INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2004; **60**:1383–1399 (DOI: 10.1002/nme.983)

Study on semi-conjugate direction methods for non-symmetric systems

Yu-Hong Dai[1,‡] and Jinyun Yuan[2,*,†]

[1]*State Key Laboratory of Scientific and Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, P.O. Box 2719, Beijing 100080, China*
[2]*Departamento de Matemática, Universidade Federal do Paraná, Centro Politécnice, CP: 19.081, CEP: 81531-990, Curitiba, Brazil*

SUMMARY

Some theoretical problems and implementation problems are studied here for the semi-conjugate direction method established by Yuan, Golub, Plemmons and Cecilio (2002). The existence of semi-conjugate directions is proved for almost all matrices except skew-symmetric matrices. A new technique is proposed to overcome the breakdown problem appeared in the semi-conjugate direction method. In the implementation of the semi-conjugate direction method, the generation of the semi-conjugate direction is very important and necessary, but very expensive. The technique of limited-memory is introduced to economize the cost of the generation of the semi-conjugate direction in the Yuan–Golub–Plemmons–Cecilio algorithm. Finally, some numerical experiments are given to confirm our theoretical results. Our results illustrate that the semi-conjugate direction method is very nice alternative for solving non-symmetric systems, and the limited-memory left conjugate direction method is a good improvement of the left conjugate direction method. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: non-singular matrix; linear system; semi-conjugate direction; left conjugate direction; left conjugate direction method; left conjugate direction method; existence; breakdown; limited-memory left conjugate direction method; conjugate gradient method

1. INTRODUCTION

Consider the non-symmetric linear system

$$Ax = b \tag{1}$$

*Correspondence to: Jinyun Yuan, Departamento de Matemática, Universidade Federal do Paraná, Centro Politécnice, CP: 19.081, CEP: 81531-990, Curitiba, Brazil.
†E-mail: jyuan@cs.ubc.ca
‡E-mail: dyh@lsec.cc.ac.cn

Contract/grant sponsor: Chinese NSF; contract/grant number: 19801033 and 10171104

Received 6 August 2002
Revised 28 May 2003
Accepted 5 August 2003

where $A \in R^{n\times n}$ is non-singular, $b \in R^n$ is a vector. To solve (1), a class of iterative methods called semi-conjugate direction (SCD) methods was proposed in Reference [1]. The methods can be regarded as some kind of extension of conjugate direction methods for symmetric and positive definite linear systems. Meanwhile, similarly to References [2, 3], the relationship between semi-conjugate direction methods and the LU decomposition was discussed in Reference [1]. The preliminary numerical results in the paper showed that one member of the methods, called left conjugate direction (LCD) method, is very promising for solving non-symmetric systems. Recently, Silva *et al*. [4] have applied the LCD method to solve linear systems arising from Petrov–Galerkin finite element method for the thermal pollution problem, and compared with traditional methods. From their comparisons, the LCD method has nice performance. However, there are still at least three important questions related to the methods:

(i) For any $A \in \mathscr{R}^{n\times n}$ non-singular, do there exist n left conjugate directions? The finite termination property of the left conjugate direction method requires a positive answer of this question.
(ii) How to overcome the breakdown problem efficiently appeared in the left conjugate direction method?
(iii) Is it possible not to store all left conjugate directions and meanwhile to save the algebraic computation amount per iteration? This question is important for the implementation of the left conjugate direction method because it implies the possibility of decreasing the cost of the method.

The above questions motivated the study of this paper on semi-conjugate direction methods. After a concise review on the methods in Section 2, we will give an answer or a partial answer to each of the questions in Sections 3–5, respectively. For simplicity, we shall discuss only the left conjugate direction method in this paper instead of the semi-conjugate direction method because the left conjugate direction method is one important type of the semi-conjugate direction method. Some discussion will be made in the last section.

The outline of this paper is as follows. We shall briefly review the left conjugate direction method with two algorithms in Section 2. The existence of the left conjugate direction vectors is showed in Section 3. A new technique to overcome breakdown problem is implemented in Section 4. The limited-memory left conjugate direction method is proposed with numerical experiments in Section 5. The discussions and conclusions are given in the last section.

2. RESTRUCTURING THE LEFT CONJUGATE DIRECTION METHOD

In this section, we will recall the definition of the left conjugate direction in Reference [1] and describe the left conjugate direction method in a different way.

Definition 2.1
Vectors $p_1, p_2, \ldots, p_l$ are called left conjugate directions of an $n \times n$ real non-singular matrix A if $p_i^{\mathrm{T}} A p_j = 0$ for $i < j$, and $p_i^{\mathrm{T}} A p_j \neq 0$ for $i = j$.

Assuming that n left conjugate directions $\{p_1, p_2, \ldots, p_n\}$ of A have been generated in some way, the left conjugate direction method can be described as follows:

Algorithm 2.2 (*Left conjugate direction method*)

1. Input x_1, A and b;
2. Calculate $r_1 = b - Ax_1$;
3. For $k = 1, 2, \ldots, n$, do

$$\alpha_k = \frac{p_k^{\mathrm{T}} r_k}{p_k^{\mathrm{T}} A p_k}, \quad x_{k+1} = x_k + \alpha_k p_k, \quad r_{k+1} = r_k - \alpha_k A p_k.$$

As shown in Reference [1], the left conjugate direction method terminates at some point x_k with $k \leqslant n+1$ and $r_k = 0$. However, this finite termination property depends on the existence of n left conjugate directions of A. In the case of skew-symmetric matrices, namely, $A^{\mathrm{T}} = -A$, there is no left conjugate direction because of $p^{\mathrm{T}} A p = 0$ for all p. Nevertheless, the existence of n left conjugate directions can be proved for all non-skew-symmetric and non-singular matrices (see Section 3).

Here, we describe the left conjugate direction method in a different way where only one matrix–vector multiplication is required at each step. This nice property does not be possessed by some other iterative methods for example Bi-CG.

Algorithm 2.3 (*Left conjugate direction method*)

1. Input x_1, A, p_1 and b;
2. Calculate $r_1 = b - Ax_1$ and $q_1 = Ap_1$;
3. For $k = 1, 2, \ldots, n$, do

(3.1) $\alpha_k = p_k^{\mathrm{T}} r_k / p_k^{\mathrm{T}} q_k$, $x_{k+1} = x_k + \alpha_k p_k$, $r_{k+1} = r_k - \alpha_k q_k$;
(3.2) $p_{k+1} = r_k$, $q_{k+1} = Ap_{k+1}$, for $i = 1, \ldots, k$, do

$$\beta_i = \frac{p_i^{\mathrm{T}} q_{k+1}}{p_i^{\mathrm{T}} q_i}, \quad p_{k+1} = p_{k+1} - \beta_i p_i, \quad q_{k+1} = q_{k+1} - \beta_i q_i$$

In practical computations, the left conjugate direction method may breakdown [1]. It is then important to study how to overcome the breakdown problem efficiently. Another disadvantage of the method is to store all the generated left conjugate directions $\{p_1, \ldots, p_k\}$ and their coupled vectors $\{q_1, \ldots, q_k\}$. Meanwhile, at the kth iteration, $5k$ vector(scalar)–vector operations are required to generate p_{k+1} and q_{k+1} besides one matrix–vector multiplication. This may be very crucial to the performance of the method if the matrix A is sparse. The technique of limited-memory will be introduced in Section 5 to save the storage and multiplications.

3. EXISTENCE OF LEFT CONJUGATE DIRECTIONS

In this section, we shall discuss the existence of left conjugate directions of a non-singular matrix A. By the relationship [1] between the left conjugate direction method and the Gaussian elimination, it is easy to know that a non-singular matrix A has n left conjugate directions if its Gaussian elimination exists.

However, the following example shows that the existence of Gaussian elimination is not necessary for the existence of n left conjugate directions. Consider $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. This matrix has no Gaussian elimination since its first leading principal submatrix is zero. But we can directly check that $p_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $p_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ are two left conjugate directions of the above matrix.

If the matrix A is skew-symmetric, namely, $A^{\mathrm{T}} = -A$, then A has no any left conjugate direction since $p^{\mathrm{T}}Ap = 0$ for every $p \in R^n$. It is then interesting to investigate some matrix 'close to' a skew-symmetric matrix.

Example 3.1
Consider the matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The fourth leading principal matrix of the above matrix is skew-symmetric and hence A can be regarded as an augmented matrix of a skew-symmetric matrix. We can check that the following vectors, with their components being either 0 or 1, are 5 left conjugate directions of A:

$$p_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}, \quad p_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad p_3 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad p_4 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad p_5 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

In fact, defining $P = (p_1, \ldots, p_5)$, we have that

$$P^{\mathrm{T}}AP = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 2 & 2 & 1 & 0 & 0 \\ 2 & 2 & 2 & 1 & 0 \\ 2 & 2 & 2 & 2 & 1 \end{pmatrix}$$

is a lower triangular matrix.

The above example might hint us that every non-singular matrix A has n left conjugate directions provided that A is not skew-symmetric. We shall prove this conjecture in the following.

Lemma 3.1
Suppose that $p_1, \ldots, p_l \in R^n$ are l left conjugate directions of a non-singular matrix A. Define $S_l = \{u \in R^n:\ p_i^{\mathrm{T}} A u = 0 \text{ for all } i = 1, \ldots, l\}$, Then for every $u_1 \in S_l \backslash \{0\}$, there must exist some vector $u_2 \in S_l$ such that $u_1^{\mathrm{T}} A u_2 \neq 0$.

Proof
Since $p_1, \ldots, p_l$ are left conjugate directions, they are linearly independent [1]. Thus it follows from the non-singularity of A that the subspace S_l has the dimension of $n-l$. Consequently, the subspace $T_l = \{v : v^{\mathrm{T}} A u = 0 \text{ for all } u \in S_l\}$ has the dimension of l. Since linearly independent $p_i \in T_l$ for $i = 1, \ldots, l$, the set $\{p_1, \ldots, p_l\}$ forms a basis of T_l.

Consider now any vector $u_1 \in S_l \backslash \{0\}$. If $u_1 \in T_l$, we can write $u_1 = \sum_{i=1}^{l} \delta_i p_i$ with at least one non-zero coefficient δ_i. Let i_0 be the least subscript i such that $\delta_i \neq 0$. By the definition of left conjugate directions and the choice of i_0, we can get that

$$p_{i_0}^{\mathrm{T}} A u_1 = \sum_{i=1}^{l} \delta_i p_{i_0}^{\mathrm{T}} A p_i = \delta_{i_0} p_{i_0}^{\mathrm{T}} A p_{i_0} \neq 0$$

This means that $u_1 \notin S_l$! The contradiction shows that $u_1 \notin T_l$, and hence there must exist some vector u_2 in S_l satisfying $u_1^{\mathrm{T}} A u_2 \neq 0$. □

Lemma 3.2
Suppose that $p_1, \ldots, p_l \in R^n$ $(l < n)$ are l left conjugate directions of a non-singular matrix A. Suppose that u_1 and u_2 are vectors in S_l with $u_1^{\mathrm{T}} A u_2 \neq 0$. Then there exist non-zero real numbers α and β such that the vectors $p_1, \ldots, p_{l-1}, \bar{p}_l = p_l + \alpha u_1, \bar{p}_{l+1} = p_l + \beta u_2$ are $l+1$ left conjugate directions of A.

Proof
To be such that $p_1, \ldots, p_{l-1}, \bar{p}_l, \bar{p}_{l+1}$ are left conjugate directions of A, it suffices to choose the parameters α and β such that

$$\bar{p}_l^{\mathrm{T}} A \bar{p}_l \neq 0, \quad \bar{p}_{l+1}^{\mathrm{T}} A \bar{p}_{l+1} \neq 0, \quad \bar{p}_l^{\mathrm{T}} A \bar{p}_{l+1} = 0 \tag{2}$$

Since $u_1, u_2 \in S_l$, we have by direct calculations that

$$\begin{aligned} \bar{p}_l^{\mathrm{T}} A \bar{p}_l &= p_l^{\mathrm{T}} A p_l + \alpha u_1^{\mathrm{T}} A p_l + \alpha^2 u_1^{\mathrm{T}} A u_1 \\ \bar{p}_{l+1}^{\mathrm{T}} A \bar{p}_{l+1} &= p_l^{\mathrm{T}} A p_l + \beta u_2^{\mathrm{T}} A p_l + \beta^2 u_2^{\mathrm{T}} A u_2 \\ \bar{p}_l^{\mathrm{T}} A \bar{p}_{l+1} &= p_l^{\mathrm{T}} A p_l + \alpha u_1^{\mathrm{T}} A p_l + \alpha\beta u_1^{\mathrm{T}} A u_2 \end{aligned} \tag{3}$$

Denote $\mathscr{A}$ to be the set of all α such that $\bar{p}_l^{\mathrm{T}} A \bar{p}_l = 0$, and $\mathscr{B}$ the set of all β such that $\bar{p}_{l+1}^{\mathrm{T}} A \bar{p}_{l+1} = 0$. Note that $\mathscr{A}$ and $\mathscr{B}$ have at most two elements. Then, there must exist some $\alpha \notin \mathscr{A} \cup \{0\}$ such that

$$\beta = -\frac{p_l^{\mathrm{T}} A p_l + \alpha u_1^{\mathrm{T}} A p_l}{\alpha u_1^{\mathrm{T}} A u_2} \notin \mathscr{B} \cup \{0\} \tag{4}$$

In this case, since $\alpha \notin \mathscr{A}$ and $\beta \notin \mathscr{B}$, the two inequalities in (2) hold. It follows from (3) and the equality in (4) that $\bar{p}_l^{\mathrm{T}} A \bar{p}_{l+1} = 0$. Thus such α and β satisfy (2). This completes our proof. □

Theorem 3.3
For every non-singular matrix $A \in R^{n\times n}$, which is not skew-symmetric, namely, $A^{\mathrm{T}} \neq -A$, there exist n left conjugate directions.

Proof
Since A is not skew-symmetric, there must exist a vector p_1 such that $p_1^{\mathrm{T}} A p_1 \neq 0$. This p_1 can be regarded as one left conjugate direction of A. Generally, assume that $l(< n)$ left conjugate directions $p_1, \ldots, p_l$ have been found. Then by Lemma 3.1, there exist $u_1, u_2 \in S_l$ satisfying $u_1^{\mathrm{T}} A u_2 \neq 0$, and by Lemma 3.2, there exist $l+1$ left conjugate directions of A. Therefore, by induction, A has n left conjugate directions. □

Thus, we have proved that every n-dimensional non-singular matrix A has n left conjugate directions unless A is skew-symmetric. A direct corollary of Theorem 3.3 is as follows.

Corollary 3.4
For every non-singular matrix $A \in R^{n\times n}$, if $A^{\mathrm{T}} \neq -A$, then there exists a non-singular matrix $P \in R^{n\times n}$ such that $P^{\mathrm{T}} A P$ is lower triangular (or upper triangular).

Suppose that $A \in R^{n\times n}$ is non-singular but not skew-symmetric, and that $q_1, \ldots, q_n$ are n linearly independent vectors in R^n with $q_1^{\mathrm{T}} A q_1 \neq 0$. In the rest of this section, we will describe a strategy to obtain n left conjugate directions of A from $\{q_1, \ldots, q_n\}$. The strategy is based on the proofs to Lemma 3.2 and Theorem 3.3.

For convenience, we introduce a notation. Assume that $p_1, \ldots, p_k$ are k left conjugate directions of A. For any $v \in R^n$, we denote $\bar{v} = \mathrm{LC}(A, p_1, \ldots, p_k, v)$ to be the vector $\bar{v} = v + \sum_{i=1}^{k} \beta_i p_i$ such that $p_i^{\mathrm{T}} A v = 0$ for $i = 1, \ldots, k$. The vector $\bar{v}$ can be obtained in a recursive way (one can refer to step 3.2 of Algorithm 2.3).

Algorithm 3.5 (*A strategy to generate left conjugate directions*)

1. $p_1 = q_1,\ l = 1,\ j = 2$;
2. $u_2 = \mathrm{LC}(A, p_1, p_2, \ldots, p_l, q_{l+1})$;

   (2.1) If $u_2^{\mathrm{T}} A u_2 \neq 0$, $p_{l+1} = u_2$, go to step 3;
   (2.2) $j = j + 1$, $u_1 = \mathrm{LC}(A, p_1, p_2, \ldots, p_l, q_j)$. If $u_1^{\mathrm{T}} A u_2 \neq 0$, $p_{l+1} = p_l + \beta u_2$, $p_l = p_l + \alpha u_1$, $q_j = u_1$ (with non-zero α and β satisfying (2), go to step 3; otherwise, repeat step 2.2.

3. $l = l + 1$, $j = l + 1$. If $l = n$, stop; otherwise, go to step 2.

In step 2.2 of the above algorithm, by Lemma 3.2, if $u_1^{\mathrm{T}} A u_2 \neq 0$, α and β can be found such that $p_1, \ldots, p_{k+1}$ are left conjugate directions of A. However, it is necessary to prove that there exists some index j such that $u_1^{\mathrm{T}} A u_2 \neq 0$, where $u_1 = \mathrm{LC}(A, p_1, p_2, \ldots, p_l, q_j)$. For this purpose, we provide the following theorem.

Theorem 3.6
Suppose that A is non-singular but not skew-symmetric, and that $q_1, \ldots, q_n$ are linearly independent vectors in R^n with $q_1^{\mathrm{T}} A q_1 \neq 0$. Then Algorithm 3.5 is well defined, and the generated vectors, $p_1, \ldots, p_n$, are left conjugate directions of A.

Proof

We prove the following statements hold for $l = 1, \ldots, n$ by induction: (i) $p_1, \ldots, p_l$ are left conjugate directions of A; (ii) $p_1, \ldots, p_l, q_{l+1}, \ldots, q_n$ are linearly independent.

Since $p_1 = q_1$ and $q_1^{\mathrm{T}} A q_1 \neq 0$, (i) and (ii) clearly hold for $l = 1$. Assume that (i) and (ii) hold for some $l < n$. Firstly, we consider the case that p_{l+1} is obtained by step 2.1, namely, $p_{l+1} = u_2$. In this case, by the definition of u_2 and $u_2^{\mathrm{T}} A u_2 \neq 0$, we have that $p_1, \ldots, p_{l+1}$ are $l+1$ left conjugate directions of A. Hence,

$$\det(p_1, \ldots, p_l, p_{l+1}, q_{l+2}, \ldots, q_n) = \det(p_1, \ldots, p_l, q_{l+1}, q_{l+2}, \ldots, q_n) \neq 0$$

which shows that $p_1, \ldots, p_{l+1}, q_{l+2}, \ldots, q_n$ are linearly independent. Thus the statements (i) and (ii) hold for $l+1$ in this case. Secondly, we consider the case that p_{l+1} is obtained by step 2.2, namely, $u_2^{\mathrm{T}} A u_2 = 0$. Denote $\bar{q}_j = \mathrm{LC}(A, p_1, \ldots, p_l, q_j)$ for $j = l+2, \ldots, n$. It is easy to verify

$$\det(p_1, \ldots, p_l, u_2, \bar{q}_{l+2}, \ldots, \bar{q}_n) = \det(p_1, \ldots, p_l, q_{l+1}, q_{l+2}, \ldots, q_n) \neq 0 \tag{5}$$

which implies the linear independency of the vectors $p_1, \ldots, p_l, u_2, \bar{q}_{l+2}, \ldots, \bar{q}_n$. It follows by definition of u_2 that

$$p_j^{\mathrm{T}} A u_2 = 0 \quad \text{for } j = 1, \ldots, l \tag{6}$$

If

$$\bar{q}_j^{\mathrm{T}} A u_2 = 0 \quad \text{for } j = l+2, \ldots, n \tag{7}$$

we can deduce by (6), (7), and the linear independence of $p_1, \ldots, p_l, u_2, \bar{q}_{l+1}, \ldots, \bar{q}_n$ that $u_2 = 0$, contradicting (5). Thus $\bar{q}_j^{\mathrm{T}} A u_2 \neq 0$ must hold for some $j \in \{l+2, \ldots, n\}$ and step 2.2 is well defined. Suppose that $u_1 = \bar{q}_{j_0}$ for some j_0. Since $\beta \neq 0$, we can get that

$$\begin{aligned}
&\det(p_1, \ldots, p_{l-1}, p_l + \alpha u_1, p_l + \beta u_2, q_{l+2}, \ldots, q_{j_0-1}, u_1, q_{j_0+1}, \ldots, q_n) \\
&\quad = \det(p_1, \ldots, p_{l-1}, p_l, p_l + \beta u_2, q_{l+2}, \ldots, q_{j_0-1}, u_1, q_{j_0+1}, \ldots, q_n) \\
&\quad = \beta \det(p_1, \ldots, p_l, q_{l+1}, \ldots, q_n) \neq 0
\end{aligned}$$

Therefore, the new $p_1, \ldots, p_l, p_{l+1}, q_{l+2}, \ldots, q_n$ obtained by step 2.2 are linearly independent. By Lemma 3.2 and the choices of α and β, $p_1, \ldots, p_{l+1}$ are left conjugate directions of A. Thus the statements (i) and (ii) also hold in this case. By induction, (i) and (ii) are true for $l = 1, \ldots, n$. Therefore, Algorithm 3.5 is well defined, and generates n left conjugate directions of A. □

A left conjugate direction method can then be designed based on Algorithms 2.2 and 3.5. We may also use Algorithm 3.5 to generate a matrix P such that $P^{\mathrm{T}} A P$ is lower triangular. In addition, the proof to Theorem 3.6 provides us with another approach to show that there must exist vectors u_1 and u_2 in S_l (see Lemma 3.1 for the definition of S_l) such that $u_1^{\mathrm{T}} A u_2 \neq 0$.

For every i, let e_i be the vector whose ith element is 1 and the others are zero. In practical computations, if $e_1^{\mathrm{T}} A e_1 \neq 0$, we may take $q_i = e_i$. In the case that all leading principal minors of A are non-zero, it is easy to see that all left conjugate directions will be generated by step 2.1, and the matrix $P = (p_1, \ldots, p_n)$ is unit upper triangular. This means that

$$P^{\mathrm{T}} A P = T, \quad \text{where } T \text{ is lower triangular.} \tag{8}$$

Thus by letting $L_1 = P^{-\mathrm{T}} T$ and $U_1 = P^{-1}$, we can obtain one kind of LU decomposition of the matrix A. This has been demonstrated by our numerical experiments. Assume that

$$A = \begin{pmatrix} 1 & 0.1632 & 0.0232 \\ 0.9288 & 0.2763 & 0.2313 \\ 0.4851 & 0.1310 & 0.8453 \end{pmatrix}$$

Direct tests show that the leading principle minors are non-zero. Using Algorithm 5.1 and prescribing $q_i = e_i$ for $i = 1, 2, 3$, we obtained

$$P = \begin{pmatrix} 1 & -0.1632 & 0.2513 \\ 0 & 1 & -1.6818 \\ 0 & 0 & 1 \end{pmatrix}, \quad T = P^{\mathrm{T}} A P = \begin{pmatrix} 1 & 0 & 0 \\ 0.7656 & 0.1247 & 0 \\ -0.8257 & -0.1579 & 0.7469 \end{pmatrix}$$

With Matlab we also obtained the LU decomposition of A as follows:

$$L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0.9288 & 1 & 0 \\ 0.4851 & 0.4156 & 1 \end{pmatrix}, \quad U_2 = \begin{pmatrix} 1 & 0.1632 & 0.0232 \\ 0 & 0.1247 & 0.2098 \\ 0 & 0 & 0.7469 \end{pmatrix}$$

Denoting $D = \mathrm{diag}(U_{11}, U_{22}, U_{33})$, we found that $L_1 = P^{-\mathrm{T}} T = L_2 D$ and $U_1 = P^{-1} = D^{-1} U_2$ in numerical computations.

The following matrix is also used to test Algorithm 3.5:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 \end{pmatrix}$$

Taking $q_i = e_i$ for $i = 1, \ldots, 5$ and $\alpha = \beta = 1$, we obtained $P = (p_1, \ldots, p_l)$ and $T = P^{\mathrm{T}}AP$ as follows:

$$P = \begin{pmatrix} 1 & 1 & -1 & -1 & 1 \\ 0 & 1 & -1 & -1 & 1 \\ 1 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}, \quad T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ -2 & -2 & 1 & 0 & 0 \\ -2 & -2 & 2 & 1 & 0 \\ 2 & 2 & -2 & -2 & 1 \end{pmatrix}$$

Thus the above two examples demonstrated the usefulness of Algorithm 3.5.

4. A TECHNIQUE OF OVERCOMING BREAKDOWN

If A is skew-symmetric when no left conjugate directions exist, the direct application of the left conjugate direction method is not possible. A remedy [1] is to multiply A by some permutation matrix so that the permuted matrix, $\bar{A}$ say, is not skew-symmetric. Then we know by Theorem 3.3 that there exist n left conjugate directions of $\bar{A}$ and the left conjugate direction method is applicable. Based on Example 3.1 and Theorem 3.3, we propose a new remedy technique of augmented matrix which is described in following theorem.

Theorem 4.1
Suppose that A and B are non-singular matrices in $R^{n\times n}$ and $R^{m\times m}$, respectively. Suppose that $p_1, \ldots, p_l$ are l left conjugate directions of the matrix A. Then

$$q_1 = \begin{pmatrix} p_1 \\ 0_m \end{pmatrix}, \ldots, q_l = \begin{pmatrix} p_l \\ 0_m \end{pmatrix}$$

are left conjugate directions of the augmented matrix $\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}$. Further, assume also that $p_{l+1} \in R^n$ satisfies

$$p_i^{\mathrm{T}} A p_{l+1} = 0, \quad i = 1, \ldots, l, \quad \text{but } p_{l+1}^{\mathrm{T}} A p_{l+1} = 0 \tag{9}$$

and that $v \in R^m$ satisfies $v^{\mathrm{T}} B v \neq 0$. Then $q_1, \ldots, q_l, q_{l+1} = \begin{pmatrix} p_{l+1} \\ v \end{pmatrix}$ are $l+1$ left conjugate directions of the augmented matrix.

Proof
The statements follow directly by the definition of left conjugate direction. □

For the choice of v, we may choose

$$B = tI_1, \quad \text{where } t \neq 0 \tag{10}$$

In this case, every non-zero real number can be chosen as v. We then summarize the breakdown free algorithm of the left conjugate direction method as follows.

Algorithm 4.2
Suppose that it happens that $p_k^{\mathrm{T}} q_k = 0$ in step 3.1 of Algorithm 2.3. Then we do:

1. $A = \begin{pmatrix} A & 0 \\ 0 & t \end{pmatrix}$, $b = \begin{pmatrix} b \\ 0 \end{pmatrix}$, where $t \neq 0$ is any real number;
2. $p_i = \begin{pmatrix} p_i \\ 1 \end{pmatrix}$, $q_i = \begin{pmatrix} q_i \\ t \end{pmatrix}$, $i = 1, \ldots, k$;
3. $x_k = \begin{pmatrix} x_k \\ 0 \end{pmatrix}$, $r_k = \begin{pmatrix} r_k \\ 0 \end{pmatrix}$;
4. go to step (3.1).

In theory, the value t in the above algorithm is only required to be non-zero.

Example 4.1
Consider the non-singular system in Reference [1],

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 3 & 2 \end{pmatrix} x = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}$$

where the exact solution is $x^* = (1, 3, -5)^{\mathrm{T}}$. If we choose $p_1 = (1, 0, 0)^{\mathrm{T}}$, the left conjugate direction method will break down. To overcome the problem, a technique by perturbing A to $A + \varepsilon I$ is used in Reference [1]. But they showed that it is not possible to get a good approximation whose relative error is better than the order of 10^{-6}. Another difficulty with this perturbation technique is how to choose the parameter ε. However, if we apply Algorithm 4.2 with $t = 1$ to overcome the breakdown, the left conjugate direction method will obtain a solution in four iterations. The relative error of the solution is

$$\frac{\|x^* - x\|_2}{\|x^*\|_2} = 3.7532 \times 10^{-17}$$

Example 4.2
Consider the skew-symmetric system $Ax = b$ in Reference [1], where

$$A = \begin{pmatrix} 0 & 474 & 316 & 158 \\ -474 & 0 & 474 & 316 \\ -316 & -474 & 0 & 474 \\ -158 & -316 & -474 & 0 \end{pmatrix}$$

$$b = \begin{pmatrix} -790 \\ -632 \\ -1738 \\ -948 \end{pmatrix}, \quad x_{\mathrm{exact}} = \begin{pmatrix} 1 \\ -2 \\ 3 \\ -5 \end{pmatrix}$$

By the perturbation technique, an approximate solution is obtained in Reference [1] with its relative error being in the order of 10^{-5}. By the permutation technique, an approximation with

smaller relative error (at least of order $O(10^{-11})$) can be obtained [1]. If we use Algorithm 4.1 with $t = 1$, $x_0 = 0$, $p = b/\|b\|_2$, then we can obtain a solution in five iterations and its relative error is 1.3486×10^{-11}. Similar results were also obtained for $p = \text{eye}(4, 1)$, $p = \text{rand}(n, 1)$, and $p = A(4, 1)/\|A(4, 1)\|_2$.

The above examples demonstrated the efficiency of the technique of augmented matrix. In theory, if there are only finite numbers, m say, of indices k such that $p_k^{\mathrm{T}} q_k = 0$, then the left conjugate direction method with Algorithm 4.2 will give the exact solution in at most $n + m$ iterations. This is because, assuming that the final augmented matrix is $\tilde{A}$, the left conjugate direction method can be regarded to apply for a linear system with its coefficient matrix being $\tilde{A}$. However, we still do not know yet whether the case that $p_k^{\mathrm{T}} q_k = 0$ can occur for infinite times or not.

5. LIMITED-MEMORY LEFT CONJUGATE DIRECTION METHOD

Unlike the conjugate gradient method for symmetric and positive definite linear systems, the left conjugate direction method does not have short-recurrence formula generally [5]. The latter needs to store all the generated left conjugate directions $\{p_1, \ldots, p_k\}$ and their coupled vectors $\{q_1, \ldots, q_k\}$. As a result, the storage required by the left conjugate direction method is increasing with the iteration number k. At the same time, we can see from step 3.2 in Algorithm 2.3 that the cost to calculate the next left conjugate directions becomes higher and higher with k.

A natural idea to avoid the two disadvantages of the left conjugate direction method is to discard those *old* left conjugate directions and only preserve some *recent* left conjugate directions. Given any positive integer m, a variant of the left conjugate direction method is described as Algorithm 5.1, in which only recent $\max(k, m)$ left conjugate directions are preserved. This stage is similar to the limited-memory BFGS method (for example, see Reference [6]), in which only pairs $\{(s_{k-1}, y_{k-1}), \ldots, (s_{k-\max(k,m)}, y_{k-\max(k,m)})\}$ are used for generating the new approximation matrix B_{k+1}. Thus we call Algorithm 5.1 as limited-memory left conjugate direction method.

Algorithm 5.1 (*Limited-memory left conjugate direction method*)

1. Input x_1, A, b, m, ε;
2. Calculate $r_1 = b - Ax_1$, $p_1 = r_1$, $q_1 = Ap_1$, $k = 1$, $l = 1$;
3. While $\|r_k\| > \varepsilon$, do
   - (3.1) Calculate $t_k = p_k^{\mathrm{T}} q_k$. If $t_k = 0$, apply Algorithm 4.2;
   - (3.2) $\alpha_k = p_k^{\mathrm{T}} r_k / t_k$, $x_{l+1} = x_l + \alpha_k p_k$, $r_{l+1} = r_l - \alpha_k q_k$;
   - (3.3) If $l \geqslant m$, then $k = \text{mod}(k, m)$, $j_1 = k + 1$, $j_2 = k + m$;
     else $j_1 = 1$, $j_2 = k$;
   - (3.4) $p = r_{l+1}$, $q = Ap$;
     For $j = j_1, \ldots, j_2$, do
     $i = \text{mod}(j, m)$, $\beta_i = p_i^{\mathrm{T}} q / t_i$, $p = p - \beta_i p_i$, $q = q - \beta_i q_i$;
     $p_{k+1} = p$, $q_{k+1} = q$;
   - (3.5) $k = k + 1$, $l = l + 1$.

We have tested the limited-memory left conjugate direction method by two examples. All tests in this section were done with MATLAB in the machine SGI2100 in the Academy of Mathematics and System Sciences, Chinese Academy of Sciences. The first example is Example 7.1 in Reference [1] (see also Reference [7]).

Example 5.1
Consider the test problem derived by discretizing the elliptic partial differential equation

$$-\Delta u + 2\delta_1 u_x + 2\delta_2 u_y - \delta_3 u = f \tag{11}$$

with constant coefficients δ_1, δ_2 and δ_3 on the unit square $\Omega = \{(x, y) : 0 \leqslant x, y \leqslant 1\}$, and with boundary condition $u(x, y) = 0$ on $\partial\Omega$. The function f is chosen so that $u(x, y) = xe^{xy}\sin(\pi x)\cos(\pi y)$ solves (11). We use symmetric finite differences on a uniform $(n+2) \times (n+2)$ grid, including boundary points, and the standard five-point stencil to approximate Δu to yield a linear system $N = n^2$ equations for n^2 unknowns $u_{ij} = u(ih, jh)(1 \leqslant i, j \leqslant n)$:

$$\begin{aligned}(4 - \delta_3 h^2)u_{ij} - (1 + \delta_1 h)u_{i-1,j} - (1 - \delta_1 h)u_{i+1,j} - (1 + \delta_2 h)u_{i,j-1} \\ -(1 - \delta_2 h)u_{i,j+1} = h^2 f_{ij}\end{aligned} \tag{12}$$

where $f_{ij} = f(ih, jh)$ and $h = 1/(n+1)$. Same as References [1, 7], the following three cases are used for our numerical tests:

Case I: $(\delta_1, \delta_2, \delta_3) = (30, 40, 40)$;
Case II: $(\delta_1, \delta_2, \delta_3) = (60, 80, 40)$;
Case III: $(\delta_1, \delta_2, \delta_3) = (80, 80, 40)$.

The following stopping rule is used in this example:

$$\|r_k\|_2 / \|r_1\|_2 \leqslant 10^{-6} \tag{13}$$

We tested the limited-memory left conjugate direction method with each value of $m \in [1, 20]$. The left conjugate direction method has also been tested, which is corresponding to the case that $m = \infty$. We checked that the relation (13) is achieved by each test. We listed the iteration numbers and CPU times in Tables I and II. The tables are corresponding to the cases that $n = 30$ and that $n = 40$, respectively.

The second example is drawn from Reference [8].

Example 5.2
Consider the three-dimensional convection–diffusion equation

$$-(u_{xx} + u_{yy} + u_{zz}) + q(u_x + u_y + u_z) = 0 \tag{14}$$

on the unit cube $\Omega = [0, 1] \times [0, 1] \times [0, 1]$, with constant coefficient q and subject to Dirichlet-type boundary conditions. Suppose that the seven-point finite centered difference is used in discretizing both the diffusive terms and the convective terms. We then get the systems of linear equations $Ax = b$ with the coefficient matrix

$$A = T_x \otimes I \otimes I + I \otimes T_y \otimes I + I \otimes I \otimes T_z \tag{15}$$

Table I. Results for Example 5.1 with $N = 30^2$.

	Case I		Case II		Case III	
m	Iter	Time	Iter	Time	Iter	Time
1	82	2.80	106	3.60	108	3.65
2	80	2.73	204	7.05	315	10.89
3	85	2.95	93	3.22	102	3.53
4	91	3.19	97	3.36	103	3.57
5	92	3.28	101	3.57	106	3.72
6	97	3.46	97	3.46	100	3.56
7	101	3.64	100	3.64	107	3.85
8	94	3.43	98	3.58	107	3.87
9	96	3.54	109	3.98	107	3.91
10	96	3.56	107	3.99	115	4.28
11	99	3.73	106	3.97	109	4.07
12	101	3.81	109	4.19	109	4.18
13	109	4.16	110	4.20	113	4.31
14	113	4.33	113	4.35	125	4.84
15	117	4.54	116	4.59	123	4.85
16	114	4.51	123	4.91	130	5.19
17	125	4.99	131	5.24	121	4.83
18	135	5.41	125	5.02	134	5.39
19	136	5.47	138	5.62	128	5.22
20	149	6.11	139	5.69	131	5.29
∞	62	2.97	68	3.39	68	3.35

where the equidistant step-size $h = 1/(n+1)$ is used in the discretization on all the three directions and the natural lexicographical ordering is employed to the unknowns. The symbol $\otimes$ denotes the Kronecker product. Letting

$$r = \frac{qh}{2} \tag{16}$$

and

$$t_1 = 6, \quad t_2 = -1 - r, \quad t_3 = -1 + r \tag{17}$$

the tridiagonal matrices T_x, T_y and T_z in (15) are given by

$$T_x = \text{tridiag}(t_2, t_1, t_3), \quad T_y = T_z = \text{tridiag}(t_2, 0, t_3) \tag{18}$$

The order of the coefficient matrix A in (14) is $N = n^3$. Two values of n were used in our tests: $n = 10$ and $n = 15$. The value r in (16) is the mesh Reynolds number. After n is fixed, the value of q decides the method Reynolds number. Same as Reference [8], the following four values of q were tested: $q = 1$, $q = 10$, $q = 100$ and $q = 10\,000$. In addition, the right-hand term b is $b = Au^*$ with $u^* = (1, 1, \ldots, 1)^{\mathrm{T}}$ and the initial points u_0 is zero vector.

As before, we tested the limited-memory left conjugate direction method with $m \in [1, 20]$ and the left conjugate direction method (namely $m = \infty$). The stopping rule is still (13). We checked that (13) was achieved by each test with Example 5.2. Table III listed the numerical

Table II. Results for Example 5.1 with $N = 40^2$.

	Case I		Case II		Case III	
m	Iter	Time	Iter	Time	Iter	Time
1	99	11.68	115	13.43	116	13.80
2	105	12.50	111	12.98	146	17.18
3	112	13.48	106	12.44	112	13.44
4	116	14.03	111	13.11	122	14.72
5	114	13.89	116	13.63	121	14.44
6	113	13.58	116	13.92	129	15.69
7	122	14.86	115	13.64	130	15.87
8	127	15.62	122	14.66	128	15.45
9	129	15.86	125	14.96	131	16.11
10	119	14.80	122	14.64	142	17.40
11	116	14.36	129	15.53	137	17.04
12	118	14.73	124	15.13	138	17.11
13	126	15.60	129	15.66	133	16.47
14	126	15.63	129	15.88	133	16.33
15	124	15.61	127	15.68	136	17.01
16	132	16.70	135	16.71	135	17.22
17	140	17.83	142	17.63	141	17.80
18	143	18.07	145	18.20	158	20.25
19	140	17.76	149	18.52	152	19.54
20	143	18.31	151	19.10	156	19.99
∞	80	11.61	83	11.97	83	12.06

results with $n = 10$ and different values of q, whereas Table IV listed the numerical results with $n = 15$.

From the tables, we can draw the following statements:

(1) The left conjugate direction method (namely, the case $m = \infty$) always gives the least iteration number for each problem. This is understandable since all the generated left conjugate directions are preserved and used. However, the method may not provide the least computing time since it requires more CPU time in the generation of left conjugate direction when $k > 2m$. For instance, Example 5.1 with $n = 30$ and Case II, the limited-memory left conjugate direction method with $m = 3$ provides the least computing time. Another instance is Example 5.2 with $n = 10$ and $q = 1000$.

(2) As m increases, the iteration number required by the limited-memory left conjugate direction method needs not be monotonically decreasing. The iteration number is indeed decreasing for Example 5.3 with $n = 10$ and $q = 1$. However, this is not true in all the other cases.

(3) The limited-memory left conjugate direction method is very robust for both the examples. Although its convergence is not proved yet even for positive definite but non-symmetric linear systems, the tables illustrate that the method with different values of m can provide the relation (13).

(4) It is difficult to decide the best value of m. For instance, for Example 5.1 with $n = 30$, $m = 2$ gives the least computing time in Case I, but also provides the worst numerical

Table III. Results for Example 5.2 with $N = 10^3$.

	$q = 1$		$q = 10$		$q = 100$		$q = 1000$	
m	Iter	Time	Iter	Time	Iter	Time	Iter	Time
1	53	2.38	44	1.98	85	3.94	518	24.00
2	52	2.33	48	2.17	104	4.85	515	23.78
3	50	2.28	43	1.97	98	4.60	592	27.67
4	46	2.11	47	2.16	101	4.81	515	24.33
5	45	2.07	50	2.34	100	4.78	465	21.97
6	44	2.04	48	2.26	87	4.20	1255	59.94
7	40	1.86	51	2.39	98	4.78	460	22.50
8	39	1.85	50	2.35	93	4.60	454	22.14
9	38	1.82	50	2.40	93	4.60	474	23.48
10	38	1.83	52	2.50	93	4.60	712	35.11
11	38	1.85	51	2.47	98	4.89	463	23.02
12	38	1.84	51	2.49	103	5.22	469	23.81
13	38	1.86	52	2.57	103	5.22	451	23.24
14	38	1.88	47	2.31	108	5.52	497	25.82
15	38	1.89	47	2.35	100	5.17	455	23.53
16	34	1.68	48	2.39	100	5.23	438	22.81
17	34	1.68	49	2.48	89	4.67	433	22.74
18	34	1.69	50	2.51	93	4.88	434	22.78
19	34	1.71	50	2.53	84	4.42	443	23.18
20	34	1.72	50	2.54	85	4.45	426	22.89
∞	34	1.79	34	1.78	60	3.64	244	26.67

results in Case II. There are also quite a few cases in which the limited-memory left conjugate direction method with other values of m performs very poorly. For instance, $m = 3$ and 6 for Example 5.2 with $n = 10$ and $q = 1000$. From the tables, however, it is preferable to choose $m \geqslant 4$.

6. CONCLUSIONS AND DISCUSSION

In this paper, we have carefully studied the left conjugate direction method proposed in Reference [1]. Firstly, we have proved that for any n-dimensional non-singular matrix A, there exist n left conjugate directions provided that A is not skew symmetric. A strategy to generate left conjugate directions of A has also been provided. Secondly, we have provided a simple technique, named the technique of augmented matrix, to overcome the possible breakdown occurred in the left conjugate direction method. Thirdly, to avoid the storage of all left conjugate directions and to economize the cost of the generation of left conjugate directions, the technique of limited-memory has been combined with the left conjugate direction method. Preliminary numerical results have been done, which demonstrated the robustness and efficiency of the limited-memory left conjugate direction method.

We also feel that there are some problems still required to be solved. For example, the problems how to choose the parameter t in (10) and whether the left conjugate direction method with the remedy algorithm—Algorithm 4.2 has the finite termination property. For the limited-memory conjugate direction Algorithm 5.1, we do not know yet whether this method converges

Table IV. Results for Example 5.2 with $N = 15^3$.

	$q = 1$		$q = 10$		$q = 100$		$q = 1000$	
m	Iter	Time	Iter	Time	Iter	Time	Iter	Time
1	78	40.26	68	34.00	94	46.99	378	188.93
2	76	39.75	63	31.59	109	54.70	453	229.14
3	74	37.96	70	35.10	94	47.72	390	194.00
4	73	38.26	60	30.17	87	43.72	417	207.40
5	67	35.12	62	30.91	93	46.90	394	200.63
6	66	34.28	64	32.22	91	46.38	526	262.77
7	66	33.94	71	35.83	97	49.43	384	194.81
8	65	34.34	69	35.14	97	49.20	384	194.87
9	62	32.52	67	33.83	94	47.99	378	192.37
10	59	31.17	67	33.70	99	50.28	390	196.27
11	57	30.26	71	36.00	100	51.15	368	188.56
12	57	29.81	70	35.67	101	51.36	383	195.71
13	56	29.36	72	36.66	99	50.86	376	191.85
14	56	29.89	73	37.39	98	50.16	368	189.77
15	55	29.24	67	34.43	105	53.68	382	193.36
16	55	29.28	71	36.15	103	52.85	389	199.19
17	55	28.85	74	38.06	101	51.92	374	192.65
18	55	29.16	75	38.43	102	52.42	482	246.15
19	55	29.03	66	34.15	100	51.47	364	185.89
20	55	28.97	67	34.54	107	55.03	428	221.55
∞	49	26.80	50	26.46	62	33.39	302	215.80

globally even for symmetric positive definite linear systems. At the same time, we think that this technique of limited-memory can be improved in many ways. For example, one possible way is to preserve at kth iteration a collection of left conjugate directions $\{p_1^{(k)}, \ldots, p_m^{(k)}\}$ such that the values of $(p_i^{(k)})^{\mathrm{T}} A p_i^{(k)} / \| p_i^{(k)} \|_2^2$ $(i = 1, \ldots, m)$ are relatively large. We feel that the left conjugate direction method for non-symmetric systems is far from maturity and more numerical experiments are still required.

ACKNOWLEDGEMENTS

This research was partially supported by the Chinese NSF grants 19801033 and 10171104, the CNPq, Fundação Araucária, in Brazil, and the Sino-Brazil joint research project 'Numerical Linear Algebra and Optimization'.

This work was initiated and mainly done while the first author was visiting Department of Mathematics of Federal University of Paraná, Brazil 2001. He would like to thank the CNPq in Brazil very much for supporting his visit partly. He also thanks Dr Jiahong Yin and Dr Liangzhong Hu for their many helps during his visit. Many thanks are also due to the referees, whose suggestions improve this paper greatly.

REFERENCES

1. Yuan JY, Golub GH, Plemmons RJ, Cecilio WAG. Semi-conjugate direction methods for real positive definite systems. *BIT* 2004; **44**:189–207.

2. Hestenes MR, Stiefel E. The method of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards* 1952; **49**:409–436.
3. Stewart GW. Conjugate direction methods for solving systems of linear equations. *Numerische Mathematik* 1973; **21**:285–297.
4. Renato S. Silva, Fernanda MP, Raupp Regina C. Almeida. A numerical methodolgy to solve thermal pollution problems. *IX Congresso Brasileiro de Engenharia e Cincias Trmicas*, Caxambu, Brazil, 2002 (http://www.lncc.br/proj-pesq/relpesq-02.html/32/2002).
5. Faber V, Manteuffel T. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM Journal on Numerical Analysis* 1984; **21**:352–362.
6. Liu DC, Nocedal J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming* 1989; **45**:503–528.
7. Calvetti D, Golub GH, Reichel L. Adaptive Chebyshev iterative methods for non-symmetric linear systems based on modified moments. *Numerische Mathematik* 1994; **67**:21–40.
8. Bai ZZ, Golub GH, NG MK. Hermitian and skew-Hermitian splitting methods for non-Hermitian positive linear systems. *SIAM Journal on Matrix Analysis and Applications* 2003; **24**:603–626.

