SCIENCE CHINA
Mathematics

Progress of Projects Supported by NSFC

• ARTICLES •

July 2016 Vol. 59 No. 7: 1235–1248
doi: 10.1007/s11425-016-5142-5

An orthogonally accumulated projection method for symmetric linear system of equations

PENG WuJian[1], LIN Qun[2] & ZHANG ShuHua[3,*]

[1]*Department of Mathematics and Statistics, Zhaoqing University, Zhaoqing* 526061*, China;*
[2]*Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing* 100190*, China;*
[3]*Research Center for Mathematics and Economics, Tianjin University of Finance and Economics, Tianjin* 300204*, China*
Email: wpeng@zqu.edu.cn, linq@lsec.ac.cc.cn, szhang@tjufe.edu.cn

Received September 9, 2015; accepted December 29, 2015; published online April 1, 2016

Abstract A direct as well as iterative method (called the orthogonally accumulated projection method, or the OAP for short) for solving linear system of equations with symmetric coefficient matrix is introduced in this paper. With the Lanczos process the OAP creates a sequence of mutually orthogonal vectors, on the basis of which the projections of the unknown vectors are easily obtained, and thus the approximations to the unknown vectors can be simply constructed by a combination of these projections. This method is an application of the accumulated projection technique proposed recently by the authors of this paper, and can be regarded as a match of conjugate gradient method (CG) in its nature since both the CG and the OAP can be regarded as iterative methods, too. Unlike the CG method which can be only used to solve linear systems with symmetric positive definite coefficient matrices, the OAP can be used to handle systems with indefinite symmetric matrices. Unlike classical Krylov subspace methods which usually ignore the issue of loss of orthogonality, OAP uses an effective approach to detect the loss of orthogonality and a restart strategy is used to handle the loss of orthogonality. Numerical experiments are presented to demonstrate the efficiency of the OAP.

Keywords iterative method, accumulated projection, conjugate gradient method, Krylov subspace

MSC(2010) 65F10, 15A06

Citation: Peng W J, Lin Q, Zhang S H. An orthogonally accumulated projection method for symmetric linear system of equations. Sci China Math, 2016, 59: 1235–1248, doi: 10.1007/s11425-016-5142-5

1 Introduction

The study of iterative methods for solving the linear system of equations in the form

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

where $A \in \mathbb{R}^{n\times n}$ is nonsingular, especially for large scale computing is of vital importance. Recently, all current iterative methods are classified as the extended Krylov subspace methods in [13], which are characterized by their major operations: matrix-vector multiplications with usually one or two fixed matrices and one or two fixed initial vectors. These include the most well-known stationary methods such as Jacobi, Gauss-Seidel as well as SOR methods with their iterative matrices formed on the basis of splitting the coefficient matrices [7] and the row projection methods such as Karcmarz's method and Cimmino's method where the iterative matrices (not explicitly formed in iterations) are constructed by the successive multiplications of a sequence of projection matrices with a fixed sequence length m depending

*Corresponding author

on the splitting of the coefficient matrix into m submatrices [3, 6]. The non-stationary iterative methods are also categorized as the extended Krylov subspace methods, which include the well-known Krylov subspace methods such as the conjugate gradient method (CG) for symmetric positive definite systems, MINRES, SYMMLQ for general symmetric but indefinite systems, and GMRES, BiCG, QMR, LSQR, etc. for general nonsymmetric systems [5, 7, 11, 15, 16]; many of these methods (including GMRES, MINRES, SYMMLQ, MINRES, QMR, LSQR) use the strategy of reducing certain residual errors to search for approximate solutions, while variants of the CG and BiCG methods use the strategy of producing a sequence of orthogonal residuals, thus they can reach the exact solutions with n iterations in exact arithmetic operations, where n is the number of unknowns [7].

It is noticed that current prevalent Krylov subspace methods fall into two categories: the Lanczos-process-based and the Arnoldi-process-based methods. The former includes CG, BiCG, CGS, QMR, SYMMLQ, MINRES, LSQR, BICR; and the later includes GMRES, FOM, IOM, etc. Methods based on the Arnoldi process construct sequences of mutually orthogonal vectors by the Gram-Schmidt process which requires large flop counts and storage requirements if a long sequence of vectors is required, thus a restart technique is often used (like restarted GMRES, FOM) or incomplete orthogonalizing is adopted to avoid the rapidly increasing workload and the storage needs with the price of usually slow convergence; Lanczos-based methods usually converge much faster than Arnoldi-based methods, since the existence of three-term recurrence relations among Lanczos vectors makes it more efficient in constructing sequence of orthogonal vectors and also much fewer storage spaces are needed. The main trouble of this type of methods is the possible failure caused by the loss of orthogonality which usually comes with the Lanczos process. The Arnoldi process and the Lanczos process are also widely used to construct effective iterative schemes for estimating eigenpairs of matrices [9, 10], and their major feature is to use some kind of orthogonal vectors to form low-rank Krylov subspaces so that the corresponding problems can be approximately solved much more easily and usually much faster.

It seems that the major research focuses on linear system in recent years with various preconditioning techniques, which lead to some well-known effective preconditioners such as multigrid method [8], block triangular preconditioner [1] and various incomplete factorization preconditioners [2]. Peng and Lin [13] also present a type of non-Krylov subspace type methods (AP methods) based on the so-called accumulated projection technique. These types of methods rely on successive projections over subspaces of $\mathbb{R}^n$, which produce a sequence of projections of the exact solution vector with a monotonically increasing Euclidean norm. Unlike the well-known row-projection technique which can be shown as a traditional stationary iterative method [6], the AP methods in [13, 14] do not involve matrix-vector multiplications with any fixed matrices and fixed vectors. Equipped with some accelerating technique, the AP methods exhibit some superior behaviors than the traditional extended Krylov subspace methods [13] in some cases.

Our purpose in this paper is to design a class of Krylov subspace methods based on the principle of accumulated projection to solve linear systems of equations. This type of methods not only keeps the efficiency of classical Krylov subspace methods, but also uses the restart strategy to handle the loss of orthogonality, which is generally ignored in classical Krylov subspace methods. For the sake of completeness, we are to briefly review the principle of accumulated projection technique and its applications in the next section. The other sections are devoted to the exploration of the AP technique in a more intricate way which leads to a series of algorithms for solving linear systems.

2 The principle of AP technique

Now we review the basic idea of accumulated projection methods. To approximate any vector x in $\mathbb{R}^n$, one has to construct a subspace W of $\mathbb{R}^n$ with a much smaller rank than n so that a "projection" vector p of x is easily available. Current prevalent methods depend on the strategy of reducing the lengths of residual vectors to obtain such a projection. While only a few methods use the regular orthogonal projection to get approximate vectors, which include the so-called General Error Minimizing method (which is similar to the GMRES method) [4] and the Line Projection method proposed in [12], both can be classified as the extended Krylov subspace methods, since both of them depend on a certain Krylov subspace from which a projection vector is sought. To be able to figure out the projection of x over subspace W, one has to get some "footprint" of x over W, for example, in GMRES-like methods a basis of vectors of W in the form of A^kb with b as the image of x under the transformation A is required, while in GMERR and LP methods, the inner-products between x and a basis of W are available. By this observation we can derive another class of methods for solving linear systems of equations using orthogonal projections.

The basic idea of AP is to use the orthogonal projection of vector x as its approximation, while each projection is used to form another subspace from which a better approximation is sought. Figure 1 can be used to illustrate the basic idea. Here x_i stands for the approximation to x and a_i is the projection vector of x on some subspace of $\mathbb{R}^n$, and x_{i+1} is the projection of vector x in a subspace W_i formed by x_i and a subspace $\tilde{W}_i$, where projection vector a_i of vector x is easily available.

The following algorithm describes a simple implementation of the accumulated projection idea, where vector a_i is orthogonal to vector x_i, and hereinafter we use A' to denote the transpose of matrix A.

Algorithm 1 (Accumulated projection process—AP)**.** The following procedure produces an approximate vector p to the solution vector x which satisfies $Ax = b$.

Step 1. Divide matrix A into k blocks: $A = [A_1', A_2', \ldots, A_k']'$, and divide b correspondingly: $b = (b_1', b_2', \ldots, b_k')'$.

Step 2. Initialize p_0 as $p_0 = \alpha A'b$, and $c_0 = \alpha\|b\|^2$, where $\alpha = \|b\|^2/\|A'b\|^2$.

Step 3. For $i = 1$ to k

Step 3.1. Construct matrix $W_i = [p_{i-1}, A_i']$ and vector $l = [c_{i-1}, b_i']'$.

Step 3.2. Compute the projection vector p_i of x onto subspace $\operatorname{ran}(W_i)$ (the range of matrix W_i) and the scalar c_i $(= x'p_i)$.

Step 4. Output p $(= p_k)$ and c $(= c_k)$.

This algorithm forms the basis of some more efficient solvers for linear systems of equations such as SAP, MSAP, and APAP methods introduced in [13, 14]. It is observed that these methods in general are more efficient than regular Krylov subspace methods in some cases of large scale systems. It is necessary to mention that these methods do not construct any Krylov subspace, and thus cannot be classified as the extended Krylov subspace methods. In this paper, we will show that the AP process can be also used to construct a class of Krylov subspace methods, starting from the so-called orthogonally accumulated projection solver (OAP) that follows in the next section.

3 An orthogonally accumulated projection method

In this section, we will consider to solve System (1.1) with a symmetric coefficient matrix A. The main idea is to transform the original system (1.1) into a system

$$Qx = c, \tag{3.1}$$

where Q is an orthogonal matrix, i.e., $Q'Q = QQ' = I$, where I is the identity matrix. In other words, we will search for a sequence of orthonormal vectors v_i $(i = 1, 2, \ldots, n)$ and real numbers c_i $(i = 1, 2, \ldots, n)$ so that $x'v_i = c_i$, and thus x can be taken as $\sum_{i=1}^n c_i v_i$. In the meantime, we do not have to spend too many extra storage spaces to store all vectors v_i $(i = 1, 2, \ldots, n)$; instead, we will show that a short length recurrence relationship occurs among consecutive orthogonal vectors, so that only a few extra storage spaces for these vectors are needed.

In order to figure out how this will work, let us recall the principle of the AP as illustrated in Figure 1. In general, the sequence of projection vectors a_i comes from some predetermined subspaces, and thus they are not necessary to be orthogonal. However, it is possible for us to work out a way so that all of

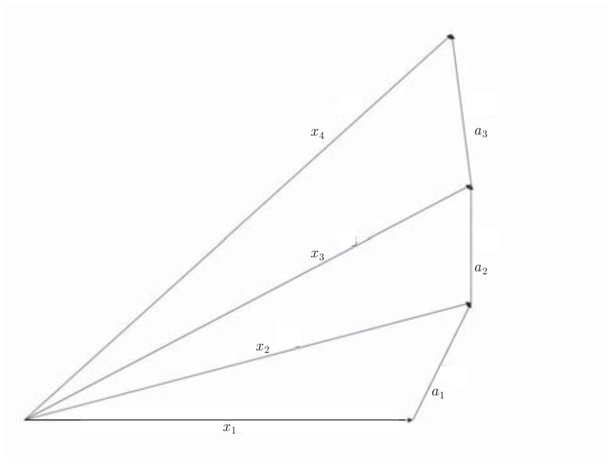


Figure 1 Accumulated projection

these projection vectors a_i $(i = 1, 2, \ldots)$ form an orthogonal sequence. The following algorithm depicts the details.

Algorithm 2 (Orthogonally accumulated projection method—OAP)**.** Let $A \in \mathbb{R}^{n\times n}$ be a symmetric and nonsingular matrix and $b \in \mathbb{R}^n$ a non-zero vector. The following process gives the exact solution x to the system $Ax = b$ (here we assume that there are no break-downs).

```
Step 1 (Initializing). Find a vector $v \neq 0$ so that $x'v$ is known. Let $v_1 = v/\|v\|$, $c_1 = x'v/\|v\|$, $v_0 = 0$, $\beta_1 = 0$, $c_0 = 0$, and $x_1 = c_1 v_1$.
Step 2. For $k = 2$ to $n$
Step 2.1. $w = Av_{k-1}$, $\alpha_{k-1} = v'_{k-1}w$.
Step 2.2. $\tilde{v}_k = w - \alpha_{k-1}v_{k-1} - \beta_{k-1}v_{k-2}$.
Step 2.3. $\beta_k = \|\tilde{v}_k\|$, and $v_k = \tilde{v}_k/\beta_k$.
Step 2.4. $c_k = (b'v_{k-1} - \alpha_{k-1}c_{k-1} - \beta_{k-1}c_{k-2})/\beta_k$.
Step 2.5. $x_k = x_{k-1} + c_k v_k$.
Step 3. Output the solution $x_n$.
```

In this algorithm, we only need storage spaces for three consecutive vectors v_{k-1}, v_k, v_{k+1} instead of the whole orthogonal matrix Q. We also need two extra vectors w and x_k during the process while the quantities: α_k, β_k, and c_k can be stored in a few predefined variables. Thus the storage of this OAP algorithm is quite economic. On the other side, the major flop counts come from only one matrix-vector multiplication in Step 2.1, and this is another advantage of this algorithm.

3.1 The analysis of the OAP

The OAP process involves mainly a key relation among three consecutive vectors: v_{k+1}, v_k, and v_{k-1} in Step 2, which is stated as follows:

$$\beta_k v_{k+1} = Av_k - \alpha_k v_k - \beta_{k-1}v_{k-1} \tag{3.2}$$

for $k = 1, 2, \ldots, n-1$. Bearing this in mind we can show that all vectors $\{v_k\}_1^n$ produced in this way form an orthogonal sequence, i.e.,

$$v'_i v_j = \delta_{ij} = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases} \tag{3.3}$$

This is stated in the following conclusion.

Theorem 3.1. *Let A be symmetric and nonsingular, $b \in \mathbb{R}^n$, and x be the solution to $Ax = b$. The vector sequences v_k $(k = 1, 2, \dots, n)$ produced in Algorithm 2 are orthonormal, and furthermore we have*

$$c_k = x'v_k = \beta_{k-1}^{-1}(b'v_{k-1} - c_{k-1}\alpha_{k-1} - \beta_{k-2}c_{k-1}) \quad \textit{for} \quad k = 2, 3, \dots, n, \tag{3.4}$$

assuming no breakdown happens, i.e., $\beta_k \neq 0$ for any $k = 1, 2, 3, \dots, n-1$.

Proof. First, we show that $v_2'v_1 = 0$. Since $v_0 = 0$, from (3.2) we have

$$\beta_1 v_2 = Av_1 - \alpha_1 v_1.$$

Multiplying both sides by v_1', we have $\beta_1 v_1'v_2 = v_1'Av_1 - \alpha_1 v_1'v_1 = 0$, since $v_1'v_1 = 1$ and $\alpha_1 = v_1'Av_1$. This gives $v_1'v_2 = 0$, since $\beta_1 \neq 0$.

Let $m \in \{3, 4, \dots, n\}$. Assume that $v_i'v_m = 0$ for any $i < m$ holds true. We now show that for $k = m+1$, $v_i'v_k = 0$ is also true for any $i < k$. Thus, by induction we have (3.3). In fact, we know from (3.2) that

$$\begin{aligned} \beta_m v_i'v_{m+1} &= v_i'Av_m - \alpha_k v_i'v_m - \beta_{m-1}v_i'v_{m-1} \\ &= v_i'Av_m - \alpha_k\delta_{i,m} - \beta_{m-1}\delta_{i,m-1}. \end{aligned} \tag{3.5}$$

If $i = m$, by (3.5) we obtain

$$\beta_m v_m'v_{m+1} = v_m'Av_m - \alpha_m = 0,$$

which leads to $v_m'v_{m+1} = 0$, since $\beta_m \neq 0$.

If $i = m-1$, by (3.5) and (3.2) we also have

$$\begin{aligned} v_{m-1}'v_{m+1} &= \beta_m^{-1}(v_{m-1}'Av_m - \beta_{m-1}) \\ &= \beta_m^{-1}(v_m'Av_{m-1} - \beta_{m-1}) \\ &= \beta_m^{-1}(v_m'(\beta_{m-1}v_m - \alpha_{m-1}v_{m-1} - \beta_{m-2}v_{m-2}) - \beta_{m-1}) \\ &= 0, \end{aligned}$$

since $A' = A$. If $i < m-1$, by (3.2) we gain

$$Av_i = \beta_i v_{i+1} + \alpha_i v_i + \beta_{i-1}v_{i-1}, \tag{3.6}$$

and by (3.5) we further have

$$v_i'v_{m+1} = \beta_m^{-1}v_i'Av_m = \beta_m^{-1}v_m'Av_i = \beta_m^{-1}v_m(\beta_i v_{i+1} + \alpha_i v_i + \beta_{i-1}v_{i-1}) = 0.$$

On the basis of the fact that $x'A = (Ax)' = b$, (3.4) comes directly from (3.2) and the definition $c_k = x'v_k$. □

3.2 Breakdowns in the OAP process

In this subsection, we show in what situation breakdown will happen and how to handle this problem, so that the OAP will be finished. As we mentioned previously, a breakdown happens if $\beta_k = 0$ for some integer k and $\beta_i \neq 0$ for $i < k$. In fact, a breakdown signals an invariant subspace of A, as stated in the following conclusion.

Theorem 3.2. *Let $\beta_k = 0$, and $\beta_i \neq 0$ for $i < k$. Then, the vector sequences $\{v_i\}_1^k$ form the basis of an invariant subspace W of linear transformation induced by matrix A, i.e., $AW \subset W$.*

Proof. Letting v $(= \sum_{i=1}^k s_i v_i) \in W = \text{span}\ \{v_1, v_2, \dots, v_k\}$, we need to show that $Av \in W$. As a matter of fact, $Av = \sum_{i=1}^{k-1} s_i Av_i + s_k Av_k = u_1 + s_k u_2$, where $u_1 = \sum_{i=1}^{k-1} s_i Av_i$, and $u_2 = Av_k$. By (3.6) we have $u_1 \in \text{span}\{v_1, v_2, \dots, v_k\}$, and $u_2 \in \text{span}\{v_{k-1}, v_k\}$, since $\beta_k = 0$. Thus, both u_1 and u_2 belong to W, which implies that we always have $v \in W$. □

Unlike a breakdown in the GMRES which means a happy ending, in OAP this does not necessarily mean the appearance of an exact solution. Instead a breakdown in the OAP depends on the choice of its initial vector v_1. The rest of this subsection is devoted to handling breakdowns. For the sake of simplicity, we call β_k $(=0)$ as our breakdown point if $\beta_i \neq 0$ for $i < k$. First, let us show some conclusion that may sound surprising.

Theorem 3.3. *Let β_k be the breakdown point in Algorithm* 2*, $r_k = b - Ax_k$ be the related residual vector, and $W = span\,\{v_1, v_2, \dots, v_k\}$. Then* (i) *$W^\perp$ is also an invariant subspace of A, and* (ii) *$r_k' v_i = 0$ for $i = 1, 2, \dots, k$.*

Proof. Let $W = \operatorname{span}\{v_1, v_2, \dots, v_k\}$ be the invariant subspace associated with the breakdown point β_k. Let $v_{k+1}, v_{k+2}, \dots, v_n$ be an orthogonal basis of $W^\perp$, the complementary subspace of W in $\mathbb{R}^n$. To see that $W^\perp$ is an invariant subspace of A, we have to show that for any $v \in W^\perp$, there holds $Av \in W^\perp$. As a matter of fact, for any $u \in W$, let $\tilde{u} = Au$, then $\tilde{u} \in W$, since W is an invariant subspace of A. Therefore, we have

$$u'Av = v'Au = v'\tilde{u} = 0,$$

which means that Av is orthogonal to any vectors in W. Hence, we must have $Av \in W^\perp$.

Apparently, vectors $v_1, v_2, \dots, v_n$ form an orthonormal basis of $\mathbb{R}^n$ and thus there exists a sequence of real numbers c_i $(i = 1, 2, \dots, n)$, such that $x = \sum_{i=1}^n c_i v_i$ and obviously we have $x_k = \sum_{i=1}^k c_i v_i$.

Note that $r_k = b - Ax_k = \sum_{i=k+1}^n c_i A v_i$, thus $r_k \in W^\perp$, which leads to (ii). □

Based on the above discussion, we now show how to handle breakdowns in the OAP process described in Algorithm 2. Actually, when r_k is not a zero vector, we only need to set v_{k+1} as $v_{k+1} = \alpha A r_k$, where α is a scalar such that v_{k+1} is a unit vector. Obviously, we have $c_{k+1} = x' v_{k+1} = \alpha b' r_k$. Note that if r_k is zero vector, there is no necessity to move on to the next vector v_{k+1}. Thus, a modified version of the OAP is stated as the following algorithm.

Algorithm 3 (An iterative orthogonally accumulated projection method—iterative OAP)**.** Let A be a symmetric and nonsingular matrix and $b \in \mathbb{R}^n$ a non-zero vector. Let ϵ be a given tolerance. The following process gives the exact solution x to the system $Ax = b$.

```
Step 1 (Initializing). Find a vector $v \neq 0$ so that $x'v$ is known. Let $v_1 = v/\|v\|$, $c_1 = x'v/\|v\|$, $v_0 = 0$, $\beta_1 = 0$, $c_0 = 0$, and $x_1 = c_1 v_1$.
Step 2. Set $k = 1$, $r_k = b - Ax_k$, and $c = c_1^2$.
Step 3. While $\|r_k\| > \epsilon$
Step 3.1. $k = k + 1$, $w = Av_{k-1}$, $\alpha_{k-1} = v_{k-1}' w$.
Step 3.2. $\tilde{v}_k = w - \alpha_{k-1} v_{k-1} - \beta_{k-1} v_{k-2}$.
Step 3.3. $\beta_k = \|\tilde{v}_k\|$.
Step 3.4. If $\beta_k \neq 0$, set $v_k = \tilde{v}_k/\beta_k$, and $c_k = (b'v_{k-1} - \alpha_{k-1}c_{k-1} - \beta_{k-1}c_{k-2})/\beta_k$; else set $w = Ar_k$, $\alpha = \|w\|$, and $v_k = w/\alpha$, $c_k = \alpha b' r_k$.
Step 3.5. $x_k = x_{k-1} + c_k v_k$, $r_k = b - Ax_k$, $c = c + c_k^2$.
Step 4. Output the solution $x_k$ (and $c$ if needed).
```

Note that in this algorithm we set up an extra quantity c to store the sum of c_k^2 $(k = 1, 2, \dots, n)$. It is easy to see that $c = x'x_k$ for each k. The purpose of this action is clear when we discuss the expansibility of the AP technique.

Compared with Algorithm 2, there are two matrix-vector multiplications needed in each loop (in Steps 2.1 and 2.5) in this algorithm. It is possible for us to alleviate part of the flops by reducing one matrix-vector multiplication based on the relation between the residual vector and the three consecutive orthogonal vectors stated in the following theorem.

Theorem 3.4. *Let $r_k = b - Ax_k$ be the residual vector associated with approximation vector x_k in Algorithm 3 and let $\Delta r_k = r_{k-1} - r_k$. Then*

$$\beta_k v_{k+1} = c_k^{-1}\Delta r_k - \alpha_k v_k - \beta_{k-1} v_{k-1}. \tag{3.7}$$

Proof. Since $x_k = \sum_{i=1}^k c_i v_i$, we have

$$r_k = b - Ax_k = b - A(x_{k-1} + c_k v_k) = (b - Ax_{k-1}) - c_k A v_k = r_{k-1} - c_k A v_k,$$

which leads to

$$Av_k = c_k^{-1}\Delta r_k. \tag{3.8}$$

Combining (3.6) with (3.8) implies that

$$c_k^{-1}\Delta r_k = \beta_k v_{k+1} + \alpha_k v_k + \beta_{k-1} v_{k-1},$$

which yields (3.7) directly. □

3.3 The connection between the OAP and the Lanczos tridiagonal procedure

It is well known that any symmetric matrix A can be converted to a symmetric three-diagonal matrix by an orthogonal transformation, i.e., there exists an orthogonal matrix V such that

$$V'AV = T \quad \text{or} \quad AV = VT, \tag{3.9}$$

where T is a symmetric tridiagonal matrix. Let V_k be the $n \times k$ matrix formed by the first k columns of V, and T_k be the k-th principal submatrix of T formed by the first k rows and first k columns of T, i.e., $V_k = [v_1, v_2, \dots, v_k]$ and T_k has the form

$$T_k = \begin{pmatrix} \alpha_1 & \beta_1 & 0 & \cdots & 0 & 0 \\ \beta_1 & \alpha_2 & \beta_2 & \cdots & 0 & 0 \\ 0 & \beta_2 & \alpha_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_{k-1} & \beta_{k-1} \\ 0 & 0 & 0 & \cdots & \beta_{k-1} & \alpha_k \end{pmatrix}. \tag{3.10}$$

Then, we have by (3.9) that

$$AV_k = V_k T_k + \beta_k v_{k+1} e_k',$$

where $e_k \in \mathbb{R}^k$ is the last column of the identity matrix $I \in \mathbb{R}^{k\times k}$, or comparing each column of (3.9) we have

$$Av_k = \beta_k v_{k+1} + \alpha_k v_k + \beta_{k-1} v_{k-1},$$

which is nothing but the three-term recurrence relation (3.2). A Lanczos process begins with a given unit vector v_1 and then searches the rest of the orthonormal basis by equating each column of (3.9), which ends up exactly the major part of our OAP process.

It is well known that the Lanczos process suffers from the loss of orthogonality in general, especially when A is ill-conditioned. Our next section is thus devoted to discussing the strategies to handle this issue.

4 The control of orthogonality

As we have already mentioned, since the conjugate gradient method is an implementation of the Lanczos tridiagonalization process, the OAP can be also regarded as such a process. It is thus necessary for us to treat the loss of orthogonality with caution. In this section, we present some approaches to tackle this task.

First we need to check at each iteration whether or not the loss of orthogonality happens. Note that in exact arithmetic operations, the approximation vector x_k in the OAP process should be orthogonal to each new unit vector v_{k+1}, thus we can calculate at each iteration the angle θ between vectors x_k and v_{k+1}. If θ is away from 90 degrees, this means a loss of orthogonality happens. One can reset v_1 as x_k and restart the OAP process again; however, a more reliable approach seems to be simply restarting the OAP process on the residual equation defined as

$$Ae_k = r_k, \tag{4.1}$$

where $e_k = x - x_k$ is the unknown here and $r_k = b - Ax_k$. The details come as follows.

Algorithm 4 (An iterative orthogonally accumulated projection method—iterative OAP version 2)**.** Let A be a symmetric and nonsingular matrix and $b \in \mathbb{R}^n$ a non-zero vector. Let ϵ be a given tolerance. The following process gives an approximate solution x_k to the system $Ax = b$.

Step 1 (Initializing)**.** Find a vector $v \neq 0$ so that $x'v$ is known. Let $v_1 = v/\|v\|$, $c_1 = x'v/\|v\|$, $v_0 = 0$, $\beta_1 = 0$, $c_0 = 0$, and $x_1 = c_1 v_1$.

Step 2. Set $k = 1$, $r_k = b - Ax_k$, and $c = c_1^2$.

Step 3. While $\|r_k\| > \epsilon$

Step 3.1. $k = k + 1$, $w = Av_{k-1}$, $\alpha_{k-1} = v'_{k-1}w$.

Step 3.2. $\tilde{v}_k = w - \alpha_{k-1}v_{k-1} - \beta_{k-1}v_{k-2}$.

Step 3.3. $\beta_k = \|\tilde{v}_k\|$.

Step 3.4. If $\beta_k \neq 0$, set $v_k = \tilde{v}_k/\beta_k$, and $c_k = (b'v_{k-1} - \alpha_{k-1}c_{k-1} - \beta_{k-1}c_{k-2})/\beta_k$; else set $w = Ar_k$, $\alpha = \|w\|$, and $v_k = w/\alpha$, $c_k = \alpha b' r_k$.

Step 3.5. Calculate $\theta = \text{arcos}\,(x'v_k/\sqrt{c})$.

Step 3.6. If $\theta = \pi/2$, set $x_k = x_{k-1} + c_k v_k$, $r_k = b - Ax_k$, and $c = c + c_k^2$; else stop the loop.

Step 4. Output the solution x_k and c.

Remark 4.1. In actual implementation, θ cannot be exactly 90 degrees, thus a small tolerance should be set instead. For example, one can change the condition as $(\theta - \pi/2) < \tau$, where τ is a small positive number far less than 1.

The above algorithm is a modified version of Algorithm 3, and it simply adds one step to check the loss of orthogonality. Again the quantity c is the inner-product between x and x_k. The next algorithm then uses Algorithm 4 and restarts the OAP each time a loss of orthogonality happens.

Algorithm 5 (A progressive orthogonally accumulated projection method—POAP)**.** Let A be a symmetric and nonsingular matrix and $b \in \mathbb{R}^n$ a non-zero vector. Let ϵ be a given tolerance. The following process gives an approximate solution x to the system $Ax = b$.

Step 1 (Initializing)**.** Set $x = 0$, $r = b$.

Step 2. While $\|r\| > \epsilon$

Step 2.1. Call Algorithm 4 to solve $Ay = r$, and let y_k be the output approximation.

Step 2.2. Update $x = x + y_k$.

Step 2.3. Calculate $r = b - Ax$.

Step 3. Output the solution x.

5 Generalization

In this section, we generalize the OAP method to solve systems with unsymmetric coefficient matrices, and we also talk about some strategies for accelerating the OAP methods.

5.1 The OAP for systems with unsymmetric coefficient matrix

Just like the CGNE and the CGNR to the CG, the same approach can be used to get corresponding OAP versions, namely the OAPNE and the OAPNR for unsymmetric systems.

The basic idea of the OAPNR is to multiply both sides of (1.1) by A' if A is square but not symmetric, and then apply the OAP process to the equivalent system $A'Ax = A'b$. In practical implementation, the matrix $A'A$ is never explicitly formed. The details come as follows.

Algorithm 6 (An orthogonally accumulated projection on normal equation residual method—OAPNR)**.** Let A be a symmetric and nonsingular matrix and $b \in \mathbb{R}^n$ a non-zero vector. Let ϵ be a given tolerance. The following process gives an approximate solution x_k to the system $Ax = b$.

Step 1 (Initializing)**.** Find a vector $v \neq 0$ so that $x'v$ is known. Let $v_1 = v/\|v\|$, $c_1 = x'v/\|v\|$, $v_0 = 0$, $\beta = \|b\|$, $\beta_1 = 0$, $c_0 = 0$, and $x_1 = c_1 v_1$.

Step 2. Set $k = 1$, $r_k = A'(b - Ax_k)$.

Step 3. While $\|r_k\| > \epsilon\beta$

Step 3.1. $k = k + 1$, $w = A'(Av_{k-1})$, $\alpha_{k-1} = v'_{k-1}w$.

Step 3.2. $\tilde{v}_k = w - \alpha_{k-1}v_{k-1} - \beta_{k-1}v_{k-2}$.

Step 3.3. $\beta_k = \|\tilde{v}_k\|$.

Step 3.4. If $\beta_k \neq 0$, set $v_k = \tilde{v}_k/\beta_k$, and $c_k = (b'Av_{k-1} - \alpha_{k-1}c_{k-1} - \beta_{k-1}c_{k-2})/\beta_k$; else set $w = A'(Ar_k)$, $\alpha = \|w\|$, and $v_k = w/\alpha$, $c_k = \alpha b'r_k$.

Step 3.5. Calculate $\theta = \arccos\,(x'_k v_k/\|x_k\|)$.

Step 3.6. If $\theta = \pi/2$, set $x_k = x_{k-1} + c_k v_k$, $r_k = A'(b - Ax_k)$; else stop the loop.

Step 4. Output the solution x_k.

Another approach to handle System (1.1) with $A \in \mathbb{R}^{n\times m}$ unsymmetric is to solve the equation $AA'y = b$ first, and then obtain x by letting $x = A'y$. Again, it is not necessary to explicitly form matrix AA' unless A is sparse. The following algorithm gives the details.

Algorithm 7 (An orthogonally accumulated projection on normal equation error method—OAPNE)**.** Let $A \in \mathbb{R}^{n\times m}$ in (1.1) be an unsymmetric matrix and $b \in \mathbb{R}^n$ a non-zero vector with $n \leqslant m$. Let ϵ be a given tolerance. The following process gives an approximate solution x to the system $Ax = b$.

Step 1 (Initializing)**.** Find a vector $v \neq 0$ so that $x'Av$ is known. Let $v_1 = v/\|v\|$, $c_1 = x'Av/\|v\|$, $v_0 = 0$, $\beta = \|b\|, \beta_1 = 0$, $c_0 = 0$, and $y_1 = c_1 v_1$.

Step 2. Set $k = 1$, and $r_k = b - A(A'y_k)$.

Step 3. While $\|r_k\| > \epsilon\beta$

Step 3.1. $k = k + 1$, $w = A(A'v_{k-1})$, $\alpha_{k-1} = v'_{k-1}w$.

Step 3.2. $\tilde{v}_k = w - \alpha_{k-1}v_{k-1} - \beta_{k-1}v_{k-2}$.

Step 3.3. $\beta_k = \|\tilde{v}_k\|$.

Step 3.4. If $\beta_k \neq 0$, set $v_k = \tilde{v}_k/\beta_k$, and $c_k = (b'v_{k-1} - \alpha_{k-1}c_{k-1} - \beta_{k-1}c_{k-2})/\beta_k$; else set $w = A(A'r_k)$, $\alpha = \|w\|$, and $v_k = w/\alpha$, $c_k = \alpha b'r_k$.

Step 3.5. Calculate $\theta = \arccos\,(y'_k v_k/\|y_k\|)$.

Step 3.6. If $\theta = \pi/2$, set $y_k = y_{k-1} + c_k v_k$, $r_k = b - AA'y_k$; else stop the loop.

Step 4. Output the solution $x_k = A'y_k$.

Remark 5.1. Again in the actual implementation, θ in Algorithms 6 and 7 cannot be exactly 90 degrees, thus a small tolerance should be set instead. For example, one can change the condition as $|\theta - \pi/2| < \tau$, where τ is a small positive number far less than 1.

Both the OAPNE and the OAPNR methods suffer from numerical instability, since they essentially use the OAP to solve the system $AA'y = Ab$ or $A'Ax = A'Ab$, and the condition number of its coefficient matrix is the square of cond$\,(A)$. However, by using the same strategy as that applied in Algorithm 5, one can get improved numerical behavior of these two approaches.

5.2 The expansibility of the AP technique

One of the major features of the AP technique, which makes it more attractive than other projection techniques, is its expansibility, i.e., this technique actually helps us to expand the original linear system to a system with more equations. In other words, System (1.1) can be expanded into

$$\tilde{A}x = \tilde{b}, \tag{5.1}$$

where $\tilde{A} \in \mathbb{R}^{(n+k)\times n}$ contains A as its submatrix and $\tilde{b} \in \mathbb{R}^{n+k}$ also contains all of b's elements while x keeps unchanged. This expansion provides us with great opportunity to search a better approximation to x, or speeds up the approximating process. Two accelerative approaches of such a type are introduced in [13, 14]. In this section, we will use an approach similar to that in [13] to accelerate our POAP algorithm.

Let $x \in \mathbb{R}^n, v \in \mathbb{R}^n$, and $c \in R$. For the sake of simplicity, we call the pair (v, c) as an AP projection of x if $c = x'v$. The following conclusion provides a foundation of accelerative scheme derived from AP technique.

Theorem 5.1. *Let $x \in \mathbb{R}^n$ be the solution to the system* (1.1) *with A symmetric, (x_k, c_k) be an AP projection of x, and (p_k, d_k) be an AP projection of e_k with $e_k = x - x_k$. Then, $(\tilde{x}_k, \tilde{c}_k)$ is also an AP projection of x with $\tilde{x} = x_k + p_k$ and $\tilde{c}_k = c_k + x_k' p_k + d_k$.*

Proof. We need to show that

$$\tilde{x}' x_k = \tilde{c}_k.$$

Note that

$$x'\tilde{x}_k = x'(x_k + p_k) = x'x_k + x'p_k = x'x_k + (x_k + e_k)'p_k = x'x_k + x_k'p_k + e_k'p_k. \tag{5.2}$$

Since (x_k, c_k) and (p_k, d_k) are the AP projections of x and e_k respectively, we have $x'x_k = c_k$ and $e_k'p_k = d_k$. Plugging them into (5.2), we have immediately that $x'\tilde{x}_k = c_k + x_k'p_k + d_k = \tilde{c}_k$. □

Based on this observation, we can further improve the performance of the POAP by inserting an accelerating step. The following algorithm gives one of such strategies.

Algorithm 8 (An accelerated progressive orthogonally accumulated projection method)**.** Let A be a symmetric and nonsingular matrix and $b \in \mathbb{R}^n$ a non-zero vector. Let ϵ be a given tolerance. The following process produces x_k approximate to the exact solution x to the system $Ax = b$.

```
Step 1 (Initializing).  Set $x_0 = 0$, $r = b$, $c_0 = 0$, and $k = 0$.
Step 2.  While $\|r\| > \epsilon$
Step 2.1.  $k = k + 1$. Call Algorithm 4 to get the AP projection $(y_k, d_k)$ of $y$ which satisfies $Ay = r$.
Step 2.2.  Calculate $c_k = c_{k-1} + x_k' y_k + d_k$.
Step 2.3.  Update $\tilde{x}_k = x_{k-1} + y_k$.
Step 2.4.  Update $x_k$ as the AP projection of $x$ over subspace span$\{\tilde{x}_k, x_{k-1}\}$.
Step 2.5.  Calculate $r = b - Ax_k$.
Step 3.  Output the solution $x_k$.
```

6 Numerical experiments

In this section, we will examine the numerical behaviors of the orthogonally accumulated projection methods proposed in the previous sections. The linear systems of equations tested here are generated from numerical partial differential equations.

As the first example, we consider the regular one dimensional two-point boundary value problem as follows:

$$-\frac{\partial}{\partial t}\left(p(t)\frac{\partial x(t)}{\partial t}\right)+q(t)x(t)=f(t), \quad t\in(0,1),$$

with $x(0)=x(1)=0$. We use the finite element method to discretize the problem, and the basis functions used here are hierarchical bases. The resulted coefficient matrices are of the form shown in Figure 2. In Figure 2(a), the initial number of subintervals is taken as 5, and the mesh is refined eight times so that the total number of grids is $5*2^8-1$ $(=1279)$; while in Figure 2(b) the initial number of subintervals is taken as 20, and the mesh is refined five times, and thus the total number of grids is $20*2^5-1$ $(=639)$.

Table 1 shows the comparison of the relative error $\|x-\tilde{x}\|/\|x\|$ among the POAP and some other prevalent Krylov subspace methods, where x and $\tilde{x}$ are the exact solution and the approximate solutions obtained by using different iterative methods, and the stopping criteria is set to be

$$\|b-A\tilde{x}\|/\|b\|<10^{-6},$$

and n is the number of unknowns. In these tests, we use the following settings: $p(t)=t$, $q(t)=1$, and $f(t)$ is taken such that the real solution $x(t)$ has the form $x(t)=\mathrm{e}^{(3-t)}\sin(\pi t)$. It can be seen from Table 1 that the POAP method usually produces more accurate approximate solutions in terms of relative errors.

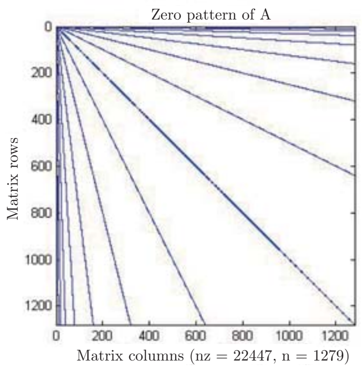
In the second experiment, we use a regular two-dimensional BVP problem

$$\Delta u=f \tag{6.1}$$

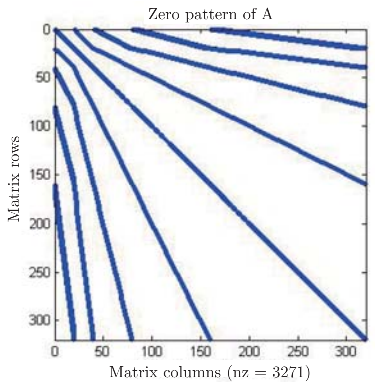
defined on an L-shaped domain

$$[0,1]\times\left[0,\frac{1}{2}\right]\cup\left[0,\frac{1}{2}\right]\times\left[\frac{1}{2},1\right],$$

where $f(t,s)$ is taken so that the exact solution is $u(t,s)=\sin(\pi t)\sin(\pi s)$. We use the five-point finite difference scheme to discretize the problem, and the zero patterns of the resulting coefficient matrices are shown in Figures 3. The comparison of the relative errors among the OAP and other Krylov subspace methods are shown in Table 2. Again one can see that the approximate solutions obtained by the POAP show better relative errors than other Krylov subspace methods in general.



(a) The distribution of non-zero elements of A when the initial mesh grid is 5 points

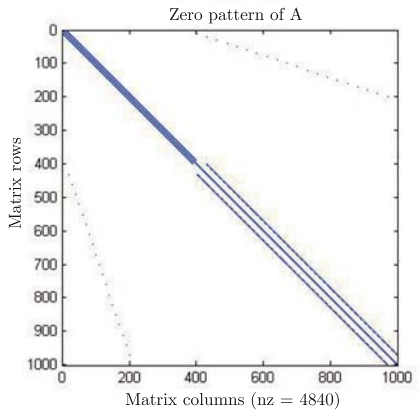


(b) The distribution of non-zero elements of A when the initial mesh grid is 20 points

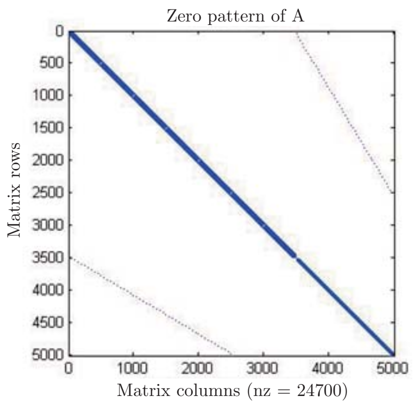
Figure 2 Example 1: The pattern of non-zero elements distribution

Table 1 The comparison of relative errors

n	POAP	MINRES	LSQR	CG	SYMMLQ	QMR	BICG
31	5.5227e−7	9.5420e−7	1.3502e−7	9.0996e−7	9.0996e−7	9.5420e−7	9.0996e−7
63	4.8098e−7	6.9631e−6	1.0474e−5	8.7554e−7	8.7562e−7	6.9633e−6	8.7547e−7
127	1.2362e−6	8.4231e−6	3.5963e−6	3.3800e−6	3.3028e−6	8.4226e−6	3.3008e−6
255	2.6426e−6	1.0753e−5	2.2577e−5	7.9867e−6	7.9867e−6	1.0753e−5	7.9867e−6
511	1.8415e−6	3.8084e−5	7.5859e−5	2.0105e−5	2.0105e−5	3.8085e−5	2.0105e−5
1023	1.0641e−5	1.3964e−4	1.6183e−4	5.1029e−5	5.1029e−5	1.3964e−4	5.1029e−5
2047	1.7943e−6	6.1468e−4	3.3880e−4	9.5182e−5	9.5182e−5	6.1468e−4	9.5181e−5



(a) The distribution of non-zero elements of A



(b) The distribution of non-zero elements of A

Figure 3 Example 2: The pattern of non-zero elements distribution

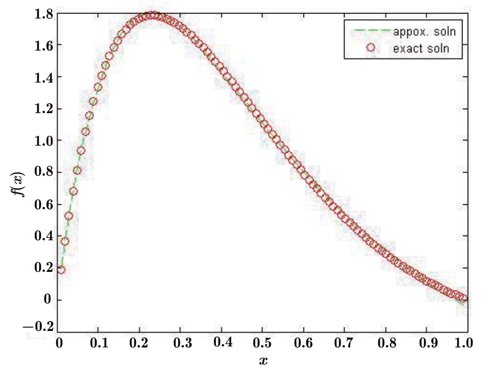
Table 2 The comparison of the relative errors

n	POAP	MINRES	LSQR	CG	SYMMLQ	QMR	BICG
300	8.8078e−8	7.5369e−7	8.7918e−7	4.0907e−7	4.0907e−7	7.5369e−7	4.0907e−7
1000	5.6365e−7	3.4215e−6	1.7742e−6	1.2624e−6	1.2624e−6	3.4215e−6	1.2624e−6
1800	2.6799e−7	4.0848e−6	2.9232e−6	1.7059e−6	1.7059e−6	4.0848e−6	1.7059e−6
2400	3.0539e−7	7.6561e−6	2.3775e−6	1.7835e−6	1.7835e−6	7.6561e−6	1.7835e−6
5000	2.5900e−7	7.4162e−6	4.1379e−6	2.7286e−6	2.7286e−6	7.4162e−6	2.7286e−6
6800	2.0703e−7	9.6869e−6	1.3799e−5	1.9012e−6	1.9012e−6	9.6869e−6	1.9012e−6

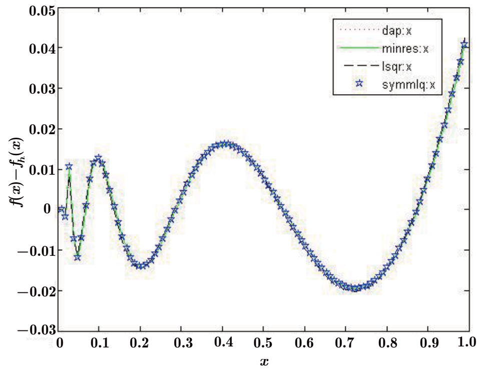
In order to check the behaviors of the POAP method on systems with extremely ill-conditioned coefficient matrices, we use the Hilbert matrices in the third experiment. The exact solution is the values of function $f(x)$ at grid points $x_i = ih$, $h = 1/(n+1)$ $(i = 1, 2, \dots, n)$ with

$$f(x) = x(1-x)\mathrm{e}^{3(1-x)}.$$

Figure 4(a) shows the exact solution and the approximated solutions by the different Krylov subspace methods, and Figure 4(b) shows the difference between the exact solution and the approximated solutions. One can see in this case that all iterative methods produce quite accurate approximate solutions. The relative errors are shown in Table 3.



(a) The exact solution and approximate solutions



(b) The difference between exact solution and approximate solutions

Figure 4 Systems with Hilbert matrices

Table 3 The comparison of relative errors

n	POAP	MINRES	LSQR	CG	SYMMLQ	QMR	BICG
500	0.004730	0.012670	0.013636	0.012663	0.012663	0.012679	0.012663
1000	0.011740	0.014192	0.015387	0.014189	0.014189	0.014195	0.014189
1500	0.013727	0.012885	0.013685	0.012883	0.012883	0.012885	0.012883
2000	0.013220	0.013324	0.014098	0.013323	0.013323	0.013324	0.013323
2500	0.013571	0.013895	0.015126	0.013891	0.013891	0.013893	0.013891
3000	0.013701	0.014611	0.013697	0.014601	0.014601	0.014619	0.014602

7 Concluding remarks

It should be mentioned that according to its construction, the OAP methods presented in this paper still belong to the so-called extended Krylov subspace methods, since they rely on the construction of a Krylov subspace based on the matrix-vector multiplication between the coefficient matrix A and a certain initial vector. In addition, the approximate solutions also come from this Krylov subspace. Unlike the most other Krylov subspace methods where the approximation vectors are searched under some criteria of reducing residual norms, the OAP algorithms depend on the construction of an orthonormal basis of $\mathbb{R}^n$, while this is efficiently carried out by the three-term recurrence relation among the basis vectors and the "coordinates" of the sought-after solution under this sequence of orthonormal vectors are also calculated economically. Therefore, in exact arithmetic operations the OAP will produce the solution after n iterations at most.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 91430108 and 11171251) and the Major Program of Tianjin University of Finance and Economics (Grant No. ZD1302).

References

1 Bai Z Z. Rotated block triangular preconditioning based on PMHSS. Sci China Math, 2013, 56: 2523–2538
2 Benzi M. Preconditioning techniques for large linear systems: A survey. J Comput Phys, 2002, 182: 418–477
3 Bramley R, Sameh A. Row projection methods for large nonsymmetric linear systems. SIAM J Sci Comput, 1992, 13: 168–193
4 Ehrig R, Deuflhard P. GMERR—an error minimizing variant of GMRES. Technical Report SC-97-63, ZIB, 1997

5 Freund R W, Nachtigal N M. QMR: A quasi-minimal residual method for non-Hermitian linear systems. Numer Math, 1991, 60: 315–339

6 Galäntai A. Projectors and Projection Methods. Berlin: Springer, 2004

7 Golub G H, Van Loan C F. Matrix Computations. Baltimore-London: The Johns Hopkins University Press, 1996

8 Hackbusch W. Multi-Grid Methods and Applications. Berlin: Springer-Verlag, 1985

9 Jia Z X. Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems. Linear Algebra Appl, 1997, 259: 1–23

10 Jia Z X. On convergence of the inexact Rayleigh quotient iteration with the Lanczos method used for solving linear systems. Sci China Math, 2013, 56: 2145–2160

11 Paige C C, Saunders M A. Solution of sparse indefinite systems of linear equations. SIAM J Numer Anal, 1975, 12: 617–629

12 Peng W. A line-projection method for solving linear system of equations. Pacific J Appl Math, 2013, 5: 17–28

13 Peng W, Lin Q. A non-Krylov subspace method for solving large and sparse linear system of equations. Numer Math Theor Meth Appl, 2016, 9: 289–314

14 Peng W, Zhang S. A stationary accumulated projection method for linear system of equations. ArXiv:1603.05356, 2016

15 Saad Y. Iterative Methods for Sparse Linear Systems, 2nd ed. Philadelphia: SIAM, 2003

16 van der Vorst H A. Iterative Krylov Methods for Large Linear Systems. Cambridge: Cambridge University Press, 2003