\ell)} = \arg \min_{x \in \mathrm{span}\{x_0^{(\ell)}\} + \mathcal{K}_k(A, r_0^{(\ell)}) + \mathrm{span}\{x_0^{(\ell-1)}\}} \|b - Ax\|_2;$   (3.3)
7:     $x_0^{(\ell+1)} = x_k^{(\ell)}$;
8:     $r_0^{(\ell+1)} = b - Ax_0^{(\ell+1)}$;
9:   End If
10: End For
11: return $x_0^{(\ell)}$ as the computed solution to $Ax = b$.
```

for understanding the behavior is that the very previous approximation $\boldsymbol{x}^{(\ell-1)}$ brings sufficient history information before $\boldsymbol{x}^{(\ell)}$ into the current search, usually enough to make up the lost of previous search spaces.

In view of our discussion so far, it is natural for us to propose modifications to the restarted GMRES—Algorithm 2 by incorporating $x_0^{(\ell-1)}$ into Line 6. As a result, we have the locally optimal restarted generalized minimal residual method (LOGMRES) as given by Algorithm 3.

Remark 2 A couple of remarks are in order.

1. Setting $x_0^{(-1)} = 0$ at Line 1 is only for the convenience of presenting Algorithm 3 at Line 6 to cover all ℓ.
2. Most heavy work is at Line 6. We will discuss how to implement it in Sect. 4. As ℓ increases, $x_0^{(\ell-1)}$ and $x_0^{(\ell)}$ becomes increasingly linearly dependent. For a robust implementation, in Sect. 4 we will replace $x_0^{(\ell-1)}$ by some combination of $x_0^{(\ell-1)}$ and $x_0^{(\ell)}$ for their "difference".

There are variants to (3.2), too. The so-called *heavy ball* method [24, p. 65] is one:

$$\boldsymbol{x}^{(\ell+1)} = \arg \min_{s,t} f(\boldsymbol{x}^{(\ell)} + t\nabla f(\boldsymbol{x}^{(\ell)}) + s(\boldsymbol{x}^{(\ell)} - \boldsymbol{x}^{(\ell-1)})), \tag{3.4}$$

drawing its name from the motion of a "heavy ball" in a potential field under the force of friction. As a variant to Algorithm 3, and also as a direct application of the heavy ball method (3.4), we obtain the so-called *heavy ball* GMRES (HBGMRES) as in Algorithm 4 which is only slightly different from Algorithm 3.

LOGMRES and HBGMRES resemble the locally optimal block preconditioned extended conjugate gradient method (LOBPECG) for large scale eigenvalue problems [14,20], evolving from LOBPCG of Knyazev [13], the inverse free preconditioned

Algorithm 4 HBGMRES(k)

In Algorithm 3, replace its Line 6 by

$$x_k^{(\ell)} = x_0^{(\ell)} + \arg \min_{z \in \mathcal{K}_k(A, r_0^{(\ell)}) + \mathrm{span}\{x_0^{(\ell)} - x_0^{(\ell-1)}\}} \|b - A(x_0^{(\ell)} + z)\|_2. \tag{3.5}$$

Krylov subspace method of Golub and Ye [10] and Quillen and Ye [25]. For a short survey of such methods for eigenvalue problems, the reader is referred to [14], where a systematic naming convention is proposed. These locally optimal type methods are very easy to implement and very effective in computing a few largest and smallest eigenpairs with suitable preconditoners and have gained more and more usage in computational quantum chemistry and physics. For example, the MATLAB package KSSOLV [32] implemented LOBPCG to iteratively solve the nonlinear eigenvalue problem from discretizing the Kohn–Sham equations, and recently extensions of LOBPCG were proposed in [1–3] to solve large scale linear response eigenvalue problems and in [16] for the hyperbolic quadratic eigenvalue problem. Despite their numerical success, their precise convergence analysis is very difficult to do and lacking. The best one can do now is to view them as enhanced versions of the so-called *extended steepest descent method* (ESD), and use the analysis in [10] for ESD as a fallback [14]. Unfortunately, the resulting bound, though often sharp for ESD, usually underestimates the true convergence speed. Turning to LOGMRES and HBGMRES, we see that in each cycle both seek their respective optimal solutions within subspaces that contain $\mathcal{K}_k(A, r_0^{(\ell)})$, and thus

$$\|r_k^{(\ell)}\|_2 \le \min_{p_k(0)=1} \|p_k(A) r_0^{(\ell)}\|_2, \tag{3.6}$$

where the minimization is taken over all polynomials p_k of degree k or less such that $p_k(0) = 1$. The right-hand side of (3.6) is simply the residual norm for the k-step GMRES with the initial guess $x_0^{(\ell)}$. What it means is that each cycle of LOGMRES or HBGMRES produces better than or as good as a k-step GMRES solution. Various bounds on $\|r_k^{(\ell)}\|_2$ can be established by using known ones for GMRES (see, e.g., [5,6,9,11,15,17,28] and references therein). But such bounds are, however, far from satisfactory since the resulting bounds, similarly to the eigenvalue problem case above, don't reflect the true rate of $\|r_k^{(\ell)}\|_2$ going to zero as overwhelming numerical evidences suggest. More research is needed to unfold the mystery.

4 Implementation

As we commented previously, most work of LOGMRES given by Algorithm 3 is at Line 6:

$$x_k^{(\ell)} = \arg \min_{x \in \mathrm{span}\{x_0^{(\ell)}\} + \mathcal{K}_k(A, r_0^{(\ell)}) + \mathrm{span}\{x_0^{(\ell-1)}\}} \|b - Ax\|_2. \tag{3.3}$$

Correspondingly, for HBGMRES which differs from Algorithm 3 only in line 6, it is

$$x_k^{(\ell)} = x_0^{(\ell)} + \arg \min_{z \in \mathcal{K}_k(A, r_0^{(\ell)}) + \mathrm{span}\{x_0^{(\ell)} - x_0^{(\ell-1)}\}} \|b - A(x_0^{(\ell)} + z)\|_2. \tag{3.5}$$

In this section we explain how to implement them.

We mentioned before that in the actual implementation, $x_0^{(\ell-1)}$ in (3.3) should be replaced by some combination of $x_0^{(\ell-1)}$ and $x_0^{(\ell)}$ for their "difference". Let d be either $x_0^{(\ell-1)}$ or the "difference" to be given later. Also to simplify cluttered notation, we drop the superscripts "(ℓ)" and "$(\ell-1)$" and consider, instead of (3.3) and (3.5),

$$x_{\mathrm{new}} = \arg \min_{x \in \mathrm{span}\{x_0\} + \mathcal{K}_k(A, r_0) + \mathrm{span}\{d\}} \|b - Ax\|_2, \tag{4.1}$$

$$x_{\mathrm{new}} = x_0 + \arg \min_{z \in \mathcal{K}_k(A, r_0) + \mathrm{span}\{d\}} \|b - A(x_0^{(\ell)} + z)\|_2. \tag{4.2}$$

4.1 Solve (4.1)

Consider the generic case[2]:

$$\dim \mathcal{K}_{k+1}(A, r_0) = k + 1. \tag{2.1}$$

Using the Arnoldi process (Lines 3–13 of Algorithm 1 with input x_0), we have V_k, H_k, and $\check{H}_k$ satisfying (2.3) and (2.4):

$$AV_k = V_k H_k + v_{k+1} h_{k+1\,k} e_k^{\mathrm{T}} =: V_{k+1} \check{H}_k. \tag{2.3}$$

Now we orthogonalize d against V_k to define

$$\tilde{d} = V_k^{\mathrm{H}} d, \; g = d - V_k \tilde{d}. \tag{4.3}$$

There are two cases: $g = 0$ or not. Such cases are usually distinguished by testing"if $\|g\|_2 \le \mathtt{tol} \times \|d\|_2$", similarly to what we said in item 2 of Remark 1. This comment applies to all the situations below in determining whether a vector should be considered a zero vector due to orthogonalization.

In the case $g \neq 0$, we define

$$\tilde{v}_{k+1} = g/\|g\|_2, \; \widetilde{V}_{k+1} = [V_k, \tilde{v}_{k+1}], \tag{4.4}$$

and orthogonalize $A\tilde{v}_{k+1}$ against V_{k+1} to define

$$a = A\tilde{v}_{k+1}, \; \hat{a} = V_{k+1}^{\mathrm{H}} a, \; h = a - V_{k+1}\hat{a} \tag{4.5}$$

[2] For the non-generic case, GMRES—Algorithm 1 will yield the exact solution.

which again spawns out two more cases: $h = 0$ or not. In what follows, we deal with these cases separately

4.1.1 Case $g \neq 0$ and $h \neq 0$.

This is the most generic and common case. In this case, we have (4.4), as well as (4.5). Define

$$\hat{v}_{k+2} = h/\|h\|_2, \quad \widehat{V}_{k+2} = [V_{k+1}, \hat{v}_{k+2}] \in \mathbb{C}^{n\times(k+1)}. \tag{4.6}$$

Thus

$$\begin{aligned} A\widetilde{V}_{k+1} &= [AV_k, A\tilde{v}_{k+1}] \\ &= [V_{k+1}\check{H}_k, V_{k+1}\hat{a} + \|h\|_2\hat{v}_{k+2}] \\ &= \widehat{V}_{k+2}\begin{bmatrix} \check{H}_k & \hat{a} \\ 0 & \|h\|_2 \end{bmatrix} \\ &=: \widehat{V}_{k+2}\widehat{H}_{k+1}. \end{aligned} \tag{4.7}$$

We caution the reader the differences in meaning among V_{k+1} in (2.2), $\widetilde{V}_{k+1}$ in (4.4), and $\widehat{V}_{k+2}$ in (4.6). Now any $x \in \operatorname{span}\{x_0\} + \mathcal{K}_k(A, r_0) + \operatorname{span}\{d\}$ can be expressed by

$$x = \alpha x_0 + \widetilde{V}_{k+1}y, \quad \alpha \in \mathbb{C}, \quad y \in \mathbb{C}^{k+1}, \tag{4.8}$$

and vice versa. For such x, we have

$$\begin{aligned} Ax - b &= \alpha Ax_0 + A\widetilde{V}_{k+1}y - b \\ &= \alpha(Ax_0 - b) + A\widetilde{V}_{k+1}y - (1-\alpha)b \\ &= -\alpha r_0 + \widehat{V}_{k+2}\widehat{H}_{k+1}y - (1-\alpha)b. \end{aligned} \tag{4.9}$$

Orthogonalize b against $\widehat{V}_{k+2}$ to define

$$\hat{b} = \widehat{V}_{k+2}^{\mathrm{H}}b, \quad \hat{q} = b - \widehat{V}_{k+2}\hat{b}. \tag{4.10}$$

There are two subcases:

1. *Subcase $\hat{q} \neq 0$.* In this case, define $q = \hat{q}/\|\hat{q}\|_2$ and $\widehat{V}_{k+3} = [\widehat{V}_{k+2}, q]$. We have by (4.9) and (4.10)

$$\begin{aligned} Ax - b &= -\alpha\|r_0\|_2\widehat{V}_{k+3}e_1 + \widehat{V}_{k+2}\widehat{H}_{k+1}y - (1-\alpha)\widehat{V}_{k+3}\begin{bmatrix} \hat{b} \\ \|\hat{q}\|_2 \end{bmatrix} \\ &= \widehat{V}_{k+3}\left\{\begin{bmatrix} \widehat{H}_{k+1} & \hat{b} - \|r_0\|_2e_1 \\ 0 & \|\hat{q}\|_2 \end{bmatrix}\begin{bmatrix} y \\ \alpha \end{bmatrix} - \begin{bmatrix} \hat{b} \\ \|\hat{q}\|_2 \end{bmatrix}\right\}. \end{aligned} \tag{4.11}$$

Therefore the minimization in (4.1) is turned into

$$\min_{\alpha,\,y}\left\|\begin{bmatrix} \widehat{H}_{k+1} & \hat{b} - \|r_0\|_2e_1 \\ 0 & \|\hat{q}\|_2 \end{bmatrix}\begin{bmatrix} y \\ \alpha \end{bmatrix} - \begin{bmatrix} \hat{b} \\ \|\hat{q}\|_2 \end{bmatrix}\right\|_2 \tag{4.12}$$

which is a least squares problem whose coefficient matrix is in $\mathbb{C}^{(k+3)\times(k+2)}$ and upper-Hessenberg and thus can be solved in the same way as we usually do for Line 14 of Algorithm 1 [11]. Let $\alpha_{\rm opt}$ and $y_{\rm opt}$ be the optimal arguments. Then

$$x_{\rm new} = \alpha_{\rm opt}x_0 + \widetilde{V}_{k+1}y_{\rm opt}, \tag{4.13}$$

and the difference

$$x_{\rm new} - \alpha_{\rm opt}x_0 = \widetilde{V}_{k+1}y_{\rm opt} \tag{4.14}$$

can be used for the d in (4.1) for the next iterative cycle.

2. *Subcase* $\hat{q} = 0$. In this case, $b = \widehat{V}_{k+2}\hat{b}$. We have

$$\begin{aligned} Ax - b &= -\alpha\|r_0\|_2\widehat{V}_{k+2}e_1 + \widehat{V}_{k+2}\widehat{H}_{k+1}y - (1-\alpha)\widehat{V}_{k+2}\hat{b} \\ &= \widehat{V}_{k+2}\left\{\left[\widehat{H}_{k+1}\ \hat{b} - \|r_0\|_2e_1\right]\begin{bmatrix} y \\ \alpha \end{bmatrix} - \hat{b}\right\}. \end{aligned} \tag{4.15}$$

Therefore the minimization in (4.1) is turned into

$$\min_{\alpha,\, y}\left\|\left[\widehat{H}_{k+1}\ \hat{b} - \|r_0\|_2e_1\right]\begin{bmatrix} y \\ \alpha \end{bmatrix} - \hat{b}\right\|_2. \tag{4.16}$$

Note $B := \left[\widehat{H}_{k+1}\ \hat{b} - \|r_0\|_2e_1\right] \in \mathbb{C}^{(k+2)\times(k+2)}$. Let $\alpha_{\rm opt}$ and $y_{\rm opt}$ be the optimal arguments in (4.16) to give $x_{\rm new}$ in (4.13) and new difference in (4.14). It is not clear if B is nonsingular or not. If it is, $x_{\rm new}$ is the exact solution to $Ax = b$.

4.1.2 *Case* $g \neq 0$ *and* $h = 0$.

In this case, we have (4.4) and $a = A\tilde{v}_{k+1} = V_{k+1}\hat{a}$ by (4.5). Thus

$$\begin{aligned} A\widetilde{V}_{k+1} &= [AV_k, A\tilde{v}_{k+1}] \\ &= [V_{k+1}\check{H}_k, V_{k+1}\hat{a}] \\ &= V_{k+1}\left[\check{H}_k\ \hat{a}\right] =: V_{k+1}\widehat{H}_{k+1}, \end{aligned} \tag{4.17}$$

where $\widehat{H}_{k+1} \in \mathbb{C}^{(k+1)\times(k+1)}$. Any $x \in \operatorname{span}\{x_0\} + \mathcal{K}_k(A, r_0) + \operatorname{span}\{d\}$ still can be expressed by (4.8) and vice versa. For such x, we have

$$\begin{aligned} Ax - b &= \alpha Ax_0 + A\widetilde{V}_{k+1}y - b \\ &= \alpha(Ax_0 - b) + A\widetilde{V}_{k+1}y - (1-\alpha)b \\ &= -\alpha r_0 + V_{k+1}\widehat{H}_{k+1}y - (1-\alpha)b. \end{aligned} \tag{4.18}$$

Orthogonalize b against V_{k+1} to define

$$\hat{b} = V_{k+1}^{\rm H}b, \quad \hat{q} = b - V_{k+1}\hat{b}. \tag{4.19}$$

There are two subcases:

1. *Subcase* $\hat{q} \neq 0$. In this case, define $q = \hat{q}/\|\hat{q}\|_2$ and $\widehat{V}_{k+2} = [V_{k+1}, q]$. We have by (4.18) and (4.19)

$$\begin{aligned} Ax - b &= -\alpha\|r_0\|_2 \widehat{V}_{k+2} e_1 + V_{k+1}\widehat{H}_{k+1} y - (1-\alpha)\widehat{V}_{k+2} \begin{bmatrix} \hat{b} \\ \|\hat{q}\|_2 \end{bmatrix} \\ &= \widehat{V}_{k+2} \left\{ \begin{bmatrix} \widehat{H}_{k+1} & \hat{b} - \|r_0\|_2 e_1 \\ 0 & \|\hat{q}\|_2 \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} - \begin{bmatrix} \hat{b} \\ \|\hat{q}\|_2 \end{bmatrix} \right\}. \end{aligned} \tag{4.20}$$

Therefore the minimization in (4.1) is turned into

$$\min_{\alpha,\, y} \left\| \begin{bmatrix} \widehat{H}_{k+1} & \hat{b} - \|r_0\|_2 e_1 \\ 0 & \|\hat{q}\|_2 \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} - \begin{bmatrix} \hat{b} \\ \|\hat{q}\|_2 \end{bmatrix} \right\|_2. \tag{4.21}$$

Note that

$$B := \begin{bmatrix} \widehat{H}_{k+1} & \hat{b} - \|r_0\|_2 e_1 \\ 0 & \|\hat{q}\|_2 \end{bmatrix} \in \mathbb{C}^{(k+2)\times(k+2)}.$$

Let α_{opt} and y_{opt} be the optimal arguments in (4.21) to give x_{new} in (4.13) and new difference in (4.14). It is not clear if B is nonsingular or not. If it is, x_{new} is the exact solution to $Ax = b$.

2. *Subcase* $\hat{q} = 0$. In this case, $b = V_{k+1}\hat{b}$. We have

$$\begin{aligned} Ax - b &= -\alpha\|r_0\|_2 V_{k+1} e_1 + V_{k+1}\widehat{H}_{k+1} y - (1-\alpha) V_{k+1}\hat{b} \\ &= V_{k+1} \left\{ \begin{bmatrix} \widehat{H}_{k+1} & \hat{b} - \|r_0\|_2 e_1 \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} - \hat{b} \right\}. \end{aligned} \tag{4.22}$$

Therefore the minimization in (4.1) is turned into

$$\min_{\alpha,\, y} \left\| \begin{bmatrix} \widehat{H}_{k+1} & \hat{b} - \|r_0\|_2 e_1 \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} - \hat{b} \right\|_2. \tag{4.23}$$

Note that $B := \begin{bmatrix} \widehat{H}_{k+1} & \hat{b} - \|r_0\|_2 e_1 \end{bmatrix} \in \mathbb{C}^{(k+1)\times(k+2)}$. Let α_{opt} and y_{opt} be the optimal arguments in (4.23) to give x_{new} in (4.13) and new difference in (4.14). It is not clear if B has full row rank or not. If it is, x_{new} is the exact solution to $Ax = b$.

4.1.3 Case $g = 0$.

In this case, $d \in \mathcal{K}_k(A, r_0)$ and thus any

$$x \in \operatorname{span}\{x_0\} + \mathcal{K}_k(A, r_0) + \operatorname{span}\{d\} = \operatorname{span}\{x_0\} + \mathcal{K}_k(A, r_0)$$

can be expressed by $x = \alpha x_0 + V_k y$ for some $\alpha \in \mathbb{C}$ and $y \in \mathbb{C}^k$. We have

$$\begin{aligned} Ax - b &= \alpha A x_0 + A V_k y - b \\ &= \alpha (A x_0 - b) + A V_k y - (1 - \alpha) b \\ &= -\alpha r_0 + V_{k+1} \check{H}_k y - (1 - \alpha) b. \end{aligned} \tag{4.24}$$

Orthogonalize b against V_{k+1} to define

$$\hat{b} = V_{k+1}^{\mathrm{H}} b, \ \hat{q} = b - V_{k+1} \hat{b}. \tag{4.25}$$

There are two subcases:

1. *Subcase* $\hat{q} \neq 0$. In this case, define $q = \hat{q} / \|\hat{q}\|_2$ and $\widehat{V}_{k+2} = [V_{k+1}, q]$. We have

$$\begin{aligned} Ax - b &= -\alpha \|r_0\|_2 \widehat{V}_{k+2} e_1 + V_{k+1} \check{H}_k y - (1 - \alpha) \widehat{V}_{k+2} \begin{bmatrix} \hat{b} \\ \|\hat{q}\|_2 \end{bmatrix} \\ &= \widehat{V}_{k+2} \left\{ \begin{bmatrix} \check{H}_k & \hat{b} - \|r_0\|_2 e_1 \\ 0 & \|\hat{q}\|_2 \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} - \begin{bmatrix} \hat{b} \\ \|\hat{q}\|_2 \end{bmatrix} \right\}. \end{aligned} \tag{4.26}$$

   Therefore the minimization in (4.1) is turned into

$$\min_{\alpha,\, y} \left\| \begin{bmatrix} \check{H}_k & \hat{b} - \|r_0\|_2 e_1 \\ 0 & \|\hat{q}\|_2 \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} - \begin{bmatrix} \hat{b} \\ \|\hat{q}\|_2 \end{bmatrix} \right\|_2 \tag{4.27}$$

   which is a least squares problem whose coefficient matrix is $(k+2) \times (k+1)$ and upper-Hessenberg and thus can be solved in the same way as we usually do for Line 14 of Algorithm 1 [11]. Let α_{opt} and y_{opt} be the optimal arguments in (4.27). Then

$$x_{\mathrm{new}} = \alpha_{\mathrm{opt}} x_0 + V_k y_{\mathrm{opt}}, \tag{4.28}$$

   and the difference

$$x_{\mathrm{new}} - \alpha_{\mathrm{opt}} x_0 = V_k y_{\mathrm{opt}} \tag{4.29}$$

   can be used for the d in (4.1) for the next iterative cycle.
2. *Subcase* $\hat{q} = 0$. In this case, $b = V_{k+1} \hat{b}$. We have

$$\begin{aligned} Ax - b &= -\alpha \|r_0\|_2 V_{k+1} e_1 + V_{k+1} \check{H}_k y - (1 - \alpha) V_{k+1} \hat{b} \\ &= V_{k+1} \left\{ \begin{bmatrix} \check{H}_k & \hat{b} - \|r_0\|_2 e_1 \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} - \hat{b} \right\}. \end{aligned} \tag{4.30}$$

   Therefore the minimization in (4.1) is turned into

$$\min_{\alpha,\, y} \left\| \begin{bmatrix} \check{H}_k & \hat{b} - \|r_0\|_2 e_1 \end{bmatrix} \begin{bmatrix} y \\ \alpha \end{bmatrix} - \begin{bmatrix} \hat{b} \\ \|\hat{q}\|_2 \end{bmatrix} \right\|_2. \tag{4.31}$$

Note $B := \big[\check{H}_k\ \hat{b} - \|r_0\|_2 e_1\big] \in \mathbb{C}^{(k+1)\times(k+1)}$. Let $\alpha_{\rm opt}$ and $y_{\rm opt}$ be the optimal arguments in (4.31) to give $x_{\rm new}$ in (4.28) and new difference in (4.29). It is not clear if B is nonsingular or not. If it is, $x_{\rm new}$ is the exact solution to $Ax = b$.

It is clear that solving (4.1) is more costly than the last step—Line 14—of the k-step GMRES as in Algorithm 1. Most significantly added cost comes from three extra orthogonalizations in (4.3), (4.5), and (4.10) and one extra matrix-vector multiplication in (4.5). To compensate the added cost, in our later numerical tests we compare REGMRES($k+1$) against LOGMRES(k). This will roughly align per cycle cost for matrices whose densities are not so cheap and for modest k.

4.2 Solve (4.2)

Now we consider implementing the heavy ball restarted GMRES (HBGMRES) which differs from Algorithm 3 only in line 6: replace it by the one in (3.5). As before, we still have (2.3), and all from (4.3)–(4.5).

4.2.1 Case $g \neq 0$ and $h \neq 0$

We have (4.6) and (4.7). Different from (4.8), any $x - x_0 \in \mathcal{K}_k(A, r_0) + \operatorname{span}\{d\}$ can be expressed by

$$x - x_0 = \widetilde{V}_{k+1} y, \quad y \in \mathbb{C}^{k+1}, \tag{4.32}$$

and vice versa. Then we have

$$\begin{aligned} Ax - b &= Ax_0 + A\widetilde{V}_{k+1} y - b \\ &= (Ax_0 - b) + A\widetilde{V}_{k+1} y \\ &= -r_0 + \widehat{V}_{k+2}\widehat{H}_{k+1} y \\ &= -\|r_0\|_2 \widehat{V}_{k+2} e_1 + \widehat{V}_{k+2}\widehat{H}_{k+1} y \\ &= -\widehat{V}_{k+2}\Big[\widehat{H}_{k+1} y - \|r_0\|_2 e_1\Big]. \end{aligned} \tag{4.33}$$

Therefore the minimization in (4.2) is turned into

$$\min_y \Big\|\widehat{H}_{k+1} y - \|r_0\|_2 e_1\Big\|_2 \tag{4.34}$$

which is a least squares problem whose coefficient matrix is $(k+2) \times (k+1)$ and upper-Hessenberg. Let $y_{\rm opt}$ be the optimal argument in (4.34). Then

$$x_{\rm new} = x_0 + \widetilde{V}_{k+1} y_{\rm opt}, \tag{4.35}$$

and the difference

$$x_{\rm new} - x_0 = \widetilde{V}_{k+1} y_{\rm opt} \tag{4.36}$$

can be used for the d in (4.2) for the next iterative cycle.

4.2.2 Case $g \neq 0$ and $h = 0$

In this case, we have $a = A\tilde{v}_{k+1} = V_{k+1}\hat{a}$ in (4.5), and (4.17). Any $x - x_0 \in \mathcal{K}_k(A, r_0) + \text{span}\{d\}$ can be expressed by (4.32) and vice versa. For such x, we have

$$\begin{aligned} Ax - b &= Ax_0 + A\widetilde{V}_{k+1}y - b \\ &= (Ax_0 - b) + A\widetilde{V}_{k+1}y \\ &= -r_0 + V_{k+1}\widehat{H}_{k+1}y. \end{aligned} \tag{4.37}$$

Therefore the minimization in (4.2) is turned into

$$\min_y \left\| \widehat{H}_{k+1}y - \|r_0\|_2 e_1 \right\|_2. \tag{4.38}$$

Note that $\widehat{H}_{k+1}$ is $(k+1) \times (k+1)$. It is not clear if $\widehat{H}_{k+1}$ is nonsingular. If it is, the exact solution to $Ax = b$ is computed immediately, and if it is not, we let y_{opt} be the optimal argument in (4.38) and we will have (4.35) and (4.36).

4.2.3 Case $g = 0$

In this case, $d \in \mathcal{K}_k(A, r_0)$ and thus any

$$x - x_0 \in \mathcal{K}_k(A, r_0) + \text{span}\{d\} = \mathcal{K}_k(A, r_0)$$

can be expressed by $x = x_0 + V_k y$ for some $y \in \mathbb{C}^k$. We have

$$\begin{aligned} Ax - b &= Ax_0 + AV_k y - b \\ &= (Ax_0 - b) + AV_k y \\ &= -r_0 + V_{k+1}\check{H}_k y. \end{aligned} \tag{4.39}$$

Therefore the minimization in (4.2) is turned into the usual k-step GMRES.

In comparison with the last step—Line 14 of the k-step GMRES, solving (4.2) costs two more extra orthogonalizations in (4.3) and (4.5), but it costs less than solving (4.1) since it does not require the orthogonalization (4.10).

5 Numerical examples

In Sect. 1, we mentioned Morgan's approach [21,22] and the general hybrid GMRES paradigm [7,8,19,23,26,30] that were designed to improve the performances of GMRES and REGMRES. All of the resulting methods, except the one in [23], need some sort of eigen-estimations which make them somewhat more complicated than GMRES and REGMRES. Any subtle difference that almost surely exists from implementation to implementation could and can yield significant performance differences. That will make fair comparison a difficult task. Therefore in selecting known methods

Table 1 Flops for GMRES variants

GMRES of k-steps	$(k+1)\,(\textsc{mv}) + 2k^2n + 4k^2$
Per cycle of REGMRES(k)	$(k+1)\,(\textsc{mv}) + 2k^2n + 4k^2$
Per cycle of LOGMRES(k)	$(k+2)\,(\textsc{mv}) + 2(k+3)^2n + 4(k+2)^2$
Per cycle of HBGMRES(k)	$(k+2)\,(\textsc{mv}) + 2(k+2)^2n + 4(k+1)^2$
Hybrid-GMRES(k) [23]	$(k+1)\,(\textsc{mv}) + 2k^2n + 4k^2$ (initial cycle) $k\,(\textsc{mv}) + 2kn$ (subsequent Richardson cycle)

of the minimizing residual type for fair comparisons, we adopt two criteria: (1) no need to do any spectral estimation, and (2) as easy to implement as REGMRES. This leaves only the hybrid GMRES in [23]. Therefore, we mainly compare the proposed methods: LOGMRES and HBGMRES, with the hybrid GMRES. Nevertheless, in experiment 2, we additionally compare with Morgan's GMRES-E in [21] as one of the most successful improvement of REGMRES.

Table 1 lists the numbers of flops for one and its variants, where (MV) is the number of flops by one matrix-vector multiplication with A. 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 three major actions within each inner cycle: matrix-vector multiplications, orthogonalizations, and solutions of the reduced least squares problems. However, the hybrid GMRES [23] in its subsequent cycles does not have the latter two actions. Instead, it simply applies the GMRES polynomial obtained in the initial GMRES run to residual vectors. For notational convenience, we use hybrid-GMRES(k) to represent the method in [23] that initially uses the k-step GMRES (the initial cycle) and then cyclically performs Richardson iterations (the Richardson cycles). Hence, in terms of flops per cycle, this hybrid GMRES is most favorable, especially when (MV) is very cheap.

In what follows, we will present two experiments to compare GMRES, REGMRES, LOGMRES, HBGMRES and two existing improvements on GMRES: the hydrid GMRES in [23] and the GMRES-E in [21]. Our accuracy measure is *Normalized Residual* (NRes)

$$\mathrm{NRes} = \frac{\|Ax-b\|_2}{\|A\|_1\|x\|_2 + \|b\|_2} \tag{5.1}$$

against the number of cycles for the last three methods in Table 1, where using $\|A\|_1$ is for its easiness in computation.

We also experimented two different reorthogonalization strategies in the Arnoldi processes for computing orthonormal bases of Krylov subspaces and in the additional orthogonalizations in LOGMRES and HBGMRES. They are *selective reorthogonalization* and *always reorthogonalization*. In the selective reorthogonalization case, we use MGS with a guarded-step as follows. Consider the **for**-loop in (2.8) and let $\alpha_0 = \|f\|_2$ right before entering the loop and let $\alpha_1 = \|f\|_2$ right after exiting the loop. We check "whether $\alpha_1 \le \texttt{tol\_orth} \times \alpha_0$", where `tol_orth` is a preset tolerance to detect any severity of cancellation in performing the **for**-loop. If the test is true, then we essentially repeat the **for**-loop one more time, i.e., immediately after the **for**-loop in (2.8), we add

```
for $j = 1, 2, \dots, i$ do
    $t = V^{\mathrm{H}}_{(:,j)} f$; $\check{H}_{(j,i)} = \check{H}_{(j,i)} + t$, $f = f - V_{(:,j)}\, t$;
end for
```

(5.2)

Otherwise the loop will not be repeated. For the numerical results we will report, we take `tol_orth` $= 10^{-2}$. With this selective reorthogonalization scheme, we recorded how many reorthogonalizations (5.2) were carried out during our tests, and we found that on average about one reorthogonalization (5.2) per cycle was performed in our tests. In the always reorthogonalization case, we essentially repeat Line 5 of Algorithm 1 twice, i.e., immediately after Line 5, we add

$$t = V^{\mathrm{H}}_{(:,1:i)} f, \;\; \check{H}_{(1:i,i)} = \check{H}_{(1:i,i)} + t, \;\; f = f - V_{(:,1:i)}\, t;$$

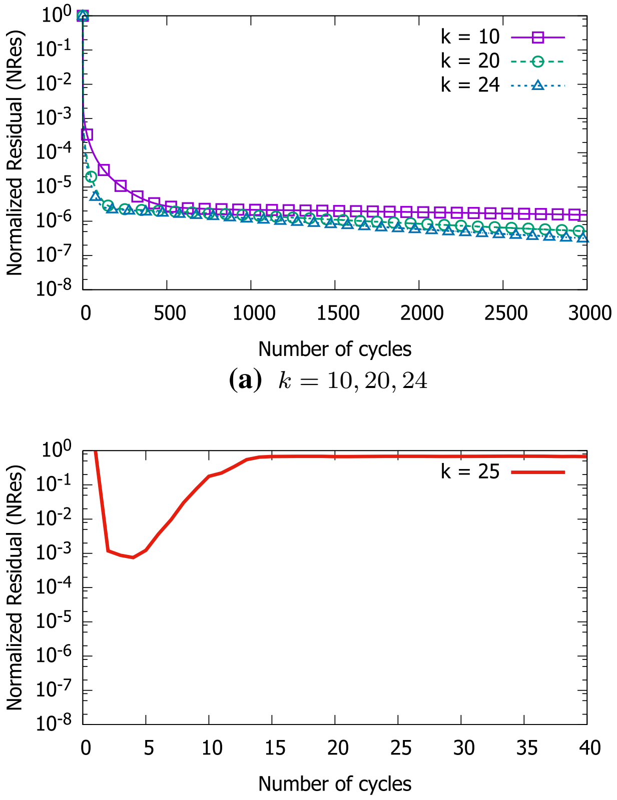
Experiment 1 Nachtigal, Reichel, and Trefethen [23, section 8] tested their hybrid-GMRES(k) and REGMRES(k), and two other methods, on three banded Toeplitz matrices, one artificial bi-diagonal and one artificial diagonal matrices. The conclusion is that hybrid-GMRES(k) (with appropriate k) usually wins over REGMRES(k) in terms of work needed to reduce residual norms to a certain level, even though it looks slower in terms of the number of cycles than REGMRES(k). We also ran tests on these matrices with random right-hand sides, and found that hybrid-GMRES(k) had advantages over LOGMRES(k) and HBGMRES(k) with the same k in terms of work needed. It turns out for these matrices, all methods are able to achieve respectable rates in residual norm reduction per cycle and to reduce the norm to a given level in comparable numbers of cycles (with, in fact, hybrid-GMRES(k) using the most cycles). This, combining with a very cheap (MV), makes hybrid-GMRES(k) the favorite. However, these matrices are very artificial, even for banded Toeplitz ones, and they do not seem to present much of a challenge to GMRES and its variants at all. Our next experiment are on real life matrices and tells a different story.

Experiment 2 All testing matrices in this experiment are taken from the University of Florida sparse matrix collection [4] with no particular preference in their selections. Each comes with their right-hand sides b. Table 2 lists 10 testing examples and their characteristics, where n is the size of the matrix, `nnz` is the number of nonzero entries, and sparsity is `nnz`$/n^2$. These are the ones on which we will report our numerical results in detail, although we have run our code on many more, including randomly generated right-hand sides b. These results are quite representative.

First, we tested how hybrid-GMRES(k) [23] behaves on these examples. We found that the method worked poorly. For each example, the method became unstable for $k = k_{\mathrm{c}}$, some critical integer to be given, or bigger in the sense that the normalized residual norm may decrease for just a few cycles and then increase to $O(1)$. The case when $k_{\mathrm{c}} = 1$ means that hybrid-GMRES(k) doesn't work at all. Unfortunately, for five out of the ten problems in Table 2 $k_{\mathrm{c}} = 1$. The other five problems for which $k_{\mathrm{c}} > 1$ are `1:cavity05`, `2:cavity10`, `4:comsol`, `8:raefsky1`, and `9:raefsky2` and their respective k_{c} are 25, 25, 5, 4, 29. Even for those k that hybrid-GMRES does seem to work, we witnessed very slow convergence. As an example, Fig. 2 shows the

Table 2 Testing matrices

ID	Matrix	n	nnz	Sparsity (%)	Application
1	`cavity05`	1182	32747	2.3439	Computational fluid dynamics
2	`cavity10`	2597	76367	1.1323	Computational fluid dynamics
3	`chipcool0`	20082	281150	0.0697	Model reduction problem
4	`comsol`	1500	97645	4.3398	Structural problem
5	`flowmeter5`	9669	67391	0.0721	Model reduction problem
6	`fpga_trans_02`	1220	7382	0.4960	Circuit simulation
7	`memplus`	17758	126150	0.0400	Circuit simulation
8	`raefsky1`	3242	294276	2.7998	Computational fluid dynamics
9	`raefsky2`	3242	294276	2.7998	Computational fluid dynamics
10	`wang3`	26064	177168	0.0261	Semiconductor device problem



(a) $k = 10, 20, 24$

(b) $k = 25$

Fig. 2 Normalized residual norm vs. cycle for `1:cavity05` by hybrid-GMRES(k) [23]. Hybrid-GMRES is unstable for $k = 25$ or larger and converges slowly for $k \leq 24$. **a** $k = 10, 20, 24$. **b** $k = 25$

convergence behaviors by hybrid-GMRES(k) on `1:cavity05` for $k = 10, 20, 24$ and 25. We see the instability at $k = 25$ and extremely slow convergence at other k. This slowness can be attributed to the ineffectiveness of the GMRES polynomials generated at the initial GMRES cycle for subsequent Richardson iterations.

In view of such poor performances of hybrid-GMRES(k) on these real life problems, in what follows, we focus on comparing GMRES, REGMRES, GMRES-E and our two variants: LOGMRES and HBGMRES. In our tests, besides NRes, we will also look into

– CPU times by our FORTRAN 90 implementations for reducing NRes below 10^{-12};
– NRes against the numbers of flops for all four methods. For this, we let the number of the Arnoldi steps in GMRES increase while fixing it in its variants within a cycle.

In order to be fair in comparing the algorithms, we use the following testing scenarios.

– We will run REGMRES($k+1$), LOGMRES(k), and HBGMRES(k) in order to best illustrate convergence against cycle indices. This is based on two considerations. One is to make sure that all approximate solutions at a cycle are computed from a subspace of dimension $k+2$: $\mathrm{span}\{x_0\}+\mathcal{K}_{k+1}(A, r_0)$ for REGMRES($k+1$), and $\mathrm{span}\{x_0\}+\mathcal{K}_k(A, r_0)+\mathrm{span}\{d\}$ for LOGMRES(k) and HBGMRES(k); the second consideration is to align the costs per cycle for all three algorithms. By Table 1, we find that the leading costs in flops for REGMRES, LOGMRES, and HBGMRES come from matrix-vector multiplications and orthogonalization, assuming that k is much smaller than n. For such k, matrix-vector multiplications either dominate or cost about as much as orthogonalizations if the number of nonzero elements of the matrix is large. In such a scenario, costs per cycle for REGMRES($k+1$), LOGMRES(k), and HBGMRES(k) are more or less the same.
– We will also run GMRES-E($k+1$, p) in order to be the same number of the matrix-vector multiplication in each cycle, where p is the number of the approximated eigenvectors additionally used in the Arnoldi step.

For each matrix there, we ran GMRES(31), GMRES-E(31, 3), LOGMRES(30), and HBGMRES(30) on the vector b that comes with the matrix with selective and always reorthogonalization in the Arnoldi process and in the additional orthogonalizations in LOGMRES and HBGMRES. Figure 2 is generated by our MATLAB implementation on a window 7 PC. For the following experiments, we also implemented the algorithms in FORTRAN90 and tested REGMRES(31), GMRES-E(31, 3), LOGMRES(30), and HBGMRES(30) on an LINUX machine with Intel Xeon CPU E5-2450 2.1 GHz, and 32 GB memory. We point out that what we will report below is only a subset of the more complete one in [12] because of space limitation.

Table 3 lists the numbers of cycles needed by the algorithms to reduce NRes to less or equal to 10^{-12}, except for those marked by "–" which mean that the maximum number 3000 of cycles is exceeded without achieving the goal. In the table, RE, DE, LO, HB stand for REGMRES, GMRES-E, LOGMRES, and HBGMRES, respectively. We can see, from the table, that GMRES-E improved convergence of REGMRES for several problems. However, the improvement dose not always succeed, e.g., `3:chipcool0`, `5:flowmeter5` and `7:memplus`. On the other hand, the table clearly demonstrates huge savings achieved by LOGMRES and HBGMRES over REGMRES for almost all problems.

Table 3 Number of iteration cycles

ID	Always reorth				Selective reorth			
	RE	DE	LO	HB	RE	DE	LO	HB
1	741	43	90	76	477	43	90	76
2	1961	127	139	171	1623	97	139	172
3	–	–	497	533	–	–	464	491
4	2931	26	81	261	–	27	83	287
5	–	–	1425	1409	–	–	1420	1400
6	155	104	41	38	155	104	41	38
7	83	80	39	38	83	80	39	38
8	204	14	38	38	218	14	38	38
9	188	33	107	140	188	33	114	135
10	26	14	23	23	26	14	23	23

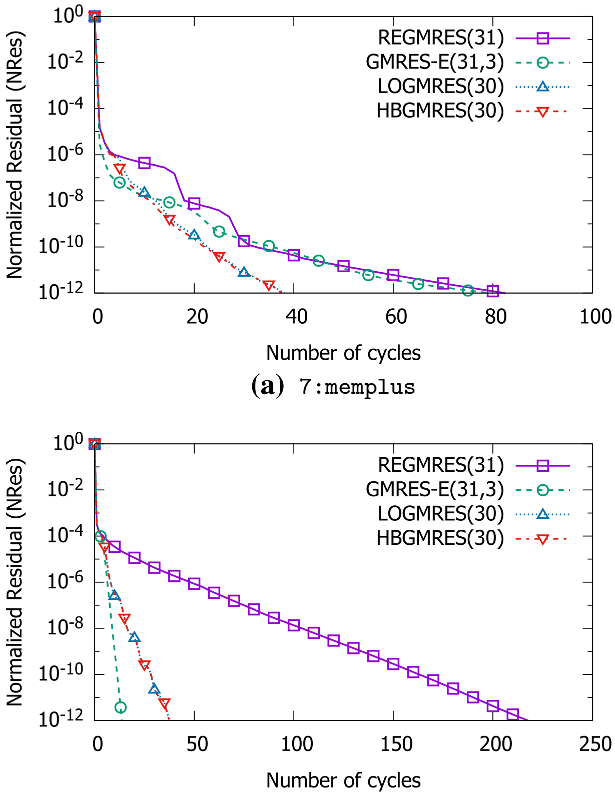
To get a better impression as to how each algorithm behaves, we plot Fig. 3 to show how NRes in the computed solutions varies against the cycle index. Although it, together with Fig. 1, cover only 4 out of the 10 examples, these plots are quite representative. See [12] for plots for the other 6 examples.

The CPU times in Table 4 are with selective reorthogonalization. For each method, we list CPU time per cycle as well as total time for reducing NRes to less than or equal to 10^{-12}. Firstly, we consider CPU times per cycle. From the table we see that GMRES-E required much larger CPU times per cycle than other three methods and other three methods showed very much the same CPU times per cycle, justifying our earlier decision on comparing REGMRES(31), LOGMRES(30), and HBGMRES(30). But, it is a whole different story for the total CPU times. We again see that REGMRES(31) stops prematurely for `3:chipcool0`, `4:comsol`, and `5:flowmeter5` because the maximum allowable number of cycles is reached before NRes becomes less or equal or equal to 10^{-12}. GMRES-E(31, 3) also reached maximum number of cycle for `3:chipcool0` and `5:flowmeter` before NRes $\leq 10^{-12}$, although GMRES-E(31, 3) improved convergence of REGMRES(31) for several problems. LOGMRES(30) and HBGMRES(30) could compute the solution such that NRes $\leq 10^{-12}$ within much smaller total computation time than REGMRES(31) for almost all problems. Overall, LOGMRES(30) and HBGMRES(30) consume about the same total CPU time, except for `4:comsol`.

We make the following observations from Figs. 1 and 3, and Tables 3 and 4.

- LOGMRES and HBGMRES are mostly comparable. We see that sometimes one is faster than the other while at other times the other is faster, and the differences in the numbers of cycles are mostly not very big. One exception is `4:comsol` for which LOGMRES use about one-fourth the number of cycles that HBGMRES takes.

  On the other hand, the coding of HBGMRES is much simpler than LOGMRES, as can be seen from Sect. 4.



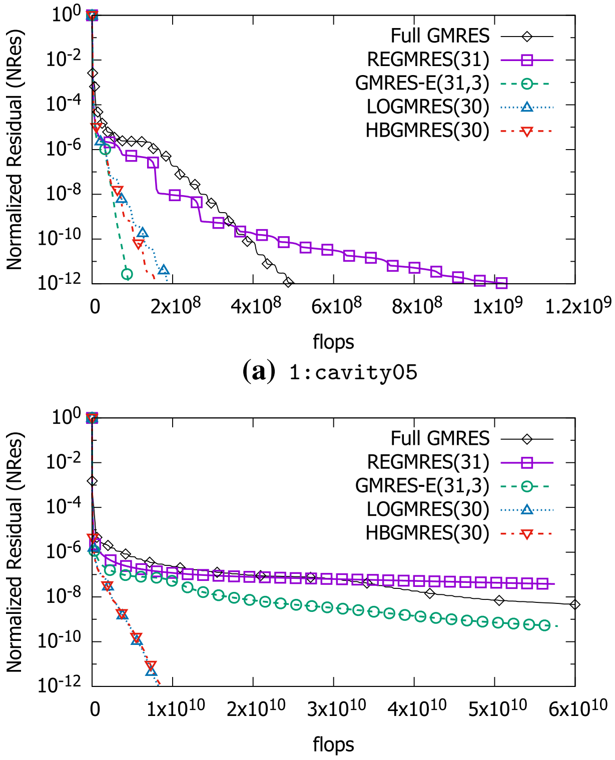
(a) `7:memplus`

(b) `8:raefsky1`

Fig. 3 NRes vs. cycle for `7:memplus` and `8:raefsky1` with selective reorthogonalization. See Fig. 1 for the results on `1:cavity05` and `3:chipcool0`

Table 4 CPU time

ID	RE		DE		LO		HB	
	Total	Cycle	Total	Cycle	Total	Cycle	Total	Cycle
1	1.7E+0	3.5E-3	2.4E-1	5.7E-3	3.3E-1	3.6E-3	2.7E-1	3.6E-3
2	1.3E+1	8.3E-3	1.1E+0	1.1E-2	1.2E+0	8.6E-3	1.5E+0	8.4E-3
3	–	3.5E-2	–	4.9E-2	1.8E+1	3.8E-2	1.8E+1	3.7E-2
4	–	1.0E-2	3.5E-1	1.3E-2	8.8E-1	1.1E-2	3.0E+0	1.1E-2
5	–	1.0E-2	–	1.7E-2	1.6E+1	1.1E-2	1.5E+1	1.1E-2
6	1.8E-1	1.1E-3	3.4E-1	3.3E-3	5.1E-2	1.2E-3	4.4E-2	1.2E-3
7	1.8E+0	2.2E-2	2.7E+0	3.4E-2	9.5E-1	2.4E-2	8.8E-1	2.3E-2
8	7.1E+0	3.3E-2	5.1E-1	3.7E-2	1.2E+0	3.3E-2	1.2E+0	3.3E-2
9	6.1E+0	3.3E-2	1.2E+0	3.7E-2	3.7E+0	3.3E-2	4.4E+0	3.3E-2
10	7.5E-1	2.9E-2	6.7E-1	4.8E-2	7.3E-1	3.2E-2	6.9E-1	3.0E-2



(a) `1:cavity05`

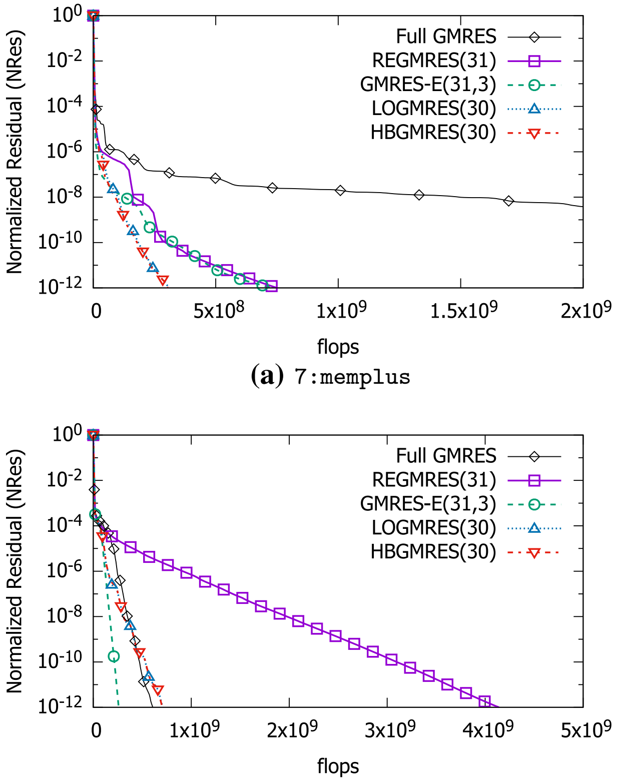
(b) `3:chipcool0`

Fig. 4 NRes vs. flops for `1:cavity05` and `3:chipcool0` with selective reorthogonalization

- GMRES-E is sometimes the fastest method, however for `3:chipcool0` and `5:flowmeter5`, it fails to converge with $k = 31$ and maximum cycle 3000. On the other hand, both LOGMRES and HBGMRES are overwhelmingly faster than REGMRES, except for `10:wang3` which seems to be an easier problem than the others for all methods. In fact, all of the methods perform well on it and are not that much different in performance.
- There is usually little differences between the number of iteration cycles for with always reorthogonalization and with selective reorthogonalization. This makes selective reorthogonalization a better choice considering cost.

In summary, these numerical results confirm our earlier argument: in the proposed methods, the very previous approximation $x^{(\ell-1)}$ recovers sufficient history information before $x^{(\ell)}$ that is simply lost in REGMRES.

The first and foremost reason for using the restarted GMRES instead of the usual GMRES is to prevent the dimension of the Krylov subspace within which an approximate solution is sought from getting too large in a computing environment where the available memory is limited. The secondary reason is to control orthogonalization cost in computing basis vectors in the Arnoldi process. However, the price one usually pays for doing so is the degradation of convergence speed, as we know. As this argument goes, one would expect that the restarted GMRES (and any of its variants) would be



(a) `7:memplus`

(b) `8:raefsky1`

Fig. 5 NRes vs. flops for `7:memplus`, and `8:raefsky1` with selective reorthogonalization

slower than GMRES in terms of flops consumed to obtain approximate solutions with comparable accuracy, assuming memory is not a concern. Next, we compare NRes against flops by GMRES, REGMRES(31), LOGMRES(30), and HBGMRES(30) with selective reorthogonalization.

What is surprising for us to read from Figs. 4 and 5 is that GMRES may lose out to LOGMRES and HBGMRES. To save space, we summarize what we found in our numerical tests, and the reader is referred to [12] for more detail. Among the 10 examples in Table 2, GMRES lose out on `1:cavity05`, `2:cavity10`, `3:chipcool0`, `6:fpga_trans_02`, `7:memplus`, and `10:wang3`. That is 6 out of 10. As for GMRES and REGMRES, GMRES does win in 8 out of all 10 matrices. In summary, fairly often both LOGMRES and HBGMRES can outperform GMRES which outperforms REGMRES in terms of reduction in NRes per flop. That is a significant finding.

6 Concluding remarks

There are many variations of REGMRES, including the hybrid GMRES method in [23] and those surveyed there and Morgan's deflated restarting REGMRES. All of them, except the one developed in [23], require estimating a few eigenvalues or rough regions that contains the spectrum of A. Such estimations are nontrivial and any slight difference in implementation may result in significant convergence differences or no convergence at all. Those methods, if implemented correctly, can be effective, but that is difficult to realize, while for GMRES and REGMRES, implementation is very much straightforward.

We seeked to develop numerical methods that can rival GMRES and REGMRES in preserving their most appealing feature—simplicity in implementation, and at the same time the ability to dramatically improve the convergence speed of REGMRES. For this purpose, we have presented two variants of the restarted GMRES. Essentially, they are combinations of the restarted GMRES with the locally optimal approach or the "heavy ball" approach, resulting naturally in names the *locally optimal GMRES* or LOGMRES for short and *the heavy ball GMRES* or HBGMRES for short. Although their creations are based more on intuitive thinking than any theoretical estimates, numerical evidences overwhelmingly favor LOGMRES and HBGMRES to the original restarted GMRES. Among the two, sometimes one performs better than the other while at other times the opposite is true, but usually performance differences aren't that dramatic, although there are exceptions.

It is quite surprising to us that in 6 out of 10 problems in Experiment 2, LOGMRES and HBGMRES outperform GMRES in the measure of residual norm reduction per flop. This finding further reenforces our motivational argument in creating the two methods to preserve the low-order Arnoldi process in the restarted GMRES and at the same time salvage the benefit of high-order polynomial approximation in GMRES. Despite their superior performances, we don't have any good theory to explain them beyond what we commented at the end of Sect. 3.

It is tempting to go beyond current LOGMRES and HBGMRES to include more previous differences $x^{(i)}$, not just the very previous one, into the search. This will add extra cost, but how much more gain will that bring? It would be an interesting question to look into in the future.

The two criteria that led us to chose the hybrid GMRES in [23] as the only known method to compare our methods against are: no need to do spectral information estimation and as easy to implement as REGMRES. It was reported in [23], and confirmed here, that the hybrid GMRES worked very well on the testing matrices there which are three banded Toeplitz matrices, one artificial bi-diagonal and one artificial diagonal matrices. But when applied to real life problems from [4], the method either diverges or converges very slowly.

We have been focusing on the plain version of GMRES, i.e., without incorporating any preconditioner. But it is not hard to notice that incorporating a preconditioner presents no difficulty to either LOGMRES or HBGMRES.

From the point of view that MINRES is simply GMRES applied to symmetric/Hermitian linear systems, except only cheaper through careful exploitations of the symmetry structure, we can easily give the *locally optimal MINRES* or LOMINRES for short and *the heavy ball MINRES* or HBMINRES for short. We omit the details.

Acknowledgments The authors are grateful to an anonymous referee for useful comments.

References

1. Bai, Z., Li, R.-C.: Minimization principle for linear response eigenvalue problem, I: theory. SIAM J. Matrix Anal. Appl. **33**(4), 1075–1100 (2012)
2. Bai, Z., Li, R.-C.: Minimization principles for the linear response eigenvalue problem II: computation. SIAM J. Matrix Anal. Appl. **34**(2), 392–416 (2013)
3. Bai, Z., Li, R.-C.: Minimization principles and computation for the generalized linear response eigenvalue problem. BIT Numer. Math. **54**(1), 31–54 (2014)
4. Davis, T., Hu, Y.: The University of Florida sparse matrix collection. ACM Trans. Math. Softw. **38**(1), 1:1–1:25 (2011)
5. Driscoll, Tobin A., Toh, Kim-Chuan, Trefethen, Lloyd N.: From potential theory to matrix iterations in six steps. SIAM Rev. **40**(3), 547–578 (1998)
6. Elman, H.C.: Iterative methods for large, sparse nonsymmetric systems of linear equations. Ph.D. thesis, Department of Computer Science, Yale University (1982)
7. Elman, H.C., Saad, Y., Saylor, P.E.: A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations. SIAM J. Sci. Stat. Comput. **7**(3), 840–855 (1986)
8. Elman, H.C., Streit, R.L.: Polynomial iteration for nonsymmetric indefinite linear systems. In: Hennart, J.-P. (ed.) Numerical Analysis. Lecture Notes in Mathematics, vol. 1230, pp. 103–117. Springer, Berlin Heidelberg (1986)
9. Ernst, Oliver G.: Residual-minimizing Krylov subspace methods for stabilized discretizations of convection–diffusion equations. SIAM J. Matrix Anal. Appl. **21**(4), 1079–1101 (2000)
10. Golub, G., Ye, Q.: An inverse free preconditioned Krylov subspace methods for symmetric eigenvalue problems. SIAM J. Sci. Comput. **24**, 312–334 (2002)
11. Greenbaum, A.: Iterative Methods for Solving Linear Systems. SIAM, Philadelphia (1997)
12. Imakura, A., Li, R.-C., Zhang, S.-L.: Locally optimal and heavy ball GMRES methods. Technical Report 2015-02, Department of Mathematics, University of Texas at Arlington. http://www.uta.edu/math/preprint/. Accessed Jan 2015
13. Knyazev, A.V.: Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method. SIAM J. Sci. Comput. **23**(2), 517–541 (2001)
14. Li, R.-C.: Rayleigh quotient based optimization methods for eigenvalue problems. In: Bai, Z., Gao, W., Su, Y (eds.) Matrix Functions and Matrix Equations, Series in Contemporary Applied Mathematics, vol. 19. Lecture summary for 2013 Gene Golub SIAM Summer School, 22 July to 2 August 2013, Fudan University, Shanghai, China. World Scientific, Singapore (2015)
15. Li, R.-C., Zhang, W.: The rate of convergence of GMRES on a tridiagonal Toeplitz linear system. Numer. Math. **112**, 267–293 (2009). (published online 19 December 2008)
16. Liang, X., Li, R.-C.: The hyperbolic quadratic eigenvalue problem. Technical Report 2014-01, Department of Mathematics, University of Texas at Arlington. http://www.uta.edu/math/preprint/. Accessed Jan 2014
17. Liesen, J., Strakoš, Z.: Convergence of GMRES for tridiagonal Toeplitz matrices. SIAM J. Matrix Anal. Appl. **26**(1), 233–251 (2004)
18. Liesen, J., Strakoš, Z.: Krylov Subspace Methods: Principles and Analysis. Oxford University Press, Oxford (2013)
19. Manteuffel, T.A.: Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration. Numer. Math. **31**(2), 183–208 (1978)
20. Money, J., Ye, Q.: EIGIFP: a MATLAB program for solving large symmetric generalized eigenvalue problems. ACM Trans. Math. Softw. **31**, 270–279 (2005)
21. Morgan, R.: A restarted GMRES method augmented with eigenvectors. SIAM J. Matrix Anal. Appl. **16**(4), 1154–1171 (1995)
22. Morgan, R.: GMRES with deflated restarting. SIAM J. Sci. Comput. **24**(1), 20–37 (2002)
23. Nachtigal, N.M., Reichel, L., Trefethen, L.N.: A hybrid GMRES algorithm for nonsymmetric linear systems. SIAM J. Matrix Anal. Appl. **13**(3), 796–825 (1992)
24. Polyak, B.T.: Introduction to Optimization. Optimization Software, New York (1987)
25. Quillen, P., Ye, Qiang: A block inverse-free preconditioned Krylov subspace method for symmetric generalized eigenvalue problems. J. Comput. Appl. Math. **233**(5), 1298–1313 (2010)
26. Saad, Y.: Least squares polynomials in the complex plane and their use for solving nonsymmetric linear systems. SIAM J. Numer. Anal. **24**(1), 155–169 (1987)
27. Saad, Y.: Iterative Methods for Sparse Linear Systems. PWS Publishing Company, Boston (1996)

28. Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd edn. SIAM, Philadelphia (2003)
29. Saad, Y., Schultz, M.: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**, 856–869 (1986)
30. Smolarski, D.C., Saylor, P.E.: An optimum iterative method for solving any linear system with a square matrix. BIT **28**(1), 163–178 (1988)
31. Takahashi, I.: A note on the conjugate gradient method. Inform. Process. Jpn. **5**, 45–49 (1965)
32. Yang, C., Meza, J.C., Lee, B., Wang, L.-W.: KSSOLV—a MATLAB toolbox for solving the Kohn–Sham equations. ACM Trans. Math. Softw. **36**(2), 1–35 (2009)