Journal of Computational and Applied Mathematics 331 (2018) 166–177

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Restarted Hessenberg method for solving shifted nonsymmetric linear systems

Xian-Ming Gu [a,b], Ting-Zhu Huang [a,*], Guojian Yin [c], Bruno Carpentieri [d], Chun Wen [a], Lei Du [e]

[a] School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, PR China
[b] Johann Bernoulli Institute of Mathematics and Computer Science, University of Groningen, Nijenborgh 9, P.O. Box 407, 9700 AK Groningen, The Netherlands
[c] School of Mathematics, Sun Yat-sen University, Guangzhou, Guangdong 510275, PR China
[d] School of Science and Technology, Nottingham Trent University, Clifton Campus, Nottingham, NG11 8NS, UK
[e] School of Mathematical Sciences, Dalian University of Technology, Dalian, Liaoning 116024, PR China

ARTICLE INFO

Article history:
Received 2 May 2016
Received in revised form 20 November 2016

MSC :
65F10
65F15
15A06
15A18

Keywords:
Shifted linear system
Hessenberg process
Pivoting strategy
Restarted Hessenberg method
Collinear
Fractional differential equations

ABSTRACT

It is known that the restarted full orthogonalization method (FOM) outperforms the restarted generalized minimum residual (GMRES) method in several circumstances for solving shifted linear systems when the shifts are handled simultaneously. Many variants of them have been proposed to enhance their performance. We show that another restarted method, the restarted Hessenberg method (Heyouni, 1996) based on Hessenberg procedure, can effectively be employed, which can provide accelerating convergence rate with respect to the number of restarts. Theoretical analysis shows that the new residual of shifted restarted Hessenberg method is still collinear with each other. In these cases where the proposed algorithm needs less enough elapsed CPU time to converge than the earlier established restarted shifted FOM, the weighted restarted shifted FOM, and some other popular shifted iterative solvers based on the short-term vector recurrence, as shown via extensive numerical experiments involving the recently popular application of handling time fractional differential equations.

1. Introduction

Considering a real large-scale sparse nonsymmetric matrix $A \in \mathbb{R}^{n\times n}$ and the right-hand side $\boldsymbol{b} \in \mathbb{R}^n$, we are interested in simultaneously solving shifted nonsingular linear systems

$$(A - \sigma_i I)\boldsymbol{x} = \boldsymbol{b}, \quad \sigma_i \in \mathbb{C}, \quad i = 1, 2, \ldots, \nu, \tag{1.1}$$

where I denotes the $n \times n$ identity matrix. Such shifted systems often arise in many scientific and engineering fields, such as control theory [1,2], structural dynamics [3], eigenvalue computations [4], numerical solutions of time-dependent partial/fractional differential equations [5,6], QCD problems [7], image restorations [8] and other simulation problems [9–12]. Among all the systems, when $\sigma_i = 0$, the linear system $A\boldsymbol{x} = \boldsymbol{b}$ is usually treated as the seed system.

* Corresponding author.
E-mail addresses: guxianming@live.cn, x.m.gu@rug.nl (X.-M. Gu), tingzhuhuang@126.com (T.-Z. Huang), guojianyin@gmail.com (G. Yin), bcarpentieri@gmail.com (B. Carpentieri), wchun17@163.com (C. Wen), dulei@dlut.edu.cn (L. Du).

https://doi.org/10.1016/j.cam.2017.09.047
0377-0427/

It is well known that Krylov subspace methods are widely used for the solution of linear systems (see e.g. [13]). Denoting the k-dimensional Krylov subspace with respect to A and $\boldsymbol{b}$ by

$$\mathcal{K}_k(A, \boldsymbol{v}) := \text{span}\{\boldsymbol{v}, A\boldsymbol{v}, \ldots, A^{k-1}\boldsymbol{v}\}, \tag{1.2}$$

we can observe that the relation, so-called shift-invariance property, described below always holds for the shifted matrices in

$$\mathcal{K}_k(A, \boldsymbol{v}) := \mathcal{K}_k(A - \sigma_i I, \boldsymbol{v}), \quad i = 1, 2, \ldots, \nu, \tag{1.3}$$

which shows that the iterations of (1.1) are dependent on the same Krylov subspace as the iterations of the seed system. This implies that if we choose the initial vector $\boldsymbol{x}_0$ properly (for example, all the initial guesses are zero), once a basis has been generated for one of these linear systems, it could also be reused for all other linear systems. Therefore, if we employ a Krylov subspace method to solve (1.1) simultaneously, a certain amount of computational efficiency can be maintained if the Krylov subspace is the same for all shifted systems each time. This happens when the generating vectors are collinear, for the basis and the square Hessenberg matrix are required to be evaluated only once; refer, e.g., to [9,11,14–16] for details.

Several numerical techniques have been proposed in the past few years that attempt to tackle this kind of linear systems (1.1). For shifted nonsymmetric (non-Hermitian) linear systems, iterative methods such as the shifted quasi-minimal residual (QMR) method, the shifted transpose-free QMR (TFQMR) method [3,17], the shifted induced dimension reduction (IDR(s)) method [9,18,19], and the shifted QMR variant of the IDR (QMRIDR(s)) method [20] have been developed. It should be mentioned that iterative methods based on the conventional Bi-Lanczos process [13, pp. 229–233] (or the A-biorthogonalization procedure [21, pp. 40–45]) have also been constructed for solving shifted non-Hermitian linear systems. Extensions of these methods, such as the shifted biconjugate gradient method and its stabilized variants, namely BiCG/BiCGStab(ℓ) [14,22], the shifted biconjugate residual method and the corresponding stabilized variant, i.e. BiCR/BiCRSTAB [23], and the shifted generalized product-type methods based on BiCG (GPBiCG), have been recently established in [24]. In addition, a recycling BiCG method was introduced and employed for handling shifted nonsymmetric linear systems from model reduction [25]. These Krylov subspace methods based on Bi-Lanczos-like procedures are not emphasized in this paper, but these alternatives are still worth mentioning.

On the other hand, the restarted generalized minimal residual (GMRES)-type methods are widely known and appreciated to be efficient on (1.1), refer to [3,26–29] for details, the computed GMRES shifted residuals are not collinear in general after the first restart so that it loses the computational efficiency mentioned above. Consequently, certain enforcement has to be made for guaranteeing the computed GMRES shifted residuals collinear to each other in order to maintain the computational efficiency; see e.g. [3,26]. Note that in this case, only the seed system has the minimum residual property, the solution of the other shifted systems is not equivalent to the GMRES method applied to those linear systems, refer to [3,16,26]. In contrast, it is more natural and more effective for the restarted full orthogonalization method (FOM) to be applied to shifted linear systems simultaneously handled, for all residuals are naturally collinear [11,30]. As a result, the computational efficiency can be maintained because the orthonormal basis and the Hessenberg matrix are required to be calculated only once each time. Jing and Huang in [15] further accelerated this method by introducing a weighted norm (i.e., the weighted Arnoldi process). In 2014, Yin and Yin have studied restarted FOM with the deflation technique which is first introduced by Morgan in [31] for solving all shifted linear systems simultaneously. Due to restarting generally slows the convergence of FOM by discarding some useful information at the restart, the deflation technique can remedy this disadvantage in some sense by keeping Ritz vectors from the last cycle, see [16] for details.

However, as we know, both the restarted GMRES method and the restarted FOM for solving shifted linear systems are derived by using the Arnoldi procedure [13, pp. 160–165], which turns to be expensive when m (the dimension of Krylov subspace) becomes large because of the growth of memory and computational requirements as m increases. So it is still meaningful to search some cheaper iterative methods for solving shifted linear systems (1.1). Here, we consider to exploit the Hessenberg reduction process [32–35] because it generally requires less arithmetic operations and storage than the Arnoldi process and is thus favorable for producing a linear system solver. Moreover, it has been proved that we can establish two families of Krylov subspace methods, namely the Hessenberg method [35] and the changing minimal residual method based on the Hessenberg process (CMRH) [33–35], by using the basic principles behind the (restarted) FOM and the (restarted) GMRES method, respectively, refer to [35,36] for this discussion. Some recent developments concerning the CMRH, which is very similar to the GMRES method, and the Hessenberg process can be found in [34,37–39]. Since the restarted FOM is built via combining the Arnoldi process and Galerkin-projection idea [13, pp. 165–168], so it is natural to extend the restarted FOM for solving shifted linear systems (1.1), refer to [30] for details. Meanwhile, the restarted Hessenberg method is also established via combining the Hessenberg process with Galerkin-projection philosophy. Moreover, as mentioned earlier, the Hessenberg process has many similar algorithmic properties of the Arnoldi process. To sum up, the framework of the restarted FOM for shifted linear systems gave us a simple and natural problem: Does there exist a variant of the original restarted Hessenberg method for solving shifted linear systems? The major contribution of the current paper is to answer this question. The answer is yes, and it requires a similar but efficient idea from that used in the restarted shifted FOM. As a result, a feature of the resulting algorithm is that all residuals are naturally collinear in each restarted cycle; and the computational efficiency can be maintained because the (non-orthogonal) basis and the square Hessenberg matrix are required to be calculated only once each time. Our method indeed may provide the attractive convergence behavior with

respect to the less number of restarts and the elapsed CPU time, which will be shown by numerical experiments described in Section 4. Moreover, the established algorithm is able to solve certain shifted systems which both the restarted shifted FOM and the restarted weighted shifted FOM [15] cannot handle sometimes.

The remainder of the present paper is organized as follows. In Section 2, we briefly review the Hessenberg process and the restarted Hessenberg method for nonsymmetric linear systems. Section 3 discusses the naturally collinear property of residual during each cycle of the restarted Hessenberg method. Then, we demonstrate how to generalize the restarted shifted Hessenberg method for solving shifted linear systems (1.1). Implementation details will be also described. In Section 4, extensive numerical experiments are reported to illustrate the effectiveness of the proposed method. Finally, some conclusions about this method are drawn in Section 5.

2. The Hessenberg process and the restarted Hessenberg method

In order to extend the restarted Hessenberg method for solving shifted linear systems well, we first recall the restarted Hessenberg method, which is established from the Hessenberg process. According to Refs. [35,37,40], it is not hard to conclude that the restarted Hessenberg method is greatly close to the restarted FOM, which is derived from the well-known Arnoldi process. Although the restarted Hessenberg method has been proved to be cheaper than the restarted FOM, the restarted Hessenberg method is still not very popular in the field of Krylov subspace methods for solving linear systems. This observation is just like the (restarted) FOM, which is simpler but often less popular than the GMRES method due to the minimal norm property of the latter [13]. However, as mentioned earlier, the (restarted) FOM is attractively extended for solving shifted linear systems, here this motivation also encourages us to revive the restarted Hessenberg method for solving such shifted systems (1.1).

2.1. The Hessenberg process

Starting point of the algorithms derived in this paper is the Hessenberg process for reducing a given nonsymmetric matrix to the Hessenberg decomposition [32,34]. In [41], the Hessenberg process is originally described as an algorithm for computing the characteristic polynomial of a given matrix A. This process can also be applied for the reduction to the Hessenberg form of A and is presented as an oblique projection in [32, pp. 377–381]. For ease of notation we will assume that both the matrix and the vectors involved in the solution algorithms are real, but the results given here and in other sections are easily modified for a complex matrix and complex vectors. Exploiting the pivoting strategy described above, the Hessenberg procedure can be reproduced in Algorithm 1.

Algorithm 1 The Hessenberg procedure with pivoting strategy

1: Set $\boldsymbol{p} = [1, 2, \ldots, n]^T$ and determine i_0 such that $|(\boldsymbol{v})_{i_0}| = \|\boldsymbol{v}\|_\infty$
2: Compute $\beta = (\boldsymbol{v})_{i_0}$, then $\boldsymbol{l}_1 = \boldsymbol{v}/\beta$ and $\boldsymbol{p}(1) \leftrightarrow \boldsymbol{p}(i_0)$
3: **for** $j = 1, 2, \ldots, k$, **do**
4: Compute $\boldsymbol{u} = A\boldsymbol{l}_j$
5: **for** $i = 1, 2, \ldots, j$, **do**
6: $h_{i,j} = (\boldsymbol{u})_{\boldsymbol{p}(i)}$
7: $\boldsymbol{u} = \boldsymbol{u} - h_{i,j}\boldsymbol{l}_i$
8: **end for**
9: **if** ($j < n$ and $\boldsymbol{u} \neq \boldsymbol{0}$) **then**
10: Determine $i_0 \in \{j+1, \ldots, n\}$ such that $|(\boldsymbol{u})_{\boldsymbol{p}(i_0)}| = \|(\boldsymbol{u})_{\boldsymbol{p}(j+1):\boldsymbol{p}(n)}\|_\infty$;
11: $h_{j+1,j} = (\boldsymbol{u})_{\boldsymbol{p}(i_0)}$; $\boldsymbol{l}_{j+1} = \boldsymbol{u}/h_{j+1,j}$; $\boldsymbol{p}(j+1) \leftrightarrow \boldsymbol{p}(i_0)$
12: **else**
13: $h_{j+1,j} = 0$; **Stop**
14: **end if**
15: **end for**

Let L_k be the $n \times k$ matrix with column vectors $\boldsymbol{l}_1, \ldots, \boldsymbol{l}_k$, $\bar{H}_k$ be the $(k+1) \times k$ upper Hessenberg matrix whose nonzero entries are the $h_{j,k}$ and by H_k the matrix obtained from $\bar{H}_k$ by deleting its last row. Then it is not hard to demonstrate that these matrices given either by Algorithm 1 satisfy the well-known formulas

$$\begin{aligned} AL_k &= L_{k+1}\bar{H}_k \\ &= L_kH_k + h_{k+1,k}\boldsymbol{l}_{k+1}\boldsymbol{e}_k^T \end{aligned} \tag{2.1}$$

and $\mathcal{P}_kL_k$ is lower trapezoidal where $\mathcal{P}_k^T = [\boldsymbol{e}_{p_1}, \boldsymbol{e}_{p_2}, \ldots, \boldsymbol{e}_{p_n}]$ and the p_i's (for $i = 1, \ldots, n$) are defined in Algorithm 1, refer to [33,34] for details. In [34], it is worth mentioning that Heyouni and Sadok had introduced the Hessenberg process with over-storage to deal with the dense matrix for saving the computational storage, but here we will not pursue it in detail.

At the end of this subsection, it is worth mentioning that the earlier work[1] of Howell & Stephens [43] and Stephens' Ph.D. dissertation [42] have really made some further progress on the backward error analysis for the Hessenberg process. They had obtained the following theorem, which can be regarded as a slight improvement on Wilkinson's results [32]. For a proof, one can consult Stephens' Ph.D. dissertation [42].

Theorem 2.1. *Let H_k be the first k columns of $\bar{H}_k$ computed in floating point arithmetic by the Hessenberg algorithm. Then assume that $\tilde{A}$ is a permutation of A from which H_k is produced. If the (i, j)-th entry of $\tilde{A}$ is a_{ij} and denote $|\tilde{A}|$ as the matrix with entries $|a_{ij}|$.*

$$(\tilde{A} + \triangle A)L_k = L_{k+1}\bar{H}_k, \quad |\triangle A| \leq \gamma_n(|\tilde{A}||L_k| + |L_{k+1}||\bar{H}_k|),$$

where $\gamma_n = n\epsilon/(1 - n\epsilon)$ and ϵ is the unit roundoff (or machine precision) [44, p. 3] *such that $1 = fl(1 + \epsilon)$ in which "$fl(\cdot)$" indicates correctly rounded floating-point arithmetic.*

According to Theorem 2.1, the Hessenberg process with pivoting strategy cannot be proved to be backward stable in finite precision arithmetic. Meanwhile, this result (i.e., Theorem 2.1) also indicated that for most problems the backward error is usually small [42,43, p. 49]. Moreover, in our practical implementations of most test problems considered in the current study, no noticeable instabilities of the Hessenberg process with pivoting strategy have been detected. Based on the above observations, Algorithm 1 can be promising and thus represents a cost-effective alternative to the Arnoldi procedure in iterative solutions of some certain systems of linear equations.

2.2. The restarted Hessenberg method

As we know, it derives the restarted FOM from the specified Hessenberg decomposition like (2.1), which is generated by Arnoldi process. Here we follow this framework of restarted FOM to derive the restarted Hessenberg method via combining the Hessenberg decomposition with the Galerkin-projection idea. Given an initial guess $\boldsymbol{x}_0$ to the seed linear system $A\boldsymbol{x} = \boldsymbol{b}$, we now consider an orthogonal projection method [13], which takes $\mathcal{L} = \mathcal{K}_m(A, \boldsymbol{r}_0)$ in which $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$. Then we search an approximate solution $\boldsymbol{x}_m$ from the affine subspace $\boldsymbol{x}_0 + \mathcal{K}_m(A, \boldsymbol{r}_0)$ of dimension m, i.e., we can express it as $\boldsymbol{x}_m = \boldsymbol{x}_0 + L_m\boldsymbol{y}_m$ for the vector $\boldsymbol{y}_m$, where $L_m = [\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_m]$ is constructed via the Hessenberg process. Furthermore, the residual vector can be computed

$$\begin{aligned} \boldsymbol{r}_m &= \boldsymbol{b} - A\boldsymbol{x}_m = \boldsymbol{r}_0 - L_m H_m \boldsymbol{y}_m - h_{m+1,m}\boldsymbol{l}_{m+1}\boldsymbol{e}_m^T\boldsymbol{y}_m \\ &= L_m(\beta\boldsymbol{e}_1 - H_m\boldsymbol{y}_m) - h_{m+1,m}\boldsymbol{l}_{m+1}\boldsymbol{e}_m^T\boldsymbol{y}_m. \end{aligned} \tag{2.2}$$

Then in this method, we need to enforce the following Galerkin condition:

$$\boldsymbol{r}_m \perp \{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_m\}, \tag{2.3}$$

where $\boldsymbol{e}_i$ is the ith vector of the canonical basis of $\mathbb{R}^n$. This orthogonality condition yields that $\boldsymbol{y}_m$ is the solution of the following $m \times m$ linear system,

$$H_m\boldsymbol{y}_m = \beta\boldsymbol{e}_1, \tag{2.4}$$

refer to [34,35,40] for details. As a consequence, the approximate solution using the above m-dimensional subspace is given by

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + L_m\boldsymbol{y}_m, \quad \text{where} \quad \boldsymbol{y}_m = H_m^{-1}(\beta\boldsymbol{e}_1). \tag{2.5}$$

Finally, an iterative solver based on Algorithm 1 and called the Hessenberg (Hessen) method is obtained, but for practical implementation, here we give the pseudo-codes of the restarted Hessenberg method as Algorithm 2.

Algorithm 2 The restarted Hessenberg method (referred to as Hessen(m))

1: **Start:** Choose $\boldsymbol{x}_0 \in \mathbb{R}^n$, the restarting frequency $m \in Z^+$. Compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$ and set $\boldsymbol{p} = [1, 2, \ldots, n]^T$ and determine i_0 such that $|(\boldsymbol{r}_0)_{i_0}| = \|\boldsymbol{r}_0\|_\infty$
2: Compute $\beta = (\boldsymbol{r}_0)_{i_0}$, then $\boldsymbol{l}_1 = \boldsymbol{r}_0/\beta$ and $\boldsymbol{p}(1) \leftrightarrow \boldsymbol{p}(i_0)$, where $\leftrightarrow$ is used to swap contents.
3: **Hessenberg process:** Generate the Hessenberg basis and the matrix H_m using the Hessenberg process (i.e. Algorithm 1) starting with $\boldsymbol{l}_1$.
4: **Approximate the solution:** Solve $\boldsymbol{y} = H_m^{-1}(\beta\boldsymbol{e}_1)$ and update $\boldsymbol{x}_m = \boldsymbol{x}_0 + L_m\boldsymbol{y}_m$, where $L_m = [\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_m]$.
5: **Restart:** If converged then stop; otherwise set $\boldsymbol{x}_0 := \boldsymbol{x}_m$ and goto 1.

The above algorithm depends on a parameter m which is the dimension of the Krylov subspace. In practice it is desirable to select m in a dynamic fashion. This would be possible if the residual norm of the solution $\boldsymbol{x}_m$ is available inexpensively

[1] A short note for describing the relations between their ELMRES method [42,43] and Sadok's CMRH [33] is available online at http://ncsu.edu/hpc/Documents/Publications/gary_howell/contents.html#codes.

(without having to compute $\boldsymbol{x}_m$ itself). Then the algorithm can be stopped at the appropriate step using this information. The following proposition gives a result in this direction.

Proposition 2.1. *The residual vector of the approximate solution $\boldsymbol{x}_m$ computed by the Hessenberg method (without restarting) is such that*

$$\boldsymbol{r}_m = \boldsymbol{b} - A\boldsymbol{x}_m = -h_{m+1,m}[\boldsymbol{y}_m]_m \boldsymbol{l}_{m+1},$$

where $[\boldsymbol{y}_m]_m$ represents the last element of $\boldsymbol{y}_m$ and therefore

$$\|\boldsymbol{b} - A\boldsymbol{x}_m\|_2 = \left|h_{m+1,m}[\boldsymbol{y}_m]_m\right| \cdot \|\boldsymbol{l}_{m+1}\|_2. \tag{2.6}$$

Proof. With the help of Eqs. (2.2) and (2.5), we can note that

$$\begin{aligned}\boldsymbol{r}_m &= L_m(\beta\boldsymbol{e}_1 - H_m\boldsymbol{y}_m) - h_{m+1,m}\boldsymbol{l}_{m+1}\boldsymbol{e}_m^T\boldsymbol{y}_m \\ &= -h_{m+1,m}\boldsymbol{l}_{m+1}\boldsymbol{e}_m^T\boldsymbol{y}_m\end{aligned}$$

because of the definition of $\boldsymbol{y}_m$, i.e., $H_m\boldsymbol{y}_m = \beta\boldsymbol{e}_1$. This can immediately lead to the identity in Eq. (2.6) by using the 2-norm. □

The previous results are also achieved for the FOM [13, Proposition 6.7], except that we have $\|L_{m+1}\|_2 = 1$ and $h_{m+1,m} > 0$, and so

$$\boldsymbol{r}_m^{fom} = -h_{m+1,m}[\boldsymbol{y}_m]_m \boldsymbol{l}_{m+1} \quad \text{and} \quad \|\boldsymbol{r}_m\|_2 = h_{m+1,m}\left|[\boldsymbol{y}_m]_m\right|. \tag{2.7}$$

Also note that these formulas imply that the 2-norm of the residual can be determined, without having to compute the correction x_m. At the end of this subsection, it follows that

$$\boldsymbol{r}_m = -h_{m+1,m}[\boldsymbol{y}_m]_m \boldsymbol{l}_{m+1}, \tag{2.8}$$

then we have $\boldsymbol{r}_m = \beta_m \boldsymbol{l}_{m+1}$ and $\beta_m := -h_{m+1,m}[\boldsymbol{y}_m]_m$, which indicates that $\boldsymbol{r}_m$ is naturally collinear with $\boldsymbol{l}_{m+1}$.

Without considering the restarting strategy, just like analyzing the convergence relations between the CMRH and the GMRES method, we may also follow the analogous ideas of Sadok & Szyld [37], Duintjer Tebbens & Meurant [38], and Schweitzer [45] to present some specified analyses which explain why the Hessenberg method can have the good convergence behavior. But this is not the emphasis of our present paper.

3. The shifted variant of the restarted Hessenberg method

Based on the above mentioned, we follow Simoncini's framework [30] about deriving the restarted shifted FOM to establish the shifted variant of the restarted Hessenberg method. Consider now the shifted systems (1.1). Shifting transforms (2.1) into

$$(A - \sigma_i I)L_m = L_m(H_m - \sigma_i I_m) + h_{m+1,m}\boldsymbol{l}_{m+1}\boldsymbol{e}_m^T\boldsymbol{y}_m, \tag{3.1}$$

where I_m is the identity matrix of order m. Due to (3.1), the only difference in the Hessenberg method is that $\boldsymbol{y}_m$ is calculated via solving the reduced shifted systems $(H_m - \sigma_i I_m)\boldsymbol{y} = \beta\boldsymbol{e}_1$. Therefore, the expensive step of constructing the non-orthogonal basis L_m is performed only once for all values of σ_i of interest, $i \in \{1, \ldots, \nu\}$, whereas ν reduced systems of size m need to be solved. This is the case if the right-hand sides are collinear. In the following, we shall assume that $\boldsymbol{x}_0 = \boldsymbol{0}$ so that all shifted systems have the same right-hand side. Restarting can also be employed in the shifted case. The key fact is that the Hessenberg method residual $\boldsymbol{r}_m$ is a multiple of the basis vector $\boldsymbol{l}_{m+1}$, see Eq. (2.8) for details. The next proposition shows that collinearity still holds in the shifted case when the Hessenberg method is exploited.

Proposition 3.1. *For each $i = 1, 2, \ldots, \nu$, let $\boldsymbol{x}_m^{(i)} = L_m\boldsymbol{y}_m^{(i)}$ be a Hessenberg method approximate solution to $(A - \sigma_i I)\boldsymbol{x} = \boldsymbol{b}$ in $\mathcal{K}_m(A - \sigma_i I, \boldsymbol{b})$, with L_m satisfying (3.1). Then there exists $\beta_m^{(i)} \in \mathbb{R}$ such that $\boldsymbol{r}_m^{(i)} = \boldsymbol{b} - (A - \sigma_i I)\boldsymbol{x}_m^{(i)} = \beta_m^{(i)}\boldsymbol{l}_{m+1}$.*

Proof. For $i = 1, 2, \ldots, \nu$, we have

$$\begin{aligned}\boldsymbol{r}_m^{(i)} &= \boldsymbol{b} - (A - \sigma_i I)\boldsymbol{x}_m^{(i)} = \boldsymbol{r}_0 - (A - \sigma_i I)L_m\boldsymbol{y}_m^{(i)} \\ &= \boldsymbol{r}_0 - L_m(H_m - \sigma_i I_m)\boldsymbol{y}_m^{(i)} - h_{m+1,m}\boldsymbol{l}_{m+1}\boldsymbol{e}_m^T\boldsymbol{y}_m^{(i)} \\ &= L_m\left[\beta\boldsymbol{e}_1 - (H_m - \sigma_i I_m)\boldsymbol{y}_m^{(i)}\right] - h_{m+1,m}[\boldsymbol{y}_m^{(i)}]_m\boldsymbol{l}_{m+1}.\end{aligned}$$

Setting $\beta_m^{(i)} = -h_{m+1,m}[\boldsymbol{y}_m^{(i)}]_m$, $i = 1, 2, \ldots, \nu$, we obtain $\boldsymbol{r}_m^{(i)} = \beta_m^{(i)}\boldsymbol{l}_{m+1}$. □

It is observed that all the residuals $\boldsymbol{r}_m^{(i)}$ are collinear with $\boldsymbol{l}_{m+1}$, and thus they are collinear with each other. This property is excellent so that we could restart the shifted Hessenberg method by taking the common vector $\boldsymbol{l}_{m+1}$ as the new initial vector, and the corresponding approximate Krylov subspace is $\mathcal{K}_m(A, \boldsymbol{l}_{m+1})$. Just as the first cycle, all the new residuals still satisfy the formula $\boldsymbol{r}_m^{(i)} = \beta_m^{(i)}\boldsymbol{l}_{m+1}$, and the restarted Hessenberg process can be repeated until convergence. This leads to the shifted restarted Hessenberg method for simultaneously solving shifted linear systems (1.1). We described this final idea in detail as following Algorithm 3.

Algorithm 3 The restarted shifted Hessenberg method

1: Given A, $\boldsymbol{b}$, $\boldsymbol{x}_0 = \boldsymbol{0}$, $\{\sigma_1, \ldots, \sigma_\nu\}$, $\mathcal{I} = \{1, 2, \ldots, \nu\}$ and the restarting frequency $m \in Z^+$.
2: Set $\boldsymbol{r}_0 = \boldsymbol{b}$ and take $\boldsymbol{p} = [1, 2, \ldots, n]^T$ and determine i_0 such that $|(\boldsymbol{r}_0)_{i_0}| = \|\boldsymbol{r}_0\|_\infty$
3: Compute $\beta_0^{(i)} = (\boldsymbol{r}_0)_{i_0}$, then $\boldsymbol{l}_1 = \boldsymbol{r}_0/\beta_0^{(i)}$, $\boldsymbol{p}(1) \leftrightarrow \boldsymbol{p}(i_0)$, where $\leftrightarrow$ is used to swap contents.
4: Set $\boldsymbol{x}_m^{(i)} = \boldsymbol{x}_0$, $i = 1, 2, \ldots, \nu$.
5: Compute the Hessenberg decomposition $AL_k = L_k H_k + h_{k+1,k}\boldsymbol{l}_{k+1}\boldsymbol{e}_k^T$ by Algorithm 1
6: **for** each $i \in \mathcal{I}$ **do**
7: Solve $\boldsymbol{y}_m^{(i)} = (H_m - \sigma_i I_m)^{-1}(\beta_m^{(i)}\boldsymbol{e}_1)$
8: Update $\boldsymbol{x}_m^{(i)} = \boldsymbol{x}_m^{(i)} + L_m\boldsymbol{y}_m^{(i)}$
9: **end for**
10: Eliminate converged systems. Update $\mathcal{I}$. If $\mathcal{I} = \emptyset$, exit. EndIf
11: Set $\beta_m^{(i)} = -h_{m+1,m}[\boldsymbol{y}_m^{(i)}]_m$ for each $i \in \mathcal{I}$
12: Set $\boldsymbol{l}_1 = \boldsymbol{l}_{m+1}$. Goto 5

Similarly, it was shown in [30] that the information sharing does not cause any degradation of convergence performance, and the convergence history of shifted Hessenberg method on each system is the same as that of the usual restarted Hessenberg method applied individually to each shifted system. We should point out that an outstanding advantage of this approach is that the non-orthogonal basis $\{\boldsymbol{l}_1, \ldots, \boldsymbol{l}_m\}$ is only required to be computed once for solving all shifted systems in each cycle, so that a number of computational cost can be saved. In addition, Algorithm 3 is also attractive when both A are real while the shifts $\{\sigma_i\}$'s are complex. Indeed, at each cycle after restarting, all the complex residuals are collinear to the $(m+1)$-st real basis vector L_{m+1}, and the expensive step for constructing the non-orthogonal basis L_{m+1} can be performed in real arithmetics, see [17,30] and Section 4 for this issue.

Next, we shall analyze the computational cost of implementing the restarted shifted Hessenberg method, the restarted shifted FOM, and the weighted restarted shifted FOM. It is known from Section 3 and Refs. [15,30] that the main difference of arithmetic operations of these three methods comes from processes in producing the basis vectors of Krylov subspaces. Other operational requirements, like solving ν linear sub-systems defined as Line 7 in Algorithm 3 and the update of $\boldsymbol{y}_m^{(i)}$, are similar for the three mentioned methods. Therefore, it makes sense to only consider the computational cost of the Hessenberg, Arnoldi and weighted Arnoldi processes which underpin the implementation of Algorithm 3, the restarted shifted FOM and the weighted restarted shifted FOM. Let us denote by Nz the number of nonzero entries of A in (1.1). The cost of an inner product is assumed to be $2n$ flops. Since the first $j-1$ elements of $\boldsymbol{l}_j$ are zero, then some arithmetic operations can be saved. For instance, the cost of updating the vector $\boldsymbol{l}_j$ (the j-loop in Algorithm 1) in the Hessenberg process reduces to $\sum_{i=1}^{m}\sum_{j=1}^{i} 2(n-(j-1)) = m(m+1)(n-(m-1)/3)$ flops instead of $2m(m+1)n$ and $\frac{5}{2}m(m+1)n$ flops in the Arnoldi process and the weighted Arnoldi process, respectively. If we neglect the cost of computing the maximum of the vector $\boldsymbol{l}_j$ in the Hessenberg procedure, then we obtain the number of operations per restart (i.e., m steps) by the Hessenberg process, the Arnoldi process and the weighted Arnoldi process (refer to [15] for instance) as Table 1.

In Table 1, m is the restarting frequency; and the definition of "D-orthogonal" basis can be found in [15, Algorithm 2]. Firstly, it is worth mentioning that the restarted shifted Hessenberg, the restarted shifted FOM, and the weighted restarted shifted FOM have the similar implementations of Lines 6–12 of Algorithm 3. More precisely, if we suppose that it requires to run those three Krylov subspace solvers within $\xi^\imath$, $\imath \in \{f, h, wf\}$ restarting cycles, respectively, then their algorithmic costs (roughly) read as,

- Restarted shifted FOM: $\xi^f c^f + \xi^f(2mNz + 2m(m+1)n)$;
- Restarted shifted Hessenberg method: $\xi^h c^h + \xi^h\Big(2mNz + m(m+1)n - \frac{1}{3}m(m-1)(m+1)\Big)$;
- Restarted weighted shifted FOM: $\xi^{wf} c^{wf} + \xi^{wf}\Big(2mNz + \frac{5}{2}m(m+1)n\Big)$,

where c^f, c^h and c^{wf} denote respectively the total cost of the restarted shifted FOM, the restarted shifted Hessenberg method and the restarted weighted shifted FOM for implementing Lines 6–12 of Algorithm 3 per restarting cycle. In fact, the main cost of these seven lines of Algorithm 3 (or corresponding pseudo-codes of the restarted shifted FOM and the restarted weighted shifted FOM) is to solve ν linear systems with shifted Hessenberg coefficient matrices of size $m \times m$, which can be solved in $\mathcal{O}(\nu m^2)$ operations via QR factorization [46]. This fact implies that we can have $c^\jmath \sim \mathcal{O}(\nu m^2)$, $\jmath \in \{f, h, wf\}$.

As observed from the above considerations, when these three mentioned shifted iterative solvers need to the similar number of restarts for solving shifted linear systems (1.1), i.e., $\xi^f \approx \xi^h \approx \xi^{wf}$, it clearly finds that the total algorithmic cost of the restart shifted Hessenberg method can be less than that of both the restarted shifted FOM and the weighted restarted shifted FOM, also see Section 4 for further discussions from numerical experiments.

4. Numerical examples

Far from being exhaustive, in this section, the feasibility of the restarted shifted Hessenberg method is demonstrated for four different, but representative groups of practical problems. We will compare the proposed method (referred to

Table 1
Comparison of the Arnoldi, weighted Arnoldi, and Hessenberg procedures.

Process	Number of operations	Orthogonal basis
Arnoldi	$2mNz + 2m(m+1)n$	Yes
Weighted Arnoldi	$2mNz + \frac{5}{2}m(m+1)n$	D-orthogonal
Hessenberg	$2mNz + m(m+1)n - \frac{1}{3}m(m-1)(m+1)$	No

Table 2
Set and characteristics of test problems in Example 1 (listed in increasing matrix size).

Index	Matrix	Size	Field	$nnz(A)$	$\sigma_j\ (j = 1, 2, \ldots, 8)$
Σ_1	Poisson3Da	13,514	Computational fluid dynamics	352,762	$\sigma_j = -8j \times 10^{-5}$
Σ_2	memplus	17,758	Circuit simulation problem	99,147	$\sigma_j = -j \times 10^{-4}$
Σ_3	FEM_3D_thermal1	17,880	Thermal problem	430,740	$\sigma_j = -j \times 10^{-3}$
Σ_4	Grond4e4	40,000	2D/3D problem	199,200	$\sigma_j = -j \times 10^{-3}$
Σ_6	shyy161	76,480	Computational fluid dynamics	329,762	$\sigma_j = -5j \times 10^{-2}$
Σ_5	vfem	93,476	Electromagnetics problem	1434,636	$\sigma_j = -5j \times 10^{-5}$

Table 3
Compared results about different shifted Krylov subspace solvers for Example 1 in aspects of the MVPs and CPU.

Index	sHessen(m)		sFOM(m)		wsFOM(m)		sIDR(1)		sQMRIDR(1)		sBiCGSTAB(2)	
	MVPs	CPU	MVPs	CPU	MVPs	CPU	MVPs	CPU	MVPs	CPU	MVPs	CPU
Σ_1	360	0.709	320	0.774	‡	‡	228	0.772	245	0.737	220	1.245
Σ_2	640	0.794	960	1.682	‡	‡	931	1.121	1628	2.093	792	3.024
Σ_3	240	0.606	280	0.886	‡	‡	255	1.031	348	1.169	260	1.970
Σ_4	560	1.206	480	1.933	‡	‡	478	1.434	866	2.749	480	3.886
Σ_5	360	1.655	320	2.558	‡	‡	538	3.152	1899	11.086	392	13.308
Σ_6	280	4.910	320	8.287	‡	‡	464	12.327	481	14.749	440	22.222

as sHessen(m)) with the restarted shifted FOM (abbreviated as sFOM(m)) [30], the restarted weighted shifted FOM (referred to as wsFOM(m)) in [15], the shifted QMRIDR(1) method (abbreviated as sQMRIDR(1)), the shifted IDR(1) method (referred to as sIDR(1)), and the shifted BiCGSTAB(2) method (abbreviated as sBiCGSTAB(2)) in all the listed experiments. Numerical comparisons about the attractive convergence performance of iterative solvers are made in two main aspects: the number of matrix–vector products (abbreviated as MVPs) and the elapsed CPU time in seconds[2] (abbreviated as CPU), some numerical experiments involving have been reported in this section. The dimension of approximation subspace is chosen to be m.

Unless otherwise stated, the initial guess solutions $\boldsymbol{x}_0$ $(= \boldsymbol{x}_0^{(i)})$ and the right-hand side vector $\boldsymbol{b}$ are taken as $\boldsymbol{x}_0 = [0, 0, \ldots, 0]^T$ and $\boldsymbol{b} = [1, 1, \ldots, 1]^T$, respectively. Suppose that $\boldsymbol{x}_k^{(i)}$ are the approximate solutions in the kth cycle, we stop the iteration procedure if all the $\boldsymbol{x}_k^{(i)}$ satisfy

$$\frac{\|\boldsymbol{b} - (A - \sigma_i I)\boldsymbol{x}_k^{(i)}\|_2}{\|\boldsymbol{b}\|_2} < \text{tol} = 10^{-8}, \quad i \in \mathcal{I}$$

or when this condition of the relative residual was not satisfied within $Max_{mvps} = 4000$ iterations for all shifted linear systems (denoted by ‡). All experiments were performed on a Windows 7 (64 bit) PC-Intel(R) Core(TM) i5-3740 CPU 3.20 GHz, 8 GB of RAM using MATLAB 2014a with machine epsilon 10^{-16} in double precision floating point arithmetic.

Example 1. All large-scale sparse test matrices in this example are from the University of Florida Sparse Matrix Collection [47], except the matrix Grond4e4.[3] For the sake of convenience, properties of test problems (matrix size and the number of nonzero elements etc.) and choices of shift parameters σ_j $(j = 1, 2, \ldots, 8)$ are displayed in Table 2. The linear system $(A - \sigma_1 I)\boldsymbol{x}^{(1)} = \boldsymbol{b}$ was treated as the *seed* system. In addition, here we choose the restart frequency $m = 40$. The numerical results for different shifted iterative solvers for targeted linear systems (1.1) are reported in Table 3.

From the results reported in Table 3, the wsFOM(40) cannot test any problems in this example, i.e, its performance is not promising. Then it is indeed worth mentioning that the proposed sHessen(40) method is better than the sFOM(40) in terms of the elapsed CPU time and even the number of matrix–vector products (except Σ_1, Σ_3, and Σ_5), it exactly follows the cost analysis that presented in the end of Section 3 about sHessen(40) and sFOM(40). Moreover, our proposed sHessen(40) method is even more promising than the last three short-term recurrence shifted Krylov subspace solvers in aspects of the elapsed CPU time. In addition, it is worth mentioning that in two cases: sIDR(1), sQMRIR(1), and sBiCGSTABA(2) for Σ_1 and sIDR(1) & sBiCGSTABA(2) for Σ_4, although the sHessen(40) method requires more number

[2] All timings are averages over 10 runs of the proposed algorithms.

[3] See our GitHub repository at https://github.com/Hsien-Ming-Ku/UESTC-Math/tree/master/Problems.

Table 4
Compared results about different shifted Krylov subspace solvers for Example 2 in aspects of the MVPs and CPU. ($\sigma_j \in \mathcal{I}$, $m = 40$)

Index	sHessen(m)		sFOM(m)		wsFOM(m)		sIDR(1)		sQMRIDR(1)		sBiCGSTAB(2)	
	MVPs	CPU	MVPs	CPU	MVPs	CPU	MVPs	CPU	MVPs	CPU	MVPs	CPU
Π_1	560	0.498	480	0.651	‡	‡	376	0.571	416	1.176	360	1.909
Π_2	280	0.258	200	0.283	‡	‡	199	0.306	210	0.636	192	1.059
Π_3	240	0.233	200	0.297	‡	‡	206	0.326	216	0.659	196	1.103
Π_4	840	14.646	760	16.682	‡	‡	682	12.235	733	24.225	660	36.971
Π_5	760	13.798	680	15.126	‡	‡	672	11.941	729	23.316	652	36.566
Π_6	1280	22.383	1200	26.271	‡	‡	546	10.409	591	19.382	1068	29.479

of matrix–vector products than the other mentioned shifted solvers for corresponding test problems, the sHessen(40) method is still cheaper in terms of the elapsed CPU time. This is because other shifted Krylov subspace solvers based on short-term vector recurrences need more extra inner products and n-vectors updated, which are sometimes time-consuming [19,20,22]. As emphasized, our proposed sHessen(m) method still can be regarded as a highly recommended choice for solving shifted linear systems (1.1) considered in Example 1.

Example 2. In the second example, we performed experiments with a set of matrices coming from a lattice quantum chromodynamics (QCD) application downloaded from the University of Florida Sparse Matrix Collection [47]. The set of test matrices is a collection of three $3,072 \times 3.072$ and three $49,152 \times 49,152$ complex matrices, i.e., conf5_0-4x4-14, conf6_0-4x4-20, conf6_0-4x4-30, conf5_4-8x8-15, conf5_4-8x8-20 and conf6_0-8x8-20. Here we orderly denote these seven test matrices as indices Π_i ($i = 1, 2, \ldots, 6$). For each matrix D from the collection, there exists some critical value κ_c such that for $\frac{1}{\kappa_c} < \frac{1}{\kappa} < \infty$, the matrix $A = \frac{1}{\kappa} I - D$ is real-positive. For each D, we took $A = (\frac{1}{\kappa_c} + 10^{-3})I - D$ as our base matrix. As described in [28,47], all the matrices D are discretizations of the Dirac operator used in numerical simulations of quark behavior at different physical temperatures. In our numerical experiments, one set of shifted values $\mathcal{I} = -\{.001, .002, .003, .004, .005, .006, .01, .02, .03, .04, 0.05, 0.06\}$ for shifted linear systems is considered. Meanwhile, the linear system $A\boldsymbol{x} = \boldsymbol{b}$ was employed as the *seed* system. The numerical results about various Krylov subspace solvers for shifted linear systems (1.1) are displayed in Table 4.

As seen from Table 4, our proposed iterative solvers (sHessen(m)) can be successfully employed to solve the shifted linear systems (1.1) in Example 2, whereas the wsFOM(m) cannot do it at all. More precisely, the proposed method, sHessen(m), is more efficient and cheaper than both the sFOM(m) for solving shifted linear systems (1.1) in terms of the elapsed CPU time. Since the required number of MVPs are similar, the computational cost of the sHessen(m) method can be less than that of both sFOM(m) and wsFOM(m). For test problems (Π_1, Π_2, and Π_3), the sHessen(m) method is even more competitive than other short-term recurrence shifted Krylov subspace methods in aspects of the elapsed CPU time. Additionally, it highlighted that the sIDR(1) method is the best choice for handling test problems (Π_4, Π_5, and Π_6) in terms of the elapsed CPU time. At the same time, the sHessen(m) method is still more efficient than both sQMRIDR(1) and sBiCGSTAB(2) method for the last three test problems (except the sQMRIDR(1) method for the test problem Π_6). Moreover, it is worth mentioning that these three short-term recurrence shifted Krylov subspace solvers require less number of MVPs than those corresponding to both the sHessen(m) method and the sFOM(m), whereas the last two shifted Krylov subspace solvers still can save the elapsed CPU time. This is due to that the other short-term vector recurrence shifted Krylov solvers need more extra inner products and n-vectors updated, which are often highly time-consuming [19,20,22]. In conclusion, we can mention that the sHessen(40) method can be still considered as a useful alternative for handling the sequence of shifted linear systems (1.1) in Example 2.

Example 3 (*Two Dimensional (2D) Heat Equation [48]*)**.** In this application about evaluating the action of a matrix exponential on a vector, we consider the fourth-order spatial semi-discretization of the following 2D heat equation

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{1}{2\pi^2}\left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2}\right), \\ u(x, y, 0) = u_0(x, y) = \sin(\pi x)\sin(\pi y) \end{cases} \tag{4.1}$$

is employed to model many applications in geo-engineering. A finite difference discretization with $N \times N$ points in the domain $\Omega = [0, 1]^2$ results in a sparse discretized matrix with a more complex structure, i.e.

$$\begin{cases} A_x \dfrac{d\boldsymbol{u}(t)}{dt} = B_x \boldsymbol{u}(t), \qquad t \in [0, T] \\ \boldsymbol{u}(0) = \boldsymbol{u}_0. \end{cases} \tag{4.2}$$

It notes that more detailed forms of two real matrices A_x and B_x can be found in [48]. Then we follow the idea proposed in [5], to compute the matrix exponential multiplying a vector $\boldsymbol{u}_0$, which is discretized from the initial condition $u_0(x, y)$. For the

Table 5
Compared results about different shifted Krylov subspace solvers for Example 3 in aspects of CPU and MVPs ($m = 40$ and $t = T = 1$).

$h_x = h_y$	sHessen(m)		sFOM(m)		wsFOM(m)		sIDR(1)		sQMRIDR(1)		sBiCGSTAB(2)	
	MVPs	CPU	MVPs	CPU	MVPs	CPU	MVPs	CPU	MVPs	CPU	MVPs	CPU
1/85	480	2.331	320	2.583	‡	‡	304	6.759	485	4.950	264	9.215
1/90	480	2.729	400	3.689	‡	‡	290	6.855	583	6.341	280	9.830
1/95	440	2.926	440	4.567	‡	‡	360	9.063	643	8.115	292	12.623
1/100	440	3.403	480	5.752	‡	‡	360	9.553	561	8.044	296	13.709
1/105	560	5.069	520	7.348	‡	‡	428	11.967	605	9.760	312	17.053
1/110	520	5.269	560	8.593	‡	‡	450	13.388	610	10.931	332	19.864
1/120	520	6.766	680	13.266	‡	‡	417	13.839	617	14.045	332	21.083

quadrature nodes of rational approximation of the matrix exponential operator, we choose $\nu = 16$ quadrature nodes for the action of a matrix exponential on a vector (i.e., $\exp(tA_x^{-1}B_x)\boldsymbol{u}_0$), we require to solve a sequence of shifted linear systems

$$(z_j I - A_x^{-1}B_x)\boldsymbol{x} = \boldsymbol{u}_0, \quad z_j \in \mathbb{C}, \quad j = 1, 2, \ldots, \nu.$$

We choose the first one $(z_1 I - A_x^{-1}B_x)\boldsymbol{x}^{(1)} = \boldsymbol{u}_0$ as the seed system. More details and choosing $\nu = 16$ complex shifts can be found in [5] and references therein.

According to numerical results listed in Table 5, the wsFOM(m) again cannot solve any test problems at all. From the CPU time perspective, the proposed sHessen(m) method outperforms the sFOM(m), even when the former one needs more number of MVPs than the later one (refer to the results of $h_x = h_y = 1/100, 1/110, 1/120$). It is in line with the cost analysis that the required number of MVPs are similar, the algorithmic cost of the sHessen(m) method can be lower than that of the sFOM(m). Furthermore, the sHessen(m) method is also more competitive than other short-term recurrence shifted Krylov subspace methods for test problems in aspects of the elapsed CPU time. Especially, the sQMRIDR(1) method even needs more number of MVPs than those required by sHessen(m). Similarly, we should mention that although both sIDR(1) and sBiCGSTAB(2) methods always require the less number of MVPs than those needed by sHessen(m) or sFOM(m), the former two shifted iterative solvers are still more expensive than the last two shifted iterative methods in aspects of the elapsed CPU time. This is because both sIDR(1) and sBiCGSTAB(2) methods require more extra inner products and n-vectors updated, which are not always cheap in terms of the elapsed CPU time [19,20,22]. In summary, the proposed sHessen(m) method is the best solvers among these mentioned shifted Krylov subspace methods for solving the sequence of shifted linear systems in Example 3.

Example 4. Applications of fractional differential equations (FDEs) have been found in physical, biological, geological and financial systems, and in the recent years there are intensive studies on them, refer, e.g., to [49,50] for this topic. Here we consider the benchmark problem coming from the 3D time-fractional convection–diffusion–reaction equation, namely

$$\begin{cases} \dfrac{\partial^\gamma u}{\partial t^\gamma} - \epsilon \triangle u + \vec{\beta} \cdot \nabla u - ru = 0, & (x, y, z) \in \Omega = (0, 1)^3, \ t \in [0, T], \\ u(x, y, z, t) = 0, & (x, y, z) \in \partial\Omega, \ t \in [0, T], \\ u(x, y, z, 0) = x(1-x)y(1-y)z(1-z), & (x, y, z) \in \bar{\Omega}. \end{cases} \tag{4.3}$$

This example can be viewed as a modification of the third example in [20]. The physical parameters are chosen as follows: $\epsilon = 1$ (diffusion), $\vec{\beta} = (0/\sqrt{5}, 250/\sqrt{5}, 500/\sqrt{5})^T$ (convection), and r (reaction). In order to solve Eq. (4.3) numerically, we start by discretizing the spatial domain into uniformly spaced grid points. Then the finite difference approximation with the natural ordering results in a system of FDEs as following form

$$\frac{d^\gamma \boldsymbol{u}}{dt^\gamma} = -A\boldsymbol{u}(t), \qquad \boldsymbol{u}(0) = \boldsymbol{u}_0. \tag{4.4}$$

Since the spatial finite difference methods for (4.3) lead to the system of FDEs with the form (4.4), where $\boldsymbol{u}$ is the vector containing the unknown solution, it is a well-known result [5,6] that, for $0 < \gamma < 1$, the exact solution of this problem (4.4) can be expressed as

$$\boldsymbol{u}(t) = e_{\gamma,1}(t; -A)\boldsymbol{u}_0, \quad \text{and} \quad e_{\gamma,1}(t; -A) = t^{1-1}E_{\gamma,1}(-t^\gamma A) = E_{\gamma,1}(-t^\gamma A), \tag{4.5}$$

where $E_{\gamma,1}(z)$ is the Mittag-Leffler (ML) function [6,50]

$$E_{\gamma,1}(z) := \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\gamma k + 1)}, \quad \gamma > 0, \ z \in \mathbb{C}.$$

In light of (4.5), to compute the solution $\boldsymbol{u}(t)$, we have to approximate the product of the matrix ML function $E_{\gamma,1}(-t^\gamma A)$ with the vector $\boldsymbol{u}_0$, which is the major computational cost for this problem. The numerical evaluation of matrix functions $E_{\gamma,1}(-t^\gamma A)\boldsymbol{u}_0$ has recently gained new interest, as shown by the recent spread of literature [5,6] in this field. Moreover, these

Table 6
Set and characteristics of test problems in Example 1 (listed in increasing matrix size).

Index	Grid size	Size	$nnz(A)$	Reaction	λ	ν
Ξ_1	$h = 0.04$	13,824	93,312	$r = 400$	0.8	10
Ξ_2	$h = 0.04$	13,824	93,312	$r = 600$	0.9	12
Ξ_3	$h = 0.025$	59,319	406,107	$r = 400$	0.6	10
Ξ_4	$h = 0.025$	59,319	406,107	$r = 600$	0.8	10
Ξ_5	$h = 0.02$	117,649	809,137	$r = 400$	0.8	10
Ξ_6	$h = 0.02$	117,649	809,137	$r = 600$	0.9	12

Table 7
Compared results about different shifted Krylov subspace solvers for Example 4 in terms of the MVPs and CPU ($t = 1, m = 30$).

Index	sHessen(m)		sFOM(m)		wsFOM(m)		sIDR(1)		sQMRIDR(1)		sBiCGSTAB(2)	
	MVPs	CPU	MVPs	CPU	MVPs	CPU	MVPs	CPU	MVPs	CPU	MVPs	CPU
Ξ_1	270	0.296	300	0.467	‡	‡	‡	‡	‡	‡	144	1.206
Ξ_2	300	0.389	390	0.680	‡	‡	‡	‡	‡	‡	156	1.522
Ξ_3	330	1.788	300	2.245	‡	‡	‡	‡	341	7.885	204	9.658
Ξ_4	360	1.989	360	2.717	‡	‡	‡	‡	333	7.505	232	10.298
Ξ_5	330	3.657	420	6.482	‡	‡	‡	‡	289	17.768	252	23.219
Ξ_6	360	4.316	450	7.213	‡	‡	‡	‡	301	20.499	276	27.255

numerical evaluation methods based on the Carathéodory–Fejér approximation [51] for $E_{\gamma,1}(t^\gamma A)v$ (we set $t = 1$) can be represented as

$$E_{\gamma,1}(-A)\boldsymbol{u}_0 = f_\nu(-A)\boldsymbol{u}_0 = \sum_{j=1}^{\nu} w_j (z_j I + A)^{-1}\boldsymbol{u}_0, \quad j = 1, 2, \ldots, \nu, \tag{4.6}$$

where w_j and z_j are quadrature weights and nodes, respectively. So in the implementation of the procedure (4.6), it requires to solve a sequence of shifted linear systems, which are similar to $(A + z_j I)\boldsymbol{x}^{(j)} = \boldsymbol{u}_0,\ z_j \in \mathbb{C}$. For simplicity, here we summarize the information about our different test problems in Table 6. The linear system $A\boldsymbol{x} = \boldsymbol{u}_0$ was treated as the *seed* system. Under this condition, it is remarked that the procedure of establishing the Krylov subspace for shifted linear systems does not involve the complex operations. Then the results about convergence performance of different shifted Krylov subspace methods are listed in Table 7.

As observed from Table 7, the sHessen(30) method outperforms both wsFOM(30) and sFOM(30) in aspects of MVPs for different test problems except Ξ_3. Moreover, we find that the sHessen(30) method requires the least amount of elapsed CPU time than other five shifted iterative solvers. It also finds that both wsFOM(30) and sIDR(1) fail to solve the sequence of shifted linear systems in Example 4, even the sQMRIDR(1) method also fails to handle the first two test problems (i.e., Ξ_1 and Ξ_2). That is because both sIDR(1) and sQMRIDR(1) methods may occur the serious break-down due to their typical short-term vector recurrence iterations for handling these test problems. Again, the performance of wsFOM(30) is not always promising. On the other hand, although the number of matrix–vector products required by both sQMRIDR(1) (except both Ξ_1 and Ξ_2 problems) and sBiCGSTAB(2) methods is less than those needed by both the sHessen(30) method and the sFOM(30), these latter two solvers are still cheaper in aspects of the elapsed CPU time. This is due to that the first two iterative methods may require more number of inner products and n-vectors updated [19,20,22], which also require to consume the extra elapsed CPU time. In conclusion, the proposed sHessen(30) method can be viewed as an efficient alternative for solving the sequence of shifted linear systems in Example 4.

5. Conclusions

Based on the above experimental results, we conclude that our proposed algorithm – the restarted shifted Hessenberg method – indeed can show considerably attractive convergence performance with respect to the elapsed CPU time compared to the restarted shifted FOM proposed in [30], the restarted weighted shifted FOM introduced in [15] and some popular shifted Krylov subspace solvers based on the short-term vector recurrence. Moreover, in some cases where sHessen(m) requires less enough number of MVPs to converge, this is a reason that this algorithm can significatively reduce the CPU consuming time. At the same time, since the Hessenberg process often requires slightly less computational storage [33,34,42] than the conventional Arnoldi process, so the sHessen(m) method seems to be preferable to the other Arnoldi-based shifted iterative solvers (sFOM(m) and wsFOM(m)) especially if the number of restarts of these three shifted iterative solvers is similar; see the analysis of computational cost described in Section 3. Additionally, it finds that different seed systems are chosen before we exploit the shifted Krylov subspace solvers for different test problems in our experiments. That is mainly due to that it is usually hard to make an optimal choice of the seed system in advance, the so-called *seed switching technique* introduced in [12] may be an option for remedying this difficulty.

At the same time, it has to mention that the body of theoretical evidence is still unavailable recently for the result that the sHessen(m) method can enjoy advantages over sFOM(m) in terms of convergence analyses. The computational

efficiency of sHessen(m) in respect of the restarting number and the elapsed CPU time is just illustrated on a set of problems arising both from extensive academic and from industrial applications. Furthermore, theoretical convergence analysis should remain a meaningful topic of further research. In addition, as earlier mentioned [2,9,14,20], the efficient preconditioning technique for the Krylov subspace methods to solve the shifted linear systems (1.1) is still a very difficult problem due to remaining the shift-invariant property (1.3) of preconditioned systems. Therefore it is considerably important that in future work we will investigate the convergence performance of the sHessen(m) method with suitable preconditioners, which can remain shift-invariance property (1.3) for preconditioned systems. In fact, our coming work [52] has investigated the (unrestarted) sHessen method as an efficient inner solvers of nested Krylov subspace solvers for shifted linear systems, which were studied by Baumann and van Gijzen [9]. Numerical results demonstrate that in the framework of nested Krylov subspace solvers, using the (unrestarted) sHessen method as an inner solver is often cheaper than using the (unrestarted) sFOM as an inner solver in terms of the elapsed CPU time. Therefore, we believe that our proposed sHessen(m) method can be employed as a meaningful and useful alternative for solving the shifted linear systems.

Acknowledgments

The authors are grateful to Dr. Ke Zhang and Dr. Jing Meng for their constructive discussions and insightful comments. Moreover, the first author also would like to thank Prof. J.A.C. Weideman and Dr. Roberto Garrappa for suggesting them to choose the practical complex shifts in Example 3–Example 4, respectively. Meanwhile, the authors are very grateful to the anonymous referees and editor Prof. Michael Ng for their useful suggestions and comments that improved the presentation of this paper. In particular, we need to thank Dr. Jens-Peter M. Zemke for making his codes of the multi-shifted QMRIDR(s) method [20] available for us. This research is supported by NSFC (61370147, 11501085, and 61402082) and the Fundamental Research Funds for the Central Universities (ZYGX2016J132 and ZYGX2016J138).

References

[1] B.N. Datta, Y. Saad, Arnoldi methods for large Sylvester-like observer matrix equations, and an associated algorithm for partial spectrum assignment, Linear Algebra Appl. 154–156 (1991) 225–244.
[2] M.I. Ahmad, D.B. Szyld, M.B. van Gijzen, Preconditioned multishift BiCG for $\mathcal{H}_2$-optimal model reduction, SIAM J. Matrix Anal. Appl. 38 (2017) 401–424.
[3] A. Feriani, F. Perotti, V. Simoncini, Iterative system solvers for the frequency analysis of linear mechanical systems, Comput. Methods Appl. Mech. Engrg. 190 (2000) 1719–1739.
[4] T. Sakurai, H. Sugiura, A projection method for generalized eigenvalue problems using numerical integration, J. Comput. Appl. Math. 159 (2003) 119–128.
[5] J.A.C. Weideman, L.N. Trefethen, Parabolic and hyperbolic contours for computing the Bromwich integral, Math. Comp. 76 (2007) 1341–1356.
[6] R. Garrappa, M. Popolizio, On the use of matrix functions for fractional partial differential equations, Math. Comput. Simulation 81 (2011) 1045–1056.
[7] J.C.R. Bloch, A. Frommer, B. Lang, T. Wettig, An iterative method to compute the sign function of a non-Hermitian matrix and its application to the overlap Dirac operator at nonzero chemical potential, Comput. Phys. Comm. 177 (2007) 933–943.
[8] T.F. Chan, M.K. Ng, Galerkin projection methods for solving multiple linear systems, SIAM J. Sci. Comput. 21 (1999) 836–850.
[9] M. Baumann, M.B. van Gijzen, Nested krylov methods for shifted linear systems, SIAM J. Sci. Comput. 37 (2015) S90–S112.
[10] G. Wu, Y.-C. Wang, X.-Q. Jin, A preconditioned and shifted GMRES algorithm for the PageRank problem with multiple damping factors, SIAM J. Sci. Comput. 34 (2012) A2558–A2575.
[11] A.K. Saibaba, T. Bakhos, P.K. Kitanidis, A flexible Krylov solver for shifted systems with application to oscillatory hydraulic tomography, SIAM J. Sci. Comput. 35 (2013) A3001–A3023.
[12] T. Sogabe, T. Hoshi, S.-L. Zhang, T. Fujiwara, A numerical method for calculating the green's function arising from electronic structure theory, in: Y. Kaneda, H. Kawamura, M. Sasai (Eds.), Frontiers of Computational Science, Springer-Verlag, Berlin/Heidelberg, 2007, pp. 189–195.
[13] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., SIAM, Philadelphia, PA, 2003.
[14] B. Jegerlehner, Krylov space solvers for shifted linear systems, arXiv preprint, hep-lat/9612014, 15 December 1996, 16 pages. Available online at http://arxiv.org/abs/hep-lat/9612014.
[15] Y.-F. Jing, T.-Z. Huang, Restarted weighted full orthogonalization method for shifted linear systems, Comput. Math. Appl. 57 (2009) 1583–1591.
[16] J.-F. Yin, G.-J. Yin, Restarted full orthogonalization method with deflation for shifted linear systems, Numer. Math. Theor. Meth. Appl. 7 (2014) 399–412.
[17] R.W. Freund, Solution of shifted linear systems by quasi-minimal residual iterations, in: L. Reichel, A. Ruttan, R.S. Varga (Eds.), Numerical Linear Algebra, de Gruyter, Berlin, Germany, 1993, pp. 101–121.
[18] S. Kirchner, IDR-Verfahren Zur LÖSung Von Familien Geshifteter Linearer Gleichungssysteme, Diplomarbeit, Fachbereich Mathematik, Bergische Universität Wuppertal, Germany, 2011 (in German).
[19] L. Du, T. Sogabe, S.-L. Zhang, IDR(s) for solving shifted nonsymmetric linear systems, J. Comput. Appl. Math. 274 (2015) 35–43.
[20] M.B. van Gijzen, G.L.G. Sleijpen, J.-P.M. Zemke, Flexible and multi-shift induced dimension reduction algorithms for solving large sparse linear systems, Numer. Linear Algebra Appl. 22 (2015) 1–25.
[21] T. Sogabe, Extensions of the Conjugate Residual Method (Ph.D Dissertation), Department of Applied Physics, University of Tokyo, Japan, 2006. Available online at http://www.ist.aichi-pu.ac.jp/person/sogabe/thesis.pdf.
[22] A. Frommer, BiCGStab(ℓ) for families of shifted linear systems, Computing 70 (2003) 87–109.
[23] X.-M. Gu, T.-Z. Huang, J. Meng, T. Sogabe, H.-B. Li, L. Li, BiCR-type methods for families of shifted linear systems, Comput. Math. Appl. 68 (2014) 746–758.
[24] M. Dehghan, R. Mohammadi-Arani, Generalized product-type methods based on bi-conjugate gradient (GPBiCG) for solving shifted linear systems, Comput. Appl. Math. (2016) 16 pages, in press. DOI: 10.1007/s40314-016-0315-y.
[25] K. Ahuja, E. de Sturler, S. Gugercin, E.R. Chang, Recycling BiCG with an application to model reduction, SIAM J. Sci. Comput. 34 (2012) A1925–A1949.
[26] A. Frommer, U. Glässner, Restarted GMRES for shifted linear systems, SIAM J. Sci. Comput. 19 (1998) 15–26.
[27] G. Gu, Restarted GMRES augmented with harmonic Ritz vectors for shifted linear systems, Int. J. Comput. Math. 82 (2005) 837–849.
[28] K.M. Soodhalter, D.B. Szyld, F. Xue, Krylov subspace recycling for sequences of shifted linear systems, Appl. Numer. Math. 81 (2014) 105–118.
[29] D. Darnell, R.B. Morgan, W. Wilcox, Deflated GMRES for systems with multiple shifts and multiple right-hand sides, Linear Algebra Appl. 429 (2008) 2415–2434.
[30] V. Simoncini, Restarted full orthogonalization method for shifted linear systems, BIT 43 (2003) 459–466.

[31] R.B. Morgan, A restarted GMRES method augmented with eigenvectors, SIAM J. Matrix Anal. Appl. 16 (1995) 1154–1171.
[32] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, UK, 1965.
[33] H. Sadok, CMRH: A new method for solving nonsymmetric linear systems based on the Hessenberg reduction algorithm, Numer. Alg. 20 (1999) 303–321.
[34] M. Heyouni, H. Sadok, A new implementation of the CMRH method for solving dense linear systems, J. Comput. Appl. Math. 213 (2008) 387–399.
[35] M. Heyouni, Méthode De Hessenberg GénéRalisée et Applications (Ph.D thesis), Université des Sciences et Technologies de Lille, France, 1996. Available online at http://ori.univ-lille1.fr/notice/view/univ-lille1-ori-134864.
[36] H. Sadok, Méthodes de projections pour les systèmes linéaires et non linéaires (Habilitation thesis), University of Lille1, Lille, France, 1994.
[37] H. Sadok, D.B. Szyld, A new look at CMRH and its relation to GMRES, BIT 52 (2012) 485–501.
[38] J. Duintjer Tebbens, G. Meurant, On the convergence of Q-OR and Q-MR Krylov methods for solving nonsymmetric linear systems, BIT 56 (2016) 77–97.
[39] K. Zhang, C. Gu, A flexible CMRH algorithm for nonsymmetric linear systems, J. Appl. Math. Comput. 45 (2014) 43–61.
[40] M. Heyouni, H. Sadok, On a variable smoothing procedure for Krylov subspace methods, Linear Algebra Appl. 268 (1998) 131–149.
[41] K. Hessenberg, BehandLung Linearer Eigenwertaufgaben Mit Hilfe Der Hamilton-Cayleyschen Gleichung, Numerische Verfahren, Bericht 1, Institut für Praktische Mathematik (IPM), Technische Hochschule Darmstadt, 1940 The scanned report and a biographical sketch of Karl Hessenberg's life are available at http://www.hessenberg.de/karl1.html.
[42] D. Stephens, ELMRES: An Oblique Projection Method To Solve Sparse Non-Symmetric Linear Systems (Ph.D dissertation), Florida Institute of Technology, Melbourne, USA, 1999. Available online at http://ncsu.edu/hpc/Documents/Publications/gary_howell/stephens.pdf.
[43] G. Howell, D. Stephens, ELMRES, an Oblique Projection Method for Solving Systems of Sparse Linear Equations, Technical Report, Florida Institute of Technology, 2000, 18 pages. Available online at http://ncsu.edu/hpc/Documents/Publications/gary_howell/elmres320ps.
[44] N.J. Higham, Accuracy and Stability of Numerical Algorithms, second ed., SIAM, Philadelphia, PA, 2002.
[45] M. Schweitzer, Any finite convergence curve is possible in the initial iterations of restarted FOM, Electron. Trans. Numer. Anal. 45 (2016) 133–145.
[46] G.H. Golub, C.F. Van Loan, Matrix Computations, fourth ed., Johns Hopkins University Press, Baltimore, MD, 2013.
[47] T. Davis, Y. Hu, The university of florida sparse matrix collection, ACM Trans. Math. Software 38 (1) (2011) Article 1, 25 pages. Available online at http://www.cise.ufl.edu/research/sparse/matrices/.
[48] H.-W. Sun, W. Wang, A fourth order compact BVM scheme for the two dimensional heat equations, in: H.R. Arabnia (Ed.), Proceedings of the 2008 International Conference on Scientific Computing, Las Vegas, Nevada, USA, 2008, pp. 310–314.
[49] S. Zhai, D. Gui, P. Huang, X. Feng, A novel high-order ADI method for 3D fractional convection–diffusion equations, Int. Commun. Heat Mass Transfer 66 (2015) 212–217.
[50] I. Podlubny, Fractional Differential Equations, Academic Press, San Diego, CA, 1999.
[51] L.N. Trefethen, J.A.C. Weideman, T. Schmelzer, Talbot quadratures and rational approximations, BIT 46 (2006) 653–670.
[52] X.-M. Gu, T.-Z. Huang, B. Carpentieri, A. Imakura, K. Zhang, L. Du, Variants of the CMRH Method for Solving Multi-Shifted Non-Hermitian Linear Systems, Technical Report, University of Groningen, Groningen, the Netherlands, 2016, 34 pages. Also available online at https://arxiv.org/abs/161100288.