Restarted Hessenberg method for shifted nonsymmetric linear systems with applications to fractional differential equations

Gu, Xian-Ming; Huang, Ting-Zhu; Yin, Guojian; Carpentieri, Bruno; Wen, Chun

Restarted Hessenberg method for shifted nonsymmetric linear systems with applications to fractional differential equations

Xian-Ming Gu[a,b*], Ting-Zhu Huang[a†], Guojian Yin[c‡],
Bruno Carpentieri[b§], Chun Wen[a¶]

a. School of Mathematical Sciences,

University of Electronic Science and Technology of China, Chengdu, 611731, P.R. China

b. Institute of Mathematics and Computing Science,

University of Groningen, Nijenborgh 9, P.O. Box 407, 9700 AK Groningen, The Netherlands

c. Shenzhen Institutes of Advanced Technology,

Chinese Academy of Science, Guangdong, Shenzhen, 518055, P.R. China

Abstract

It is known that the restarted full orthogonalization method (FOM) outperforms the restarted generalized minimum residual method (GMRES) in several circumstances for solving shifted linear systems when the shifts are handled simultaneously. Variants of them have been proposed to enhance their performance. We show that another restarted method, restarted Hessenberg method [M. Heyouni, Méthode de Hessenberg Généralisée et Applications, Ph.D. Thesis, Université des Sciences et Technologies de Lille, France, 1996] based on Hessenberg process, 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 reduction method is still collinear with each other. In these cases where our proposed algorithm needs less enough number of restarts to converge than the earlier established restarted shifted FOM and weighted restarted shifted FOM, the associated CPU consuming time is also considerably reduced, as shown via extensive numerical experiments involving the recent popular applications of handling structural dynamics, time-fractional convection-diffusion equations and space-fractional diffusion equations.

Key words: Shifted linear system; Hessenberg process; Pivoting strategy; Restarted Hessenberg method; Collinear; Fractional differential equations.

AMSC (2010): 65F10, 65F15, 15A06, 15A18

1 Introduction

Considered a real large sparse $n \times n$ 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 the shifted nonsingular linear systems

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

where I denotes the $n \times n$ identity matrix. Such shifted systems arise in many scientific and engineering fields, such as control theory [1, 2], structural dynamics [3, 4], eigenvalues computation [5, 6], numerical solutions of time-dependent partial/fractional differential equations [7–10], QCD problems [11, 12] and

*E-mail: guxianming@live.cn, x.m.gu@rug.nl

†Corresponding author. E-mail: tingzhuhuang@126.com. Tel: +86 28 61831608, Fax: +86 28 61831280.

‡E-mail: guojianyin@gmail.com

§E-mail: bcarpentieri@gmail.com

¶E-mail: wchun17@163.com

other simulation problems [13–18]. Among all the systems, when $\sigma_i = 0$, the system $A\boldsymbol{x} = \boldsymbol{b}$ is often treated as the seed system.

It is well known that Krylov subspace methods are widely used for the solution of linear systems (see e.g. [19]). Denoting the k-dimensional Krylov subspace with respect to A and $\boldsymbol{b}$ by

$$\mathcal{K}_k(A, \boldsymbol{v}) := \text{span}\{\boldsymbol{v}, A\boldsymbol{v}, \ldots, A^{k-1}\boldsymbol{v}\}, \tag{2}$$

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

$$\mathcal{K}_k(A, \boldsymbol{v}) := \mathcal{K}_k(A - \sigma_i I, \boldsymbol{v}), \qquad i = 1, 2, \ldots, \nu, \tag{3}$$

which shows that the iterations of (1) are the same Krylov subspace as the iterations of seed systems. This implies that if we choose initial vectors $\boldsymbol{x}_0$ properly (for example, all initial guesses are zero), once a basis has been generated for one of these linear systems, it could also be used for all other linear systems. Therefore, if we employ a Krylov subspace method to solve (1) simultaneously, a certain amount of computational efficiency can be maintained if the Krylov subspace is the same for all the shifted systems each time. This happens when the generating vectors are collinear, for the basis and the square Hessenberg matrix are needed to be evaluated only once; refer to e.g. [14, 16, 17, 20–22].

Several techniques have been proposed in the past few years that attempt to tackle this kind of linear systems (1). For shifted nonsymmetric (non-Hermitian) linear systems, iterative methods such as the shifted BiCGStab(ℓ) [23], the shifted (TF)QMR [4,24], shifted QMRIDR(s) [25] and shifted IDR(s) [26,27] methods have been developed. Shifted algorithms of short-term recurrence methods of CG [28], CR [29], BiCG [30], BiCR [31], BiCGStab [32] and BiCRStab [33] were discussed in [21,34]. For the case of shifted (complex) symmetric linear systems, the shifted COCG and shifted COCR methods have been proposed in [18,35], respectively. The shifted QMR_SYM(B) method with the weighted quasi-residual minimization strategy was introduced in [36]. Although the above-mentioned short-term recurrence methods have been successfully generalized to solve the shifted linear systems (1) from many practical applications, most of them are not numerically stable and often happen to (serious) break-down if the look-ahead techniques are not exploited, see [37] for instance.

On the other hand, the restarted GMRES-type methods are widely known and appreciated to be effective on (1), see [4, 38–42] 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 to guarantee the computed GMRES shifted residuals collinear to each other in order to maintain the computational efficiency; see e.g. [4, 38]. 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 GMRES applied to those systems, refer to [4, 22, 38]. In contrast, it is more natural and more effective for restarted FOM to be applied to shifted linear systems simultaneously handled, for all residuals are naturally collinear [16, 43, 44]. 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 [20] further accelerated this method by introducing a weighted norm. In 2014, Yin and Yin have studied restarted FOM with the deflation technique [46] 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 the Ritz vectors from the last cycle, see [22] for details.

However, as we know, both the restarted GMRES and FOM methods for shifted linear systems are derived by using the Arnoldi procedure [19, pp. 160-165], which turns to be expensive when m (dimension of the Krylov subspace) becomes large because of the growth of memory and computational requirements as m increases. So it is still meaningful for finding some cheaper iterative solvers for shifted linear systems (1). Here, we consider to use the Hessenberg reduction process [47–50] because it generally requires less operations and storage than Arnoldi process and is thus favorable for generating a linear system solver. Moreover, it has been proved that we can establish two families of Krylov subspace methods, namely Hessenberg method [50] and CMRH method [47, 48, 50, 51], by using the basic principles behind the (restarted) FOM method and GMRES method, respectively, refer to [50, 52] for this discussion. Some

recent developments concerning the CMRH method, which is very similar to the GMRES method, and the Hessenberg process can be found in [47,51,53,54]. Since the restarted FOM method is built via combining the Arnoldi process and Galerkin-projection idea [19, pp. 165-168], so it is natural to extend the restarted FOM method for solving shifted linear systems (1), refer to [43]. 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 restarted FOM method for shifted linear systems gave us a simple and natural problem: Does there exist a variant of the original restarted Hessenberg method for shifted linear systems? Our major contribution is to answer this question. The answer is yes, and it requires a similar but efficient idea from that used in restarted shifted FOM method. 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 CPU time elapsed, which will be shown by the numerical experiments in Section 4. Moreover, our algorithm is able to solve certain shifted systems which both Restarted Shifted FOM and Restarted Weighted Shifted FOM methods 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 discuss the naturally collinear property of residual during each cycle of the restarted Hessenberg method. Then, we show how to generalize the restarted shifted Hessenberg method for solving the shifted linear systems. Implementation details will be 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 restarted Hessenberg method

In order to extend the restarted Hessenberg method for solving the shifted linear systems well, we should recall the restarted Hessenberg method, which is established from the Hessenberg process. According to Refs. [50, 53], it is not hard to conclude that the restarted Hessenberg method is greatly close to the restarted FOM method, which is derived from the well-know Arnoldi process. Although the restarted Hessenberg method has been proved to be cheaper than the restarted FOM method, the restarted Hessenberg method is often not popular in the field of Krylov subspace methods. However, we want to revive the restarted Hessenberg method to solve the shifted linear systems (1) here.

2.1 The Hessenberg process

Starting point of the algorithms derived in this paper is the Hessenberg process for reducing the nonsymmetric matrix to a Hessenberg decomposition [47,49]. In [55], the Hessenberg process is 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 [49]. For ease of notation we will assume that 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.

Let $\boldsymbol{v}$ be a column vector of $\mathbb{R}^n$ and A an $n \times n$ real matrix. The Hessenberg reduction process (without pivoting strategy) computes a unit trapezoidal matrix $L_m = [\boldsymbol{l}_1, \ldots, \boldsymbol{l}_m]$ whose columns form a basis of the Krylov subspace $\mathcal{K}_m(A, \boldsymbol{v})$ by using the following formulas:

$$\begin{cases} \beta = (\boldsymbol{v})_1, & \boldsymbol{l}_1 = \boldsymbol{v}/\beta, \\ h_{k+1,k}\boldsymbol{l}_{k+1} = A\boldsymbol{l}_k - \sum_{j=1}^{k} h_{j,k}\boldsymbol{l}_j, & \text{for } k = 1, 2, \ldots, m. \end{cases} \tag{4}$$

The parameters $h_{j,k}$ are determined such that

$$\boldsymbol{l}_{k+1} \perp \boldsymbol{e}_1, \ldots, \boldsymbol{e}_k \quad \text{and} \quad (\boldsymbol{l}_{k+1})_{k+1} = 1. \tag{5}$$

Assume that $\{\boldsymbol{l}_i\}_{i=1,\ldots,k}$ have been computed such that the $i-1$ first components of $\boldsymbol{l}_i$ equal zero and the i-th component equals one. To obtain $\boldsymbol{l}_{k+1}$, we first compute $\boldsymbol{u} = A\boldsymbol{l}_k$ and then we subtract multiples of $\boldsymbol{l}_1, \ldots, \boldsymbol{l}_k$ in order to annihilate the components $1, \ldots, k$ of the $\boldsymbol{u}$ to obtain the $\boldsymbol{w} = A\boldsymbol{l}_k - \sum_{i=1}^{k} h_{i,k}\boldsymbol{l}_i$. Finally we choose $h_{k+1,k} = (\boldsymbol{w})_{k+1}$ and take $\boldsymbol{l}_{k+1} = \boldsymbol{w}/h_{k+1,k}$.

In addition, it is important that the entry $h_{k+1,k}$ never becomes zero. If this occurs-such situation is called a breakdown-the Hessenberg process cannot proceed. In addition, small values of $h_{k+1,k}$ can cause severe loss of accuracy. To avoid such a breakdown and also ensure numerical stability, the process can be modified to include a pivoting strategy such as in Gaussian elimination method. This is done by replacing the orthogonality condition (5) with the following one:

$$\boldsymbol{l}_{k+1} \perp \boldsymbol{e}_{p_1}, \ldots, \boldsymbol{e}_{p_k} \quad \text{and} \quad (\boldsymbol{l}_{k+1})_{p_{k+1}} = 1, \tag{6}$$

where $p_j \in \{1, 2, \ldots, n\}$. To compute p_{k+1}, we follow the practical procedure described in [49]. We suppose that $p_1, \ldots, p_k$ have already been obtained, then we compute $\boldsymbol{u} = A\boldsymbol{l}_k$ and subtract multiples of $\boldsymbol{l}_1, \ldots, \boldsymbol{l}_k$ in order to annihilate the k components $p_1, \ldots, p_k$ of the vector $\boldsymbol{u}$ to obtain a vector $\boldsymbol{w} = A\boldsymbol{l}_k - \sum_{i=1}^{k} h_{i,k}\boldsymbol{l}_i$. Then we set $p_{k+1} = i_0$, where i_0 satisfies $\|\boldsymbol{w}\|_\infty = |(\boldsymbol{w})_{i_0}|$. Finally we normalize the vector $\boldsymbol{l}_{k+1}$ by taking $h_{k+1,k} = (\boldsymbol{w})_{i_0}$ and $\boldsymbol{l}_{k+1} = \boldsymbol{w}/(\boldsymbol{w})_{i_0}$.

Notice that if $\|\boldsymbol{w}\|_\infty = 0$ at step k, then, in exact arithmetic, the minimal polynomial with respect to the vector $\boldsymbol{v}$ has the degree k which means that we have constructed an invariant subspace and the process must be stopped. Using the pivoting strategy described above, the Hessenberg process is reproduced in Algorithm 1.

Algorithm 1 The Hessenberg procedure with pivoting strategy

1: $\boldsymbol{p} = [1, 2, \ldots, n]^T$ 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 $i_0 \in \{j+1, ..., n\}$ 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**

Letting 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

$$\begin{aligned} AL_k &= L_{k+1}\bar{H}_k, \\ &= L_k H_k + h_{k+1,k}\boldsymbol{l}_{k+1}\boldsymbol{e}_k^T \end{aligned} \tag{7}$$

and $\mathcal{P}_k L_k$ 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 [47, 48] for details. In [47], it was interesting to note that Heyouni and

Sadok 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 details.

At the end of this subsection, it is remarkable to mention that the earlier work[1] of Howell and Stephens [56] and Stephens's Ph.D. dissertation [57] have really made some further progress on the backward error analysis for Hessenberg process. They had obtained the following theorem, which can be regarded as a slight improvement on Wilikinson's results [49]. For a proof, one can consult Stephens's Ph.D. dissertation [57].

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 $|a_{ij}|$.*

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

where $\gamma_n = n\epsilon/(1-n\epsilon)$ and ϵ is the smallest number 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 is not backward stable in finite precision arithmetic. However, in practice for most of the test problems that we considered in our study, it showed to be numerically accurate. We give an insight on the quality of the Krylov basis computed by Algorithm 1 by plotting in Fig. 1 the four smallest singular values of the matrices L_k and V_k generated after k steps of the Hessenberg and the Arnoldi process, respectively, for different values of k. We use a normalized random vector as initial vector for both algorithms. A singular value close to zero indicates near-singularity, and consequently loss of linear dependence of the basis vectors. The experiments refer to four test matrices extracted from Tim Davis's matrix collection at the University of Florida [58]. As we can see, for low to moderate values of k, the Krylov basis computed by the Hessenberg process maintain linear independence very well as four smallest singular values of L_k are far from zero. The Arnoldi process ensures in addition vectors orthogonality for the first three test matrices, but this comes at the price of higher computational cost. Moreover, for the last test problem, when the k increases, the Arnoldi algorithm starts to loose linear independence of the basis, in contrast to the Hessenberg method.

It is well known that the convergence of Ritz values plays an important role in the rate of convergence of Krylov methods (see e.g. the analyses presented in [59] for the CG method and in [60] for the GMRES method). An improvement of the speed of Krylov methods (the so-called superlinear convergence) seems to appear in the iterative procedure due to a modest degree of convergence of the Ritz values. In Fig. 2, for some problems we compare the Ritz values of the matrix H_k after $k = 40$ steps of the Hessenberg and Arnoldi procedures against the eigenvalues of the coefficient matrix. The Ritz values generated by both procedures are good approximation of (part of) the spectrum of the test matrices. The Hessenberg process can capture the spectral information effectively, sometimes even better than the Arnoldi process.

Based on these observations, Algorithm 1 can be accurate and thus represent a cost-effective alternative to the Arnoldi method in iterative solutions of some systems of linear equations.

2.2 The restarted Hessenberg method

As we know, we derive the restarted FOM method from the specified Hessenberg decomposition like (7), which is generated by Arnoldi process. Here we follow this framework of restarted FOM method 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 [19], which takes $\mathcal{L} = \mathcal{K}_m(A, \boldsymbol{r}_0)$ in which $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$. Then we seek an approximate

[1] A short note for describing the relations between their ELMRES method [56, 57] and Sadok's CMRH method [48] are available at `http://ncsu.edu/hpc/Documents/Publications/gary_howell/contents.html#codes`.

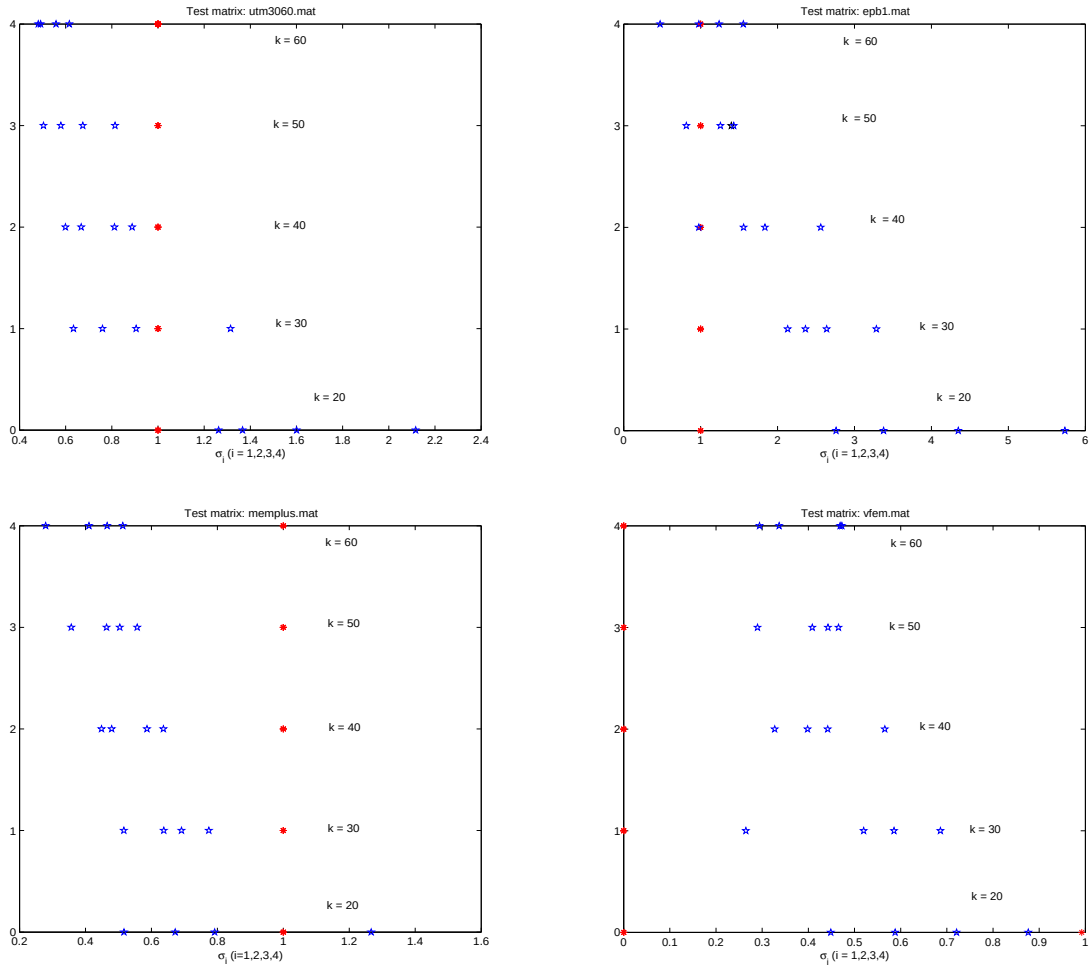

Fig. 1: Four smallest singular values of the matrices L_k and V_k with different k. 1) Arnoldi process (red stars); 2) Hessenberg process (blue pentagrams).

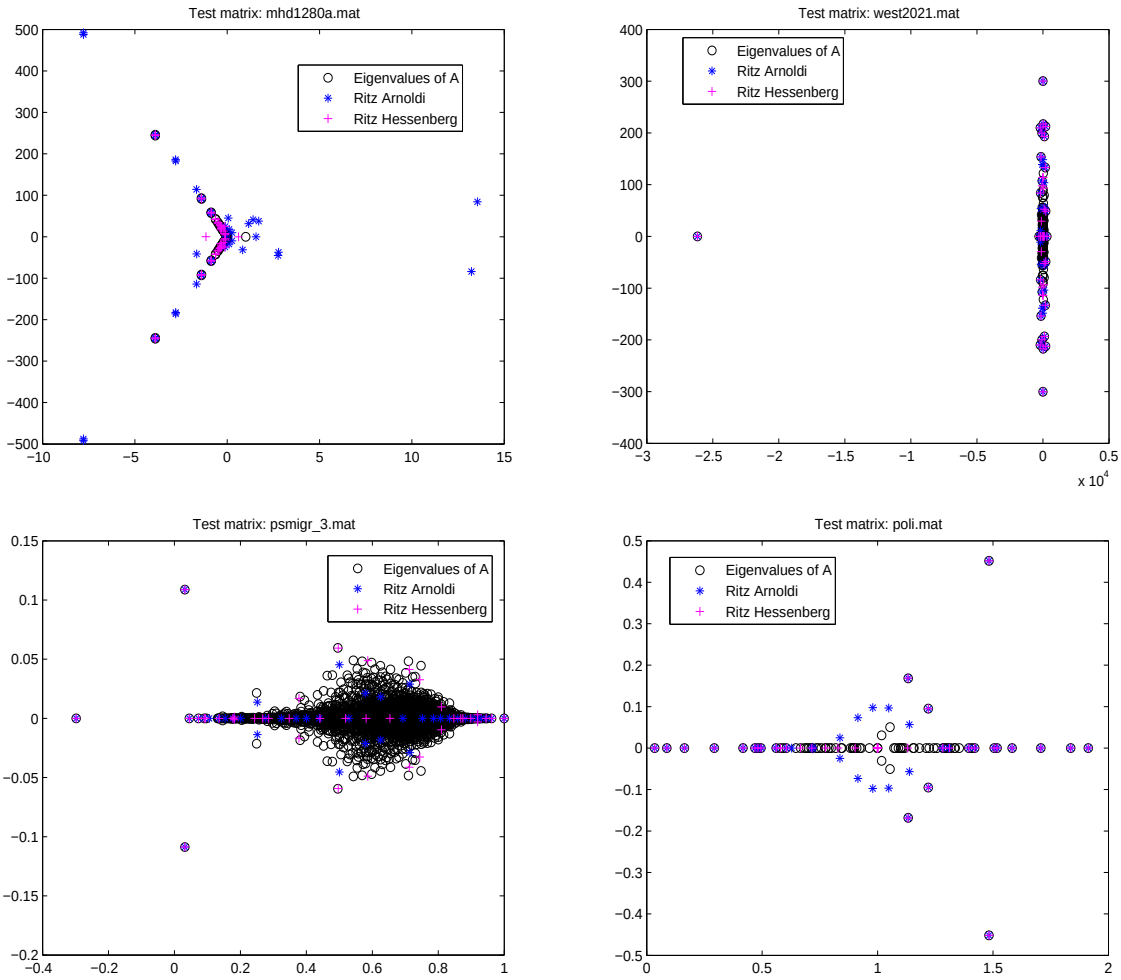

Fig. 2: Ritz values generated by Arnoldi and Hessenberg procedures. The approximations of the eigenvalues.

solution $\boldsymbol{x}_m$ from the affine subspace $\boldsymbol{x}_0 + \mathcal{K}_m$ of dimension m, i.e., we can express it as $\boldsymbol{x}_m = \boldsymbol{x}_0 + L_m \boldsymbol{y}_m$ for some vector $\boldsymbol{y}_m$. Furthermore, the residual vector can be computed

$$\begin{aligned} \boldsymbol{r}_m &= \boldsymbol{b} - A\boldsymbol{x}_m = \boldsymbol{b} - A(\boldsymbol{x}_0 + L_m \boldsymbol{y}_m) \\ &= \boldsymbol{r}_0 - AL_m \boldsymbol{y}_m \\ &= \boldsymbol{r}_0 - L_m H_m \boldsymbol{y}_m - h_{m+1,m} \boldsymbol{l}_{m+1} \boldsymbol{e}_m^T \boldsymbol{y}_m \\ &= L_m(\beta \boldsymbol{e}_1 - H_m \boldsymbol{y}_m) - h_{m+1,m} \boldsymbol{l}_{m+1} \boldsymbol{e}_m^T \boldsymbol{y}_m. \end{aligned} \tag{8}$$

So if we want to make the residual close to zeros, we can enforce the "local" solution condition:

$$H_m \boldsymbol{y}_m = \beta \boldsymbol{e}_1. \tag{9}$$

As a result, the approximate solution using the above m-dimensional subspaces is given by

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + L_m \boldsymbol{y}_m, \quad \text{where} \quad \boldsymbol{y}_m = H_m^{-1}(\beta \boldsymbol{e}_1). \tag{10}$$

Finally, an iterative solver based on Algorithm 1 and called the Hessenberg method (Hessen) 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 (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 Algorithm (without restarting) is such that*

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

and therefore,

$$\|\boldsymbol{b} - A\boldsymbol{x}_m\|_2 = h_{m+1,m} |\boldsymbol{e}_m^T \boldsymbol{y}_m| \cdot \|\boldsymbol{l}_{m+1}\|_2, \tag{11}$$

Proof. We have the relation,

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

By the definition of $\boldsymbol{y}_m$, $H_m \boldsymbol{y}_m = \beta \boldsymbol{e}_1$, and so $\boldsymbol{r}_0 - L_m H_m \boldsymbol{y}_m = \boldsymbol{0}$ from which the result follows. □

At the end of this subsection, it follows that

$$\begin{aligned} \boldsymbol{r}_m &= \boldsymbol{b} - A\boldsymbol{x}_m = \boldsymbol{b} - A(\boldsymbol{x}_0 + L_m\boldsymbol{y}_m) \\ &= \boldsymbol{r}_0 - AL_m\boldsymbol{y}_m \\ &= -h_{m+1,m}[\boldsymbol{y}_m]_m\boldsymbol{l}_{m+1}, \end{aligned} \tag{12}$$

where $[\boldsymbol{y}_m]_m$ represents the last element of $\boldsymbol{y}_m$. Denote $\beta_m = -h_{m+1,m}[\boldsymbol{y}_m]_m$, then we have $\boldsymbol{r}_m = \beta_m\boldsymbol{l}_{m+1}$, which indicates that $\boldsymbol{r}_m$ is collinear with $\beta_m\boldsymbol{l}_{m+1}$.

Without considering the restarting strategy, just like analyzing the convergence relations between CMRH method and GMRES method, we can also follow the analogous idea of Sadok and Szyld [53] to present the specified analysis which explains why Hessenberg method can have the good convergence behavior. But this is not the emphasis we pursue here.

3 The shifted variant of restarted Hessenberg method

Based on the above mentioned, we follows Simoncini's framework [43] about deriving the restarted shifted FOM method to establish the shifted variant of restarted Hessenberg method. Consider now the shifted system (1). Shifting transforms (7) into

$$(A - \sigma_i I)L_m = L_m(H_m - \sigma_i I_m) + h_{m+1,m}\boldsymbol{l}_{m+1}\boldsymbol{e}_m^T\boldsymbol{y}_m, \tag{13}$$

where I_m is the identity matrix of size m. Due to (13), the only difference in Hessenberg method is that $\boldsymbol{y}_m$ is computed by solving the reduced shifted system $(H_m - \sigma_i I_m)\boldsymbol{y} = \beta\boldsymbol{e}_1$. Therefore, the expensive step of constructing the non-orthogonal basis L_m is performed only once for all values of σ of interest, $i \in \{1, \ldots, \nu\}$, whereas ν reduced systems of size m need 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. (12) for details. The next proposition shows that collinearity still holds in the shifted case when the Hessenberg method is applied.

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 (13). 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

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

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 FOM method for simultaneously solving shifted linear systems (1). We described this final algorithm in detail as following Algorithm 3.

Similarly, it was shown in [43] 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

Algorithm 3 The restarted shifted Hessenberg method

1: Given $A, \boldsymbol{b}, \boldsymbol{x}_0^{(i)} = \boldsymbol{0}$, $\{\sigma_1, \ldots, \sigma_\nu\}$, $\mathcal{I} = \{1, 2, \ldots, \nu\}$ 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 $\boldsymbol{l}_1 = \boldsymbol{r}_0/\beta_0^{(i)}$ and $\boldsymbol{p}(1) \leftrightarrow \boldsymbol{p}(i_0)$, where $\leftrightarrow$ is used to swap contents.
4: **for** $\ell = 1, 2, \ldots,$ **do**
5: Compute Hessenberg decomposition $AL_k = L_k H_k + h_{k+1,k}\boldsymbol{l}_{k+1}\boldsymbol{e}_k^T$ by Algorithm 1.
6: Solve $\boldsymbol{y}_m^{(i)} = (H_m - \sigma_i I_m)^{-1}(\beta_{\ell-1}^{(i)}\boldsymbol{e}_1), \quad i \in \mathcal{I}$.
7: Update $\boldsymbol{x}_{m,\ell}^{(i)} = \boldsymbol{x}_{m,\ell-1}^{(i)} + L_m \boldsymbol{y}_m^{(i)}, \quad i \in \mathcal{I}$.
8: Update $\mathcal{I}$. If $\mathcal{I} = \emptyset$, exit. EndIf
9: Set $\beta_\ell^{(i)} = -h_{m+1,m}[\boldsymbol{y}_m^{(i)}]_m, \quad i \in \mathcal{I}$.
10: Set $\boldsymbol{l}_1 = \boldsymbol{l}_{m+1}$.
11: **end for**

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 Section 4 for this issue.

Here, we shall analyze the computational cost of implementing the restarted shifted Hessenberg method, the restarted shifted FOM method, and the weighted restarted shifted FOM method. It is known from Section 3 and Refs. [20,43] that the difference of arithmetic operations of these three methods comes from processes in generating the basis vectors. Other operational requirements, like solving ν linear sub-systems defined as line 6 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, restarted shifted FOM and weighted restarted shifted FOM methods. Let us denote by Nz the number of nonzero entries of A in (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 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 and weighted Arnoldi processs, respectively. If we neglect the cost of computing the maximum of the vector $\boldsymbol{l}_j$ in the Hessenberg process, 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 (see [20] for instance) as following Table 1.

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

In Table 1, m is the restarting frequency; and the definition of "D-orthogonal" basis can be found in [20, Algorithm 2]. Firstly, it is remarkable that the restarted shifted Hessenberg method, the restarted shifted FOM method, and the weighted restarted shifted FOM method have the similar implementations of lines 6-10 of Algorithm 3. As seen from Table 1, when the computational cost of (similar) lines 6-10 in these three mentioned shifted iterative solvers are comparable, e.g., the required number of restarts are similar, the computational cost of the restart shifted Hessenberg methods can be less than that of both restarted shifted FOM and weighted restarted shifted FOM methods, also refer to Section 4 for futher

discussion.

At the end of this section, it is still worth noting that the preconditioning techniques can be often exploited to accelerate the Krylov subspace methods for solving the linear systems $A\boldsymbol{x} = \boldsymbol{b}$. However, the development of particular preconditioners for the iterative methods, which are used to solve shifted linear systems, is not prosperous in most of cases. It is chiefly because most of preconditioning techniques cannot make the preconditioned systems maintain the shift-invariance property, i.e. Eq. (3), of the Krylov subspace. As far as we know, the particular preconditioning techniques in current research concerning shifted linear systems can be almost classified into the following cases:

1. **Shift-and-invert preconditioners**: The main idea of these preconditioners is to transform the original shifted linear systems into a new shifted linear systems. More precisely, from [2, 3, 14, 61], it is well-known that the shift-and-invert preconditioner $P = A - \tau I$ applied to the shifted linear systems (1) leads to:

$$\begin{aligned}(A-\sigma_i I)(A-\tau I)^{-1}(A-\tau I) &= \Big[(A-\tau I)-(\sigma_i-\tau)I\Big](A-\tau I)^{-1}(A-\tau I)\\ &= [I-(\sigma_i-\tau)(A-\tau I)^{-1}](A-\tau I),\\ &= \Big[\frac{1}{\sigma_i-\tau}I-(A-\tau I)^{-1}\Big](\sigma_i-\tau)(A-\tau I),\end{aligned}$$

   so we define the following auxiliary vectors,

$$\boldsymbol{y}^{(i)} = (\tau-\sigma_i)(A-\tau I)\boldsymbol{x}^{(i)} \quad \Leftrightarrow \quad \boldsymbol{x}^{(i)} = \frac{1}{\tau-\sigma_i}(A-\tau I)^{-1}\boldsymbol{y}^{(i)},$$

   the preconditioned shifted problem can be rewritten as the new unpreconditioned shifted linear systems

$$\Big[(A-\tau I)^{-1}-\eta_i I\Big]\boldsymbol{y}^{(i)} = \boldsymbol{b}, \qquad \eta_i = \frac{1}{\sigma_i-\tau}, \quad \boldsymbol{x}^{(i)} = \frac{1}{\tau-\sigma_i}(A-\tau I)^{-1}\boldsymbol{y}^{(i)}.$$

   As a result, solving systems with the preconditioner P only involves the matrix $A - \tau I$; only one LU-decomposition of the matrix $A - \tau I$ has to be computed that can be used for both the preconditioning operations and for computing the solutions of the unpreconditioned shifted systems. The shift-and-invert preconditioner described above has, however, some significant disadvantages. The preconditioner is very effective for shifted systems with σ_i close to τ, but is much less effective for shifts further away from τ. Another disadvantage is that computing the LU decomposition of $A - \tau I$ may still be prohibitively expensive for large 3D problems, [2, 14].

2. **Polynomial preconditioner**: Generally speaking, we apply a preconditioning polynomial $P_n(z)$ and solve the system of linear equations $P_n(A)A\boldsymbol{y} = \boldsymbol{b}$, $\boldsymbol{x} = P_n(A)\boldsymbol{y}$. $P_n(z)$ will generally depend on the shift σ_i, so we are looking for polynomials $P_{n,\sigma_i}(z)$ which satisfy

$$P_{n,\sigma_i}(A-\sigma_i I)(A-\sigma_i I) = P_{n,0}(A)A - \eta_i I$$

   and which are good preconditioners. Suitable polynomials can for example be constructed from Chebychev-, Leja- or GMRES-polynomials, refer to [21]. Recently, Ahmad et al. have designed an efficient Chebychev-polynomial preconditioner [2] for two shifted linear systems arising from $\mathcal{H}^2$-optimal model reduction, their numerical results shown the effectiveness of the proposed preconditioner for accelerating the shifted BiCG for solving two shifted linear systems simultaneously. In 2012, Wu et al. also proposed an efficient polynomial preconditioner for accelerating shifted restarted GMRES algorithms to handle the PageRank problems, see [15] for details. Although the polynomial preconditioner is easier to established and more efficient than those of the shift-and-invert preconditioners, the polynomial preconditioners are always not popular among the research of preconditioning techniques.

3. **Flexible preconditioner**: This preconditioning idea was firstly introduced by Gu and Zhou in [44]. Then it was successfully used to solve the shifted linear systems coming from some interesting practical applications [16, 17]. They began to use the flexible Arnoldi process with some suitable parameters τ_k to generate the basis of Krylov subspace, then the flexible FOM and GMRES (including their restarted versions) for shifted linear systems can be derived directly. In general, we think it still belongs to the shift-and-invert preconditioner techniques, the difference is just using some variable (approximation) shifted parameters τ_k during the each preconditioned step. Although many numerical results had been done for illustrating the effectiveness of the flexible preconditioners, they still suffer the similar difficulties of shift-and-invert preconditioners: (i) need to call the LU-decomposition for solving k sub-linear systems in the flexible Arnoldi process, which turns to be prohibitively expensive for large 3D problems; (ii) Moreover, we still require to select the approximation shifted parameters τ_k (although only two or three different $\{\tau_k\}$'s) carefully according to the clustered distributing of the shifts σ_i.

4. **Nested iterative techniques**: In order to remedy the mentioned difficulties of above different preconditioning techniques, Baumann and Gijzen [14] recently made full use of the collinear property of residual and Krylov subspace to establish a new family of nested iterative techniques. The proposed techniques do not need to select the approximation shifted parameters. Meanwhile, we only need to use a inner Krylov subspace method (such as shifted FOM and shifted IDR(s) methods) to solve sub-shifted linear systems roughly but the outer Krylov subspace methods (such as shifted GMRES and shifted QMRIDR(s) methods) for shifted linear systems (1) can still converge to the desired approximation solutions very well. From the numerical results presented in [14], the nested iterative techniques are very useful to accelerate the iterative solvers for shifted linear systems arising in the (numerical) solutions of the elastic wave equations.

Finally, it is remarkable to mention that all these four frameworks of preconditioning techniques can be employed to accelerate the proposed restarted shifted Hessenberg method for solving the shifted linear systems (1). For instance, we can derive the flexible Hessenberg process [54] by following the framework of flexible Arnoldi process, which will be similar to the Hessenberg decomposition in [44], then the flexible (restarted) shifted Hessenberg method is not hard to construct via following the philosophy of flexible (restarted) shifted FOM method for handling the shifted linear systems (1). But for the sake of simplicity, we will not pursue the specified preconditioning techniques for the restarted shifted Hessenberg method proposed in this paper for shifted linear systems.

4 Numerical examples

Far from being exhaustive, in order to show the attractive convergence behavior with respect to both the number of restarts (abbreviated as `Rests`), CPU time elapsed[2] (abbreviated as `CPU`) and the final true relative residual 2-norms of ν linear systems (i.e. $TRR_i := \log_{10}(\|\boldsymbol{b} - (A - \sigma_i I)\boldsymbol{x}_k\|_2/\|\boldsymbol{b} - (A - \sigma_i I)\boldsymbol{x}_0\|_2),\ i = 1, 2, \ldots, \nu$) using Algorithm 3 to solve (1) simultaneously, some numerical experiments have been reported in this section. In addition, we define the "mean value" of TRR_i as follows,

$$\overline{TRR_i} = \frac{TRR_1 + TRR_2 + \ldots + TRR_\nu}{\nu}, \quad i = 1, 2, \ldots, \nu.$$

The dimension of approximation subspace is chosen to be different m. We will compare the proposed method (referred to as `sHessen(`m`)`) with the restarted shifted FOM method (referred to as `sFOM(`m`)`) proposed by Simoncini in [43] and the restarted weighted shifted FOM method (referred to as `wsFOM(`m`)`) introduced by Jing and Huang in [20].

Unless otherwise noted, the initial guess solution $\boldsymbol{x}_0$ 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 $\boldsymbol{x}_k$ is the approximate solution in the k-th

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

cycle, we stop the procedure if $\boldsymbol{x}_k$ satisfies

$$\frac{\|\boldsymbol{b}-(A-\sigma_i I)\boldsymbol{x}_k\|_2}{\|\boldsymbol{b}-(A-\sigma_i I)\boldsymbol{x}_0\|_2} < \text{tol} = 10^{-8}, \quad i \in \mathcal{I}$$

or the maximal number of restarts, e.g., 500, is reached. 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. In the first example, we consider a classical test problem: let the test matrix A be a 800×800 upper bidiagonal matrix, with entries $0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 10, 11, \ldots, 802$ on the main diagonal and all ones on the super diagonal. In our numerical experiments, two sets of shifted values $\mathcal{I}_1 = \{\sigma_1 = -0.5, \sigma_2 = 0.5\}$ and $\mathcal{I}_2 = \{\sigma_1 = -1, \sigma_2 = 1\}$, for shifted systems are considered. The numerical results of different shifted iterative solvers for handling the shifted linear systems (1) are displayed in Tables 2-3.

Table 2: Comparison for Example 1 in terms of the `Rests`, `CPU` and TRR_i ($i \in \mathcal{I}_1$).

	`sHessen`(m)			`sFOM`(m)			`wsFOM`(m)		
m	`Rests`	`CPU`	TRR_i	`Rests`	`CPU`	TRR_i	`Rests`	`CPU`	TRR_i
20	24	0.0782	-8.7680	†	†	†	†	†	†
			-8.2676			†			†
30	17	0.0805	-8.2224	†	†	†	22	0.1142	-8.0110
			-8.2676			†			-8.0417
40	14	0.0824	-8.4611	†	†	†	19	0.1168	-8.5784
			-9.8259			†			-8.0907
50	13	0.0848	-8.5224	50	0.3099	-8.9028	†	†	†
			-10.2510			-8.9778			†

Table 3: Comparison for Example 1 in terms of the `Rests`, `CPU` and TRR_i ($i \in \mathcal{I}_2$).

	`sHessen`(m)			`sFOM`(m)			`wsFOM`(m)		
m	`Rests`	`CPU`	TRR_i	`Rests`	`CPU`	TRR_i	`Rests`	`CPU`	TRR_i
20	19	0.0728	-8.0473	†	†	†	49	0.1218	-8.2434
			-8.3069			†			-8.6141
30	14	0.0751	-8.7570	35	0.1336	-8.6597	19	0.1013	-8.0891
			-8.6746			-8.9933			-8.8851
40	10	0.0787	-9.0297	33	0.1616	-8.0348	12	0.0917	-8.1719
			-8.1725			-8.0140			-8.2135
50	10	0.0841	-8.4248	27	0.1667	-8.9799	12	0.1196	-9.2894
			-9.0494			-9.5218			-8.9802

As seen from Tables 2-3, our proposed iterative solvers (`sHessen`(m)) with different m can be successfully employed to solve the shifted linear systems (1), whereas `sFOM`(m) and `wsFOM`(m) solvers can not do it in some cases. More precisely, the proposed method, `sHessen`(m), is more efficient and cheaper than both `sFOM`(m) and `wsFOM`(m) solvers for shifted linear systems (1) in terms of the number of restarts and CPU time elapsed. At the same time, apart from `wsFOM(50)` in Table 2, the weighted strategy for Arnoldi process behind the shifted iterative solvers of Jing and Huang improves indeed the convergence behavior of restarted shifted FOM method, which is constructed by classical Arnoldi process. When all these shifted iterative solvers converge to the desired approximation solutions of shifted linear systems, they almost lead to the similar accuracy of solutions in aspects of TRR_i. In conclusion, the proposed `sHessen`(m) method can be viewed as the best solvers for shifted linear systems in Example 1.

Example 2. Here, we still consider a classical test problem [22]: the test matrix $A \in \mathbb{R}^{2000\times 2000}$ is defined as following form

$$A = \begin{pmatrix}
1 & 0.21 & 1.2 & 0 & 0.13 & 1.42 & & & & \\
0.45 & 2 & 0.21 & 1.2 & 0 & 0.13 & 1.42 & & & \\
0 & 0.45 & 3 & 0.21 & 1.2 & 0 & 0.13 & & & \\
0.12 & 0 & 0.45 & 4 & 0.21 & & & \ddots & & \\
0.11 & 0.12 & 0 & 0.45 & & & \ddots & & 0.13 & 1.42 \\
 & 0.11 & 0.12 & & & \ddots & & 1.2 & 0 & 0.13 \\
 & & & & \ddots & & 1997 & 0.21 & 1.2 & 0 \\
 & & & \ddots & & 0 & 0.45 & 1998 & 0.21 & 1.2 \\
 & & & & 0.11 & 0.12 & 0 & 0.45 & 1999 & 0.21 \\
 & & & & & 0.11 & 0.12 & 0 & 0.45 & 2000
\end{pmatrix}.$$

Similarly, in our numerical experiments, only one set of shifted values $\mathcal{I}_1 = \{\sigma_1 = -0.5, \sigma_2 = 0.5\}$ for shifted linear systems is considered.

Table 4: Comparison for Example 2 in terms of the `Rests`, `CPU` and TRR_i ($i \in \mathcal{I}_1$).

	`sHessen`(m)			`sFOM`(m)			`wsFOM`(m)		
m	`Rests`	`CPU`	TRR_i	`Rests`	`CPU`	TRR_i	`Rests`	`CPU`	TRR_i
20	91	0.2622	-8.0328	63	0.1957	-8.0073	†	†	†
			-8.1289			-8.0830			†
30	86	0.3498	-8.7528	25	0.1417	-8.0121	†	†	†
			-8.3199			-8.4126			†
40	18	**0.1253**	-8.2803	16	0.1288	-8.2030	†	†	†
			-10.0432			-8.4805			†
50	16	0.1563	-8.2780	13	0.1489	-8.7887	†	†	†
			-8.6143			-8.4539			†

According to the results reported in Table 4, `sFOM`(m) outperforms the others two shifted iterative solvers, `sHessen`(m) and `sFOM`(m), in terms of the number of restarts. Even apart from `sFOM`(40) and `sHessen`(40), the performance of `sFOM`(m) is much better than that of `sHessen`(m) for solving shifted linear systems (1) in terms of CPU time elapsed. However, it is remarkable that `sHessen`(40) needs more `Rests` but still slightly less `CPU` to converge for solving shifted linear systems (1) with compared to `sFOM`(40). That is because the Hessenberg process is often slightly cheaper to implement than the Arnoldi process, especially when the number of restarts of these two algorithms are similar, refer to [47,48] for this discussion. Unfortunately, all kinds of `wsFOM`(m) solvers fail to solve the shifted linear systems (1) in Example 2 completely. Similarly, when all these shifted iterative solvers converge to the desired approximation solutions of shifted linear systems, they almost lead to the same accuracy of solutions in terms of TRR_i. In conclusion, our proposed algorithm (`sHessen`(m)) still can be regarded as an acceptable alternative considerably.

Example 3. In this example, we consider a large set of publicly available test matrices arising from several application areas and having increasing levels of difficulty. These test matrices are extracted from Tim Davis's matrix collection at the University of Florida. We summarize in Table 5 the characteristics of these shifted linear systems that were solved.

From the results reported in Table 6, the proposed `sHessen`(40) can solve the first, sixth and eighth test problems very well, whereas both `sFOM`(m) and `wsFOM`(m) are not at all able to converge to the desired approximation solutions of shifted linear systems (1). Besides, the proposed algorithm is more

Table 5: Set and characteristics of test matrices in Example 3 (listed in increasing matrix size).

Matrices	Size	Field	$nnz(A)$	Set of shifts
bfwa782	782	Electromagnetics	7,514	$\{\sigma_1 = -0.003, \sigma_2 = 0.003\}$
orsirr_2	886	Oil reservoir modeling	5,970	$\{\sigma_1 = -0.5, \sigma_2 = 0.5\}$
orsirr_1	1,030	Oil reservoir modeling	6,858	$\{\sigma_1 = -0.5, \sigma_2 = 0.5\}$
rdb1250l	1,250	Chemical engineering	7,300	$\{\sigma_1 = -0.2, \sigma_2 = 0.2\}$
orsreg_1	2,205	Oil reservoir modeling	14,133	$\{\sigma_1 = -1, \sigma_2 = 1\}$
cavity16	4,562	Finite element modeling	137,887	$\{\sigma_1 = -0.1, \sigma_2 = 0.1\}$
shyy41	4,720	Computational fluid dynamics	20,042	$\{\sigma_1 = -1, \sigma_2 = 1\}$
memplus	17,758	Circuit simulation	303,468	$\{\sigma_1 = -0.01, \sigma_2 = 0.01\}$
matrix_9	103,430	Semiconductor device problem	1,205,518	$\{\sigma_1 = -0.9, \sigma_2 = 0.9\}$

Table 6: Results of different solvers for Example 3 with different m in terms of the Rests, CPU and TRR_i.

		sHessen(m)			sFOM(m)			wsFOM(m)		
Matrix	m	Rests	CPU	TRR_i	Rests	CPU	TRR_i	Rests	CPU	TRR_i
bfwa782	40	98	0.3706	-8.0955	†	†	†	†	†	†
				-8.5808			†			†
orsirr_2	20	146	0.2637	-8.4626	210	0.3534	-8.6617	†	†	†
				-8.0278			-8.0666			†
orsirr_1	30	100	0.3068	-8.0730	137	0.4222	-8.0538	†	†	†
				-8.0601			-8.0404			†
rdb1250l	40	18	0.0914	-8.0377	17	0.0956	-8.8174	†	†	†
				-9.0494			-9.4118			†
orsreg_1	50	12	0.1271	-8.6118	9	0.1332	-8.3282	†	†	†
				-8.5994			-8.0713			†
cavity16	40	90	1.5435	-8.2723	†	†	†	†	†	†
				-8.0371			†			†
shyy41	20	16	0.0922	-8.1014	125	0.7497	-8.3370	71	0.5035	-8.2818
				-8.2407			-8.0434			-8.1367
shyy41	30	11	0.1061	-8.0967	11	0.1839	-8.9070	13	0.2405	-8.6073
				-8.6036			-8.1820			-8.0881
shyy41	40	7	0.0981	-8.4446	16	0.3519	-9.1335	9	0.2635	-9.2537
				-8.0639			-8.9935			-8.0711
shyy41	50	6	0.1172	-8.5789	7	0.2472	-8.6861	6	0.2541	-8.8901
				-8.6889			-8.4041			-8.8895
memplus	40	35	1.8512	-10.3687	†	†	†	†	†	†
				-8.6259			†			†
matrix_9	40	1	0.4634	-15.6015	1	0.6390	-15.3415	1	0.7054	-15.2871
				-15.6400			-15.0908			-15.3918

efficient than both `sFOM(`m`)` and `wsFOM(`m`)` in terms of the number of restarts and CPU time elapsed, when these three shifted iterative solvers are applied to solve test problems: `orsirr_2`, `orsirr_1` and `shyy41` ($m = 20, 30, 40, 50$). Moreover, although `sHessen(`m`)` need more number of restarts for solving `rdb1250l` and `orsreg_1` than those required by `sFOM(`m`)`, `sHessen(`m`)` still needs slightly less CPU time elapsed to converge than those needed by `sFOM(`m`)`. For `rdb1250l`, `orsreg_1`, `matrix_9` and `shyy41` ($m = 30, 50$), these three shifted iterative solvers require the similar number of restarts to converge, but the proposed `sHessen(`m`)` still needs slightly less CPU time elapsed than other shifted iterative solvers. Especially, for `matrix_9`, our proposed `sHessen(`m`)` method can achieve more accurate solution than other two iterative methods in terms of TRR_i. In fact, we had given an explanation to these phenomena in Example 2. That is because the Hessenberg process is often cheaper to implement than the Arnoldi process, especially when the number of restarts of these two algorithms are similar. The `wsFOM(`m`)` solvers fail to solve the shifted linear systems (1) in Example 3 except `shyy41` ($m = 20, 30, 40, 50$). We also note that when all these shifted iterative solvers converge to the desired approximation solutions of shifted linear systems, they almost lead to the similar accuracy of solutions in aspects of TRR_i. In conclusion, our proposed algorithm (`sHessen(`m`)`) still can be regarded as a highly recommend choice for shifted linear systems (1) with compared to other two mentioned shifted iterative solvers.

Example 4. In this example, we consider test problems stem from a structural dynamics engineering problem, whose algebraic formulation was discussed in detail in [3]. Direct frequency analysis leads to the solution of the following algebraic linear system

$$[(-\omega_j^2 M + K) + \iota(\omega_j C_V + C_H)]\boldsymbol{x} = \boldsymbol{b}, \quad \iota = \sqrt{-1}, \tag{14}$$

where $\omega_j,\ j = 1, 2, \ldots, \nu$ are the driving circular frequencies, M and K are the inertia and the stiffness matrices, C_V and C_H are the viscous and the hysteretic damping matrices, respectively. We take $M = I, C_V = 10I, C_H = \mu K$ with $\mu = 0.02$ a damping coefficient, and K the five-point centered difference matrix approximating the negative Laplacian operator with homogeneous Dirichlet boundary conditions on an uniform mesh $h = 1/(p+1)$ in the unit square $[0,1] \times [0,1]$. The matrix $K \in \mathbb{R}^{n\times n}$ possesses the tensor-product form $K = B_p \otimes I + I \otimes B_p$, with $B_p = h^{-2} \cdot \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{p\times p}$. Hence, K is an $n \times n$ block-tridiagonal matrix, with $n = p^2 = 64^2$. We set the right-hand side vector $\boldsymbol{b}$ to be $\boldsymbol{b} = \boldsymbol{e}_{1160} + \boldsymbol{e}_{2298}$, with $\boldsymbol{e}_i$ being the i-th column vector of identity matrix. As before, we normalize the system by multiplying both sides through by h^2; see [43,62] for details. Due to $M = I$, then linearization yields the algebraic system (14),

$$\Big[\begin{pmatrix} \iota h^2 C_V & h^2(K+\iota C_H) \\ h^2(K+\iota C_H) & 0 \end{pmatrix} - \omega_j \begin{pmatrix} h^2 M & 0 \\ 0 & h^2(K+\iota C_H) \end{pmatrix}\Big] \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} h^2\boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix} \quad \Leftrightarrow \quad (T - \omega_j S)\boldsymbol{z} = \boldsymbol{f}$$
$$\Leftrightarrow \Big[\begin{pmatrix} \iota C_V & I \\ K+\iota C_H & 0 \end{pmatrix} - \omega_j \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}\Big]\hat{\boldsymbol{z}} = \boldsymbol{f}, \qquad \begin{bmatrix} h^2 M & 0 \\ 0 & h^2(K+\iota C_H) \end{bmatrix} \boldsymbol{z} = \hat{\boldsymbol{z}}.$$

Whenever S is nonsingular, we can write the problem above as in (1), namely $(TS^{-1} - \omega_j I)\hat{\boldsymbol{z}} = \boldsymbol{f}$ and $S\boldsymbol{z} = \hat{\boldsymbol{z}}$. For the sake of simplicity, we only report results for two parameters $\omega_1 = 2\pi$ and $\omega_2 = 6\pi$. The linearized problem has dimension 8192.

As observed from Table 7, our proposed `sHessen(`m`)` methods outperform both `sFOM(`m`)` and `wsFOM(`m`)` methods in terms of the number of restarts and CPU time elapsed. Especially, it is remarkable that `sHessen(40)` almost needs only half of the number of restarts and CPU time elapsed with compared to `sFOM(40)`. Again, the `wsFOM(`m`)` methods still fail to solve the resulting shifted linear systems (1) in Example 4. Again, when all these shifted iterative solvers converge to the desired approximation solutions of shifted linear systems, they almost lead to the same accuracy of solutions in terms of TRR_i. In conclusion, the proposed `sHessen(`m`)` method is the best solver among these mentioned shifted iterative methods for handling the resulting shifted linear systems in Example 4.

Example 5. In this example, we consider the semi-discretization by finite difference method of the following three dimensional time-fractional convection-diffusion problems, which are modified from

Table 7: Comparison for Example 4 in terms of the `Rests`, `CPU` and TRR_i.

		`sHessen`(m)			`sFOM`(m)			`wsFOM`(m)		
μ	m	`Rests`	`CPU`	TRR_i	`Rests`	`CPU`	TRR_i	`Rests`	`CPU`	TRR_i
$\mu = 0.02$	40	27	2.2267	-8.4181	59	5.2156	-8.0859	†	†	†
				-8.0418			-8.2960			†
	50	19	2.2042	-8.8472	24	3.3261	-8.1201	†	†	†
				-8.3296			-8.0489			†
$\mu = 0.04$	40	22	1.8625	-8.0957	55	4.8742	-8.1618	†	†	†
				-8.2702			-8.2534			†
	50	18	2.0801	-8.0754	23	2.9937	-8.2278	†	†	†
				-8.0704			-8.3017			†

Ref. [63].

$$\begin{cases} D_t^\lambda u - \Delta u + \tau_1 u_{x_1} + \tau_2 u_{x_2} = 0, & \text{on } (0,1)^3 \times (0,T), \\ u(x,t) = 0, & \partial(0,1)^3 \text{ for all } t \in [0,T], \\ u(x,0) = u_0(x), & x \in (0,1)^3. \end{cases} \tag{15}$$

The operator D_t^λ is Caputo's fractional derivative, defined for $0 < \lambda < 1$ by

$$D_t^\lambda f(t) = \frac{1}{\Gamma(1-\lambda)} \int_0^t \frac{f'(s)}{(t-s)^\lambda} ds,$$

where $\Gamma(\cdot)$ is the Gamma function. As $\lambda \to 1$, the fractional derivative approaches the ordinary derivative, and (15) reduces to the standard partial differential equation, the PDE is classified as sub-diffusive [63]. Discretizing the Laplacian by the usual five/seven-point stencil and the first-order derivatives, u_{x_1} and u_{x_2}, by central differences on a uniform grid with step size $h = 1/(n+1)$ leads to an ordinary initial value problem

$$\begin{cases} \frac{d^\lambda \boldsymbol{u}(t)}{dt^\lambda} = A\boldsymbol{u}, & t \in (0,T), \\ \boldsymbol{u}(0) = \boldsymbol{u}_0 \end{cases} \tag{16}$$

with the matrix

$$A = I_n \otimes I_n \otimes C_1 + [B \otimes I_n + I_n \otimes C_2] \otimes I_n \in \mathbb{R}^{N \times N}, \quad N = n^3,$$

where I_n is the identity matrix of order n, and

$$B = \frac{1}{h^2}\text{tridiag}(1, -2, 1), \quad C_j = \frac{1}{h^2}\text{tridiag}(1+\mu_j, -2, 1-\mu_j), \quad j = 1, 2$$

with $\mu_j = \tau_j h/2$. The nonsymmetric matrix A is a popular test matrix because its eigenvalues are explicitly known: If $|\mu_j| > 1$ (for at least one j), they are complex; more precisely (cf. [45]),

$$\Lambda(A) \subset \frac{1}{h^2}[-6 - 2\cos(\pi h)\text{Re}(\theta), -6 + 2\cos(\pi h)\text{Re}(\theta)] \times \frac{1}{h^2}[-2\iota\cos(\pi h)\text{Im}(\theta), 2\iota\cos(\pi h)\text{Im}(\theta)$$

with $\theta = 1 + \sqrt{1-\mu_1^2} + \sqrt{1-\mu_2^2}$. As in [45], we choose $T = 1, h = 1/20, \tau_1 = 80, \tau_2 = 120$ which leads to $\mu_1 = 2, \mu_2 = 3$, respectively. Then the analytical solution to the semi-discrete problem (16) can be expressed in terms of the Mittag-Leffler function [8, 65]

$$\boldsymbol{u}(t) = E_{\lambda,1}(At^\lambda)\boldsymbol{u}_0 = \boldsymbol{u}_0 + t^\lambda E_{\lambda,\lambda+1}(At^\lambda)A\boldsymbol{u}_0, \quad \text{where } E_{\lambda,\varsigma}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\lambda k + \varsigma)}.$$

As $\lambda \to 1$, this reduces to $\boldsymbol{u}(t) = \exp(At)\boldsymbol{u}_0$, which is the matrix exponential solution to the semi-discrete differential equations. According to the numerical method introduced in [7], we require to solve

the shifted linear systems $(A-\sigma_i I)\boldsymbol{x}=\boldsymbol{v}_0,\ \sigma_i\in\mathbb{C}$. Here refer to [8] for choosing the complex shifts σ_i. For simplicity, here we set $\boldsymbol{v}_0=\frac{A\boldsymbol{u}_0}{\|A\boldsymbol{u}_0\|_2}$, $\boldsymbol{u}_0=\boldsymbol{b}$ and the set of complex shifts $\{\sigma_1,\sigma_2,\ldots,\sigma_8\}$ can be drawn as Fig. 3.

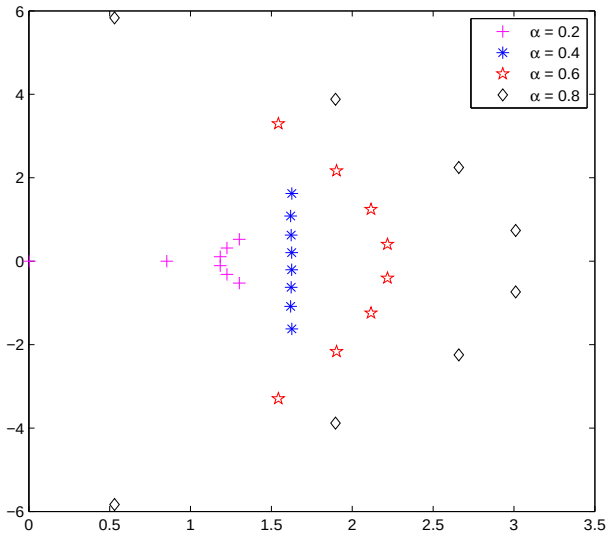

Fig. 3: Poles localization of the Carathéodory-Fejér (CF) approximation [8] to $E_{\alpha,\alpha+1}$ for $\nu=8$.

Table 8: Comparison for Example 5 in terms of the `Rests`, `CPU` and $\overline{TRR_i}$ ($t=1, h=1/20, \tau_1=80, \tau_2=120$).

		`sHessen`(m)			`sFOM`(m)			`wsFOM`(m)		
λ	m	`Rests`	`CPU`	$\overline{TRR_i}$	`Rests`	`CPU`	$\overline{TRR_i}$	`Rests`	`CPU`	$\overline{TRR_i}$
0.2	20	12	0.1967	-8.6435	7	0.1548	-8.0349	†	†	†
	30	5	0.1276	-8.6894	5	0.1675	-8.1463	†	†	†
	40	4	0.1348	-10.5550	4	0.1948	-9.2291	†	†	†
	50	3	0.1339	-14.6137	3	0.2047	-12.5138	†	†	†
0.4	20	12	0.1958	-8.6581	7	0.1561	-8.0435	†	†	†
	30	5	0.1253	-8.6981	5	0.1682	-8.1553	†	†	†
	40	4	0.1352	-10.5638	4	0.1957	-9.2383	12	0.5513	-8.4688
	50	3	0.1344	-14.6183	3	0.2079	-12.5220	5	0.3548	-8.4630
0.6	20	12	0.1963	-8.6661	7	0.1553	-8.0483	†	†	†
	30	5	0.1248	-8.7028	5	0.1669	-8.1602	†	†	†
	40	4	0.1325	-10.5686	4	0.2010	-9.2433	12	0.5429	-8.7063
	50	3	0.1317	-14.6042	3	0.2025	-12.5267	5	0.3556	-8.7585
0.8	20	12	0.1889	-8.6682	7	0.1534	-8.0495	†	†	†
	30	5	0.1199	-8.7040	5	0.1659	-8.1614	†	†	†
	40	4	0.1278	-10.5698	4	0.1898	-9.2445	11	0.4958	-8.7903
	50	3	0.1269	-14.6271	3	0.2006	-12.5278	5	0.3399	-8.6269

As observed from Tables 8, the `sFOM(20)` method outperforms both `wsFOM(20)` and `sHessen(20)` methods in terms of the number of restarts and CPU time elapsed for different cases of λ. However, it is remarkable that our proposed `sHessen(30)` method is preferable to both `sFOM(30)` and `wsFOM(30)` methods in terms of CPU time elapsed and $\overline{TRR_i}$ for different λ. Moreover, both `sHessen(40)` and `sHessen(50)` methods are also two good alternatives in aspects of CPU time elapsed and $\overline{TRR_i}$ for the

various cases of λ. Again, the `wsFOM(`m`)` methods often are expensive and easily fail to solve the shifted linear systems (1) in Example 5, such as the case of $\lambda = 0.2$. In conclusion, the proposed `sHessen(`m`)` method, rather than the `sFOM(`m`)` method, can be regarded as a highly recommend choice for handling shifted linear systems in Example 5 with compared to other two mentioned shifted iterative solvers.

Example 6. From the methods introduced in [7], the numerical evaluation of matrix exponential plays a fundamental role in solving linear differential equations. In this example, we consider the computation of the product of a matrix exponential with a vector, which arises from the numerical solution of the following fractional diffusion equation [64]

$$\begin{cases} \frac{\partial u(x,t)}{\partial t} = d(x)\,{}_aD_x^{\alpha} u(x,t) + f(x,t), & (x,t) \in (0,1) \times (0,1], \\ u(x,0) = x^3, & x \in [0,1], \\ u(0,t) = 0, \quad u(1,t) = e^{-t}, & t \in [0,1] \end{cases} \tag{17}$$

with the coefficient

$$d(x) = \frac{\Gamma(4-\alpha)}{6} x^{1+\alpha}.$$

and the source term $f(x,t) = -(1+x)e^{-t}x^3$. Here $1 < \alpha < 2$ and for the definition of the fractional order derivative, we refer to [65].

After the spatial discretization by the second-order numerical scheme proposed in [66], the equation (17) reduces to a semi-discretized ordinary differential equations of the form

$$\begin{cases} \frac{d\boldsymbol{u}(t)}{dt} = A\boldsymbol{u} + \boldsymbol{b}(t), & t \in (0,1), \\ \boldsymbol{u}(0) = \boldsymbol{u}_0, \end{cases} \tag{18}$$

where $A = \frac{DQ}{h^{\alpha}}$ with h being the grid size, D is a diagonal matrix arising from the discretization of the differential coefficient $d(x)$, and Q is a lower Hessenberg Toeplitz matrix generated by the discretization of the fractional derivative. In addition, it mentioned that the matrix A is very ill-conditioned and it is not a Toeplitz matrix; see [64, 66] for the details of the discretization. The vector $\boldsymbol{b}(t) = e^{-t}\tilde{\boldsymbol{b}}$, where be consists of the discretization of $f(x,t)/e^{-t}$ and boundary conditions, and it is independent of t. By the variation-of-constants formula, the solution of (18) at time t can be expressed as

$$\begin{aligned} \boldsymbol{u}(t) &= e^{tA}\boldsymbol{u}_0 + \int_0^t e^{(t-\varepsilon)A} e^{-\varepsilon}\tilde{\boldsymbol{b}}\, d\varepsilon \\ &= e^{tA}[\boldsymbol{u}_0 + (A+I)^{-1}\tilde{\boldsymbol{b}}] - e^{-t}(A+I)^{-1}\tilde{\boldsymbol{b}}, \end{aligned} \tag{19}$$

provided that $A+I$ is invertible. In this example, we consider two different discretized gird nodes $N = 512$ and $N = 1024$. In light of (19), to compute the solution $\boldsymbol{u}(t)$, we have to approximate the product of the matrix exponential e^{tA} with the vector

$$\hat{\boldsymbol{b}} = \boldsymbol{u}_0 + (A+I)^{-1}\tilde{\boldsymbol{b}},$$

which is the major computational cost for this problem. From the numerical treatments introduced in [10], we need to solve several shifted linear systems

$$(A - z_j I)\boldsymbol{x} = \hat{\boldsymbol{b}}, \qquad z_j \in \mathbb{C} \quad j = 1, 2, \ldots, \nu.$$

More details and choosing $\nu = 14$ complex shifts can be found in [10] and references therein. For simplicity, we plot these 14 complex shifts (i.e., poles localization of CF approximation) in Fig. 4.

As observed from Tables 9-10, our proposed `sHessen(`m`)` methods considerably outperform both `sFOM(`m`)` and `wsFOM(`m`)` methods in terms of the number of restarts, $\overline{TRR_i}$ and CPU time elapsed. Especially for the case of $\alpha = 1.3$, it is remarkable that `sHessen(40)` and `sHessen(50)` almost needs only 30% of the number of restarts and CPU time elapsed with compared to `sFOM(40)` and `sFOM(50)`.

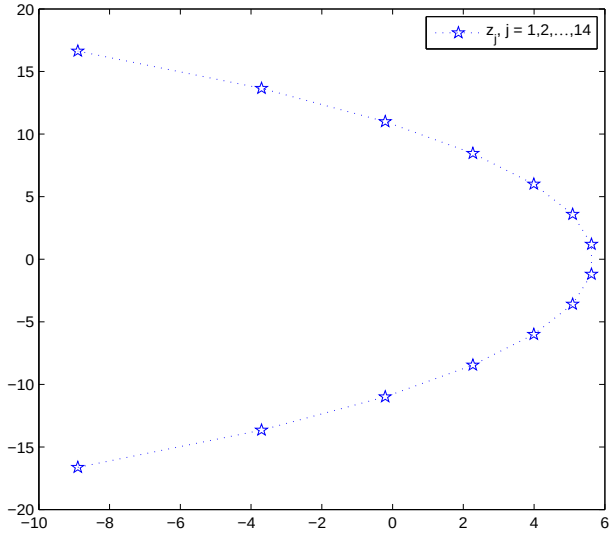

Fig. 4: Poles localization of the CF approximation to e^x with $\nu = 14$.

Table 9: Comparison for Example 6 in terms of the `Rests`, `CPU` and TRR_i ($N = 512$).

		`sHessen`(m)			`sFOM`(m)			`wsFOM`(m)		
α	m	`Rests`	`CPU`	$\overline{TRR_i}$	`Rests`	`CPU`	$\overline{TRR_i}$	`Rests`	`CPU`	$\overline{TRR_i}$
1.3	20	48	0.8447	-8.1742	†	†	†	†	†	†
	30	14	0.4618	-8.5087	†	†	†	†	†	†
	40	10	0.3492	-8.2594	92	1.4897	-8.1044	†	†	†
	50	8	0.2978	-8.2864	29	0.6620	-8.1839	†	†	†
1.7	20	39	0.9092	-8.1705	†	†	†	†	†	†
	30	28	0.8468	-8.4024	†	†	†	†	†	†
	40	23	0.7453	-8.3276	70	1.4605	-8.0964	†	†	†
	50	17	0.5686	-8.6098	35	0.9918	-8.1551	†	†	†

Table 10: Comparison for Example 6 in terms of the `Rests`, `CPU` and TRR_i ($N = 1024$).

		`sHessen`(m)			`sFOM`(m)			`wsFOM`(m)		
α	m	`Rests`	`CPU`	$\overline{TRR_i}$	`Rests`	`CPU`	$\overline{TRR_i}$	`Rests`	`CPU`	$\overline{TRR_i}$
1.5	20	46	5.8556	-8.1449	†	†	†	†	†	†
	30	28	4.0957	-8.2141	†	†	†	†	†	†
	40	24	3.8067	-8.2320	†	†	†	†	†	†
	50	16	2.7466	-8.0921	†	†	†	†	†	†
1.8	20	128	16.6387	-8.2001	†	†	†	†	†	†
	30	59	8.7667	-8.1602	†	†	†	†	†	†
	40	46	7.7286	-8.2009	†	†	†	†	†	†
	50	31	5.7586	-8.4335	†	†	†	†	†	†

Again, the wsFOM(m) methods still fail to solve all the shifted linear systems (1) in Example 6. Again, when all these shifted iterative solvers converge to the desired approximation solutions of shifted linear systems, they almost lead to the same accuracy of solutions in terms of $\overline{TRR_i}$. However, our proposed method still can archive the slightly better accuracy of final solutions in aspects of $\overline{TRR_i}$. Moreover, for the case of $N = 1024$, the sFOM(m) methods are also not able to solve any resulting shifted linear systems. In conclusion, the proposed sHessen(m) method is the best solvers among these mentioned shifted iterative methods for handling the shifted linear systems in Example 6.

5 Conclusions

Based on the results of numerical experiments we conclude that our proposed algorithm-the Restarted Shifted Hessenberg method-indeed can show considerably attractive convergence behaviors with respect to CPU time elapsed compared to the Restarted Shifted FOM proposed in [43], the Restarted Weighted Shifted FOM introduced in [20] and the shifted restarted FOM method with deflation established in [22]. Moreover, in some cases where sHessen(m) requires less enough number of restarts to converge, this algorithm can significatively reduce the CPU consuming time. In addition, since the Hessenberg process often needs slightly less computational storage than the classical Arnoldi process, so the sHessen(m) method seems to be preferable to the other Arnolid-based shifted iterative solvers (sFOM(m), wsFOM(m) and dsFOM(m)) especially if the number of restarts of these four shifted iterative solvers is similar. Besides, sHessen(m) is able to handle certain shifted linear systems while sFOM(m) and wsFOM(m) cannot.

However, the body of theoretical evidence is not available recently for the fact that sHessen(m) has advantage over sFOM(m) in terms of convergence analysis. The numerical and computational efficiency of sHessen(m) in respects of restarting number and CPU time elapsed is just illustrated on a set of problems arising both from extensive academic and from industrial applications. Furthermore, convergence analysis is also under consideration. In addition, as earlier mentioned, the preconditioning techniques are meaningful for accelerating the Krylov subspace methods to solve the shifted linear systems (1). We will develop the suitable and efficient preconditioning framework, such as Baumann and van Gijzen's idea [14] and our recent work [67], to enhance the convergence behavior of the proposed iterative solvers for shifted linear systems (1) in the future work.

Acknowledgements

The authors are grateful to Prof. Jun-Feng Yin, Dr. Ke Zhang and Dr. Jing Meng for their constructive discussions and insightful comments. Moreover, the author also would like to thank Dr. Roberto Garrappa and Prof. J.A.C. Weideman for suggesting them to choose the practical complex shifts in Examples 5-6, respectively. This research is supported by 973 Program (2013CB329404), NSFC (61170309, 61170311, 11301057, and 61402082) and the Fundamental Research Funds for the Central Universities (ZYGX2013J106, ZYGX2014J084, and ZYGX2013Z005).

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), pp. 225-244.

[2] M.I. Ahmad, D.B. Szyld, M.B. van Gijzen, Preconditioned multishift BiCG for $\mathcal{H}_2$-optimal model reduction, Report 12-06-15, Department of Mathematics, Temple University, Revised March 2013 and June 2015, 21 pages. Available online at https://www.math.temple.edu/~szyld/reports/IRKA.BICG.report.rev2.pdf.

[3] V. Simoncini, F. Perotti, On the numerical solution of $(\lambda^2 A + \lambda B + C)\boldsymbol{x} = \boldsymbol{b}$ and application to structural dynamics, SIAM J. Sci. Comput., 23 (2002), pp. 1875-1897.

[4] A. Feriani, F. Perotti, V. Simoncini, Iterative system solvers for the frequency analysis of linear mechanical systems, Comput. Methods Appl. Mech. Eng., 190 (2000), pp. 1719-1739.

[5] T. Sakurai, H. Sugiura, A projection method for generalized eigenvalue problems using numerical integration, J. Comput. Appl. Math., 159 (2003), pp. 119-128.

[6] T. Ikegami, T. Sakurai, U. Nagashima, A filter diagonalization for generalized eigenvalue problems based on the Sakurai-Sugiura projection method, J. Comput. Appl. Math., 233 (2010), pp. 1927-1936.

[7] J.A.C. Weideman, L.N. Trefethen, Parabolic and hyperbolic contours for computing the Bromwich integral, Math. Comp., 76 (2007), pp. 1341-1356.

[8] R. Garrappa, M. Popolizio, On the use of matrix functions for fractional partial differential equations, Math. Comput. Simulat., 81 (2011), pp. 1045-1056.

[9] R. Garrappa, A family of Adams exponential integrators for fractional linear systems, Comput. Math. Appl., 66 (2013), pp. 717-727.

[10] H.-K. Pang, H.-W. Sun, Fast numerical contour integral method for fractional differential equations, J. Sci. Comput., in press, 19 March 2015. Available online at `https://dx.doi.org/10.1007/s10915-015-0012-9`.

[11] A. Frommer, B. Nockel, S. Gusken, T. Lippert, K. Schilling, Many masses on one stroke: Economic computation of quark propagators, Int. J. Mod. Phys. C, 6 (1995), pp. 627-638.

[12] 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), pp. 933-943.

[13] S. Olver, Shifted GMRES for oscillatory integrals, Numer. Math., 114 (2010), pp. 607-628.

[14] M. Baumann, M.B. van Gijzen, Nested krylov methods for shifted linear systems, Tech. Report 14-01, Delft University of Technology, 36 pages, 2014, The Netherlands.

[15] 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), pp. A2558-A2575.

[16] 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), pp. A3001-A3023.

[17] T. Bakhos, A.K. Saibabab, P.K. Kitanidis, A fast algorithm for parabolic PDE-based inverse problems based on Laplace transforms and flexible Krylov solvers, J. Comput. Phys., to appear, 10 July 2015. Available online at `http://dx.doi.org/10.1016/j.jcp.2015.07.007`.

[18] R. Takayama, T. Hoshi, T. Sogabe, S.-L. Zhang, T. Fujiwara, Linear algebraic calculation of Green's function for large-scale electronic structure theory, Phys. Rev. B, 73, 165108 (2006), pp. 1-9.

[19] Y. Saad, Iterative Methods for Sparse Linear Systems (second ed.), SIAM, Philadelphia, PA (2003).

[20] Y.-F. Jing, T.-Z. Huang, Restarted weighted full orthogonalization method for shifted linear systems, Comput. Math. Appl., 57 (2009), pp. 1583-1591.

[21] 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`.

[22] J.-F. Yin, G.-J. Yin, Restarted full orthogonalization method with deflation for shifted linear systems, Numer. Math. Theor. Meth. Appl., 7 (2014), pp. 399-412.

[23] A. Frommer, BiCGStab(ℓ) for families of shifted linear systems, Computing, 70 (2003), pp. 87-109.

[24] R.W. Freund, Solution of shifted linear systems by quasi-minimal residual iterations, L. Reichel, A. Ruttan, R.S. Varga (Eds.), Numerical Linear Algebra, de Gruyter, Berlin, Germany (1993), pp. 101-121.

[25] 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), pp. 1-25.

[26] L. Du, T. Sogabe, S.-L. Zhang, IDR(s) for solving shifted nonsymmetric linear systems, J. Comput. Appl. Math., 274 (2015), pp. 35-43.

[27] S. Kirchner, IDR-Verfahren zur Lösung von Familien geshifteter linearer Gleichungssysteme, Diplomarbeit, Bergische Universität Wuppertal, Fachbereich Mathematik, Germany, Mai, 2011. (In German)

[28] M.R. Hestenes, E.L. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Natl. Bur. Stand., 49 (1952), pp. 409-436.

[29] E. Stiefel, Relaxationsmethoden bester Strategie zur Lsung linearer Gleichungssysteme, Comment. Math. Helv., 29 (1955), pp. 157-179.

[30] R. Fletcher, Conjugate gradient methods for indefinite systems, Lecture Notes in Math., vol. 506, Springer-Verlag, Berlin, Heidelberg, New York (1976), pp. 73-89.

[31] T. Sogabe, M. Sugihara, S.-L. Zhang, An extension of the conjugate residual method to nonsymmetric linear systems, J. Comput. Appl. Math., 226 (2009), pp. 103-113.

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

[33] K. Abe, G.L.G. Sleijpen, BiCR variants of the hybrid BiCG methods for solving linear systems with nonsymmetric matrices, J. Comput. Appl. Math., 234 (2010), pp. 985-994.

[34] 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), pp. 746-758.

[35] T. Sogabe, S.-L. Zhang, An extension of the COCR method to solving shifted linear systems with complex symmetric matrices, East Asian J. Appl. Math., 1 (2011), pp. 97-107.

[36] T. Sogabe, T. Hoshi, S.-L. Zhang, T. Fujiwara, On a weighted quasi-residual minimization strategy of the QMR method for solving complex symmetric shifted linear systems, Electron. Trans. Numer. Anal., 31 (2008), pp. 126-140.

[37] B.N. Parlett, D.R. Taylor, Z.A. Liu, A look-ahead Lanczos algorithm for unsymmetric matrices, Math. Comp., 44 (1985), pp. 105-124.

[38] A. Frommer, U. Glässner, Restarted GMRES for shifted linear systems, SIAM J. Sci. Comput., 19 (1998), pp. 15-26.

[39] K.M. Soodhalter, D.B. Szyld, F. Xue, Krylov subspace recycling for sequences of shifted linear systems, Appl. Numer. Math., 81 (2014), pp. 105-118.

[40] G. Gu, Restarted GMRES augmented with harmonic Ritz vectors for shifted linear systems, Int. J. Comput. Math., 82 (2005), pp. 837-849.

[41] G. Gu, J. Zhang, Z. Li, Restarted GMRES augmented with eigenvectors for shifted linear systems, Int. J. Comput. Math., 80 (2003), pp. 1037-1047.

[42] D. Darnell, R.B. Morgan, W. Wilcox, Deflated GMRES for systems with multiple shifts and multiple right-hand sides, Linear Algebra Appl., 429 (2008), pp. 2415-2434.

[43] V. Simoncini, Restarted full orthogonalization method for shifted linear systems, BIT, 43 (2003), pp. 459-466.

[44] G.-D. Gu, X.-L. Zhou, L. Lin. A flexible preconditioned Arnoldi method for shifted linear systems, J. Comput. Math., 25 (2007), pp. 522-530.

[45] I. Moret, P. Novati, An interpolatory approximation of the matrix exponential based on Faber polynomials, J. Comput. Appl. Math., 131 (2001), pp. 361-380.

[46] R.B. Morgan, A restarted GMRES method augmented with eigenvectors, SIAM J. Matrix Anal. Appl., 16(1995), pp. 1154-1171.

[47] M. Heyouni, H. Sadok, A new implementation of the CMRH method for solving dense linear systems, J. Comput. Appl. Math., 213 (2008), pp. 387-399.

[48] H. Sadok, CMRH: A new method for solving nonsymmetric linear systems based on the Hessenberg reduction algorithm, Numer. Algorithms, 20 (1999), pp. 303-321.

[49] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, UK, 1965.

[50] M. Heyouni, Méthode de Hessenberg Généralisée et Applications, Ph.D. Thesis, Université des Sciences et Technologies de Lille, France, 1996.

[51] S. Duminil, M. Heyouni, P. Marion, H. Sadok, Algorithms for the CMRH method for dense linear systems, Numer. Algorithms, in press, 20 May, 2015. Available online at `http://dx.doi.org/10.0.3.239/s11075-015-9997-2`.

[52] H. Sadok, Méthodes de projections pour les systèmes linéaires et non linéaires, Habilitation thesis, University of Lille1, Lille, France, 1994.

[53] H. Sadok, D.B. Szyld, A new look at CMRH and its relation to GMRES, BIT, 52 (2012), pp. 485-501.

[54] K. Zhang, C. Gu, A flexible CMRH algorithm for nonsymmetric linear systems, J. Appl. Math. Comput., 45 (2014), pp. 43-61.

[55] 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`.

[56] G. Howell, D. Stephens, ELMRES, an oblique projection method for solving systems of sparse linear equations, Technical Report, Florida Institute of Technology, March 24, 2000, 18 pages. Available online at `http://ncsu.edu/hpc/Documents/Publications/gary_howell/elmres320.ps`.

[57] D. Stephens, ELMRES: an oblique projection method to solve sparse non-symmetric linear systems, Ph.D. Dissertation, Florida Institute of Technology, USA, August 1999. Available online at `http://ncsu.edu/hpc/Documents/Publications/gary_howell/stephens.pdf`.

[58] T. Davis, Y. Hu, The University of Florida Sparse Matrix Collection, ACM Trans. Math. Softw., 38 (1) (2011), Article 1, 25 pages. Available online at `http://www.cise.ufl.edu/research/sparse/matrices/`.

[59] A. van der Sluis, H.A. van der Vorst, The rate of convergence of conjugate gradients, Numer. Math., 48 (1986), pp. 543-560.

[60] H.A. van der Vorst, C. Vuik, The superlinear convergence behaviour of GMRES, J. Comput. Appl. Math., 48 (1993), pp 327-341.

[61] K. Meerbergen, The solution of parametrized symmetric linear systems, SIAM J. Matrix Anal. Appl., 24 (2003), pp. 1038-1059.

[62] Z.-Z. Bai, M. Benzi, F. Chen, Modified HSS iteration methods for a class of complex symmetric linear systems, Computing, 87 (2010), pp. 93-111.

[63] S. Zhai, X. Feng, Y. He, An unconditionally stable compact ADI method for three-dimensional time-fractional convection-diffusion equation, J. Comput. Phys., 269 (2014), pp. 138-155.

[64] G. Wu, H.-K. Pang, J.-L. Sun, Preconditioning the restarted and shifted block FOM algorithm for matrix exponential computation, arXiv preprint, 24 pages, 4 May 2014. Available online at `http://arxiv.org/abs/1405.0707`.

[65] I. Podlubny, Fractional Differential Equations, Academic Press, San Diego, USA, 1999.

[66] E. Sousa, C. Li, A weighted finite difference method for the fractional differential equation based on the Riemann-Liouville derivative, Appl. Numer. Math., 90 (2015), pp. 22-37.

[67] W.-H. Luo, T.-Z. Huang, L. Li, Y. Zhang, X.-M. Gu, Efficient preconditioner updates for unsymmetric shifted linear systems, Comput. Math. Appl., 67 (2014), pp. 1643-1655.