Numerical Analysis Project
Manuscript NA-90-07

June 1990

Adaptive Lanczos Methods for Recursive Condition Estimation

by

William R. Ferng
Gene H. Golub
Robert J. Plemmons

Numerical Analysis Project
Computer Science Department
Stanford University
Stanford, California 94305

Adaptive Lanczos Methods for Recursive Condition Estimation

William R. Ferng* Gene H. Golub[†]
Robert J. Plemmons [‡]

May 31, 1990

Abstract

Estimates for the condition number of a matrix are useful in many areas of scientific computing, including: recursive least squares computations, optimization, eigenanalysis, and general nonlinear problems solved by linearization techniques where matrix modification techniques are used. The purpose of this paper is to propose an adaptive Lanczos estimator scheme, which we call ale, for tracking the condition number of the modified matrix over time. Applications to recursive least squares (RLS) computations using the covariance method with sliding data windows are considered. *ale* is fast for relatively small n - parameter problems arising in RLS methods in control and signal processing, and is adaptive over time, i.e., estimates at time t are used to produce estimates at time $t+1$. Comparisons are made with other adaptive and non-adaptive condition estimators for recursive least squares problems. Numerical experiments are reported indicating that ale yields a very accurate recursive condition estimator.

Subject Classifications: AMS(MOS): 15A18, 65F10, 65F20, 65F35
Key Words: adaptive methods, condition estimation, control, downdating, eigenvalues, Lanczos methods, matrix modifications, recursive least squares, signal processing, singular values, updating
Running Title: Adaptive Lanczos Estimator

***Department of Mathematics, NC State University, Raleigh, NC 276958205. Research supported by the US Air Force under grant no. AFOSR-88-0285.**

[†]Department of Computer Science, Stanford University, Stanford, CA 94305. Research supported by the US Army under grant no. DAAL03-90-G-105.

[‡] Department of Mathematics and Computer Science, Wake Forest University, P.O. Box 7311, Winston-Salem, N.C. 27109. Research supported by the US Air Force under grant no. AFOSR-88-0285.

1 Introduction

Repeated estimates for the condition number of a matrix are useful assessing the accuracy and stability of algorithms employed in many application areas of scientific computing, including: optimization, least squares computations, eigenanalysis, and general nonlinear problems solved by linearization techniques [3], [20], [21], [35]. Our purpose in this paper is to propose an adaptive Lanczos estimator scheme, which we call ale, for tracking the condition number of the modified matrix over time. The key computations involve the adaptive estimation of extreme singular values and vectors for a triangular factor L. We develop adaptive Lanczos schemes for estimating extreme singular values and vectors of L for each recursive modification. Approximations to the secular equations for the modified matrix are used to obtain good starting vectors for the Lanczos schemes. Applications to recursive least squares (RLS) computations using the covariance method with sliding data windows in control and signal processing are considered. The computations are adaptive over time in the sense that estimates at time t are used to obtain estimates at time $t + 1$.

An alternative adaptive condition estimation scheme, called ace and based on optimization principles, has been suggested by Pierce and Plemmons [31], and applied to computations in adaptive control and signal processing in [32], [33]. A non-adaptive incremental condition estimation scheme, called ice, has been suggested by Bischof [5], and applied to computations in signal processing in [6] in a different context. It might be help - ful to clarify the use of the words 'incremental' versus 'adaptive'. 'Incremental' ice obtains condition estimates of a triangular factor that grows, whereas 'adaptive' ale and ace maintain condition estimates when information is added/extracted from an already existing factorization. Comparisons of ale with ace and ice on recursive least squares applications are included in this paper. We begin by reviewing least squares computations.

1.1 Least Squares

The *linear least squares problem* can be posed as follows: Given a real $m \times n$ matrix X with full column rank n and a real m-vector s, find the n-vector w that solves

$$\min \|s - Xw\|_2, \tag{1}$$

where $\|.\|_2$ denotes the usual Euclidean norm. The solution to (1) is given by

$$w = (X^T X)^{-1} X^T s. \tag{2}$$

If R denotes the upper triangular *Cholesky factor* of the cross product matrix $X^T X$, i.e., $R^T R = X^T X$, then w can be obtained by solving the triangular systems $R^T v = X^T s$, followed by $Rw = v$, where v is an intermediate vector. However, in many applications where accuracy and stability are important [20], R is computed directly from X by a sequence of orthogonal transformations; that is

$$QX = \begin{bmatrix} R \\ 0 \end{bmatrix}, \quad Q^T Q = I. \tag{3}$$

Then setting

$$Qs = \begin{bmatrix} c \\ d \end{bmatrix}, \text{ c an n-vector,} \tag{4}$$

w is computed by solving the upper triangular system $Rw = c$. In addition, the matrix

$$P \equiv (X^T X)^{-1} \tag{5}$$

is called the *covariance matrix* for (1). It measures the expected errors in the least squares vector w. Its inverse

$$P^{-1} = X^T X, \tag{6}$$

often called the *normal equations matrix* [20], is known *as* the *information matrix* for (1) in the signal processing literature [1], [4], [21]. It measures the information content in the experiment leading to (1).

1.2 Recursive Least Squares

In recursive least squares (RLS) it is required to recalculate w when observations (i.e., equations) are successively added to, or deleted from, the problem (1). For example, in many applications information arrives continuously and must be incorporated into the solution *w*. This is called *updating.* It is sometimes important to delete old observations and have their effect excised from w. This is called *downdating* and is associated with a sliding data window. Alternatively, an exponential forgetting factor λ, with $0 < \lambda \leq 1$ (see, e.g., [21]), may be incorporated into the updating computations to exponentially decay the effect of the old data over time. The use of λ is associated with an exponentially-weighted data window [3], [21]. In this paper we will consider the standard updating and downdating methods, which can be associated with applications to sliding window methods in RLS. For details on implementations of these sliding window methods see [2], [4], [7], [8], [12], or [33].

There are two main approaches to solving recursive least squares problems; the *information matrix method* based upon modifying $P^{-1} = X^T X$ and the *covariance matrix method* based upon modifying P [1], [4], [21]. Instead of modifying P^{-1} or P directly, it is generally preferable for stability and cost reasons to modify their Cholesky factors [4]. We will concentrate on the covariance matrix method in this paper. Applications of ale to tracking the conditioning of $P^{-1} = X^T X$ in the information matrix method are similar.

RLS computations arising in adaptive control and signal processing can be described as follows. We assume that the arriving data is prewindowed (e.g., [1], [3]) so that the discrete time index begins at 1. For $k \geq 1$, we let $x(k)$ denote the arriving column data vector of dimension n at time k. At time $m \geq n$, the data matrix X defined in (1) is denoted by $X(m)$. Consequently, the covariance matrix P defined in (5) can be written as

$$P \equiv P(m) = (X(m)^T X(m))^{-1} = \left[\sum_{k=1}^{m} x(k)x(k)^T\right]^{-1}. \tag{7}$$

To initialize the RLS process, P, or P^{-1}, is often set to a scalar multiple of the identity matrix δI.

From (7) it follows that if the data vectors $z(k)$ do not go to zero, then the eigenvalues of the covariance matrix P decrease monotonically. A common procedure, e.g., [1], [3], [21], is to monitor the smallest singular value, $\lambda_{min}(P(m))$, and reset $P(m+1)$ to δI when $\lambda_{min}(P(m))$ falls below some tolerance factor. This process is called *covariance resetting*, or *reinitialization* (e.g., [35] p. 62). Alternatively, a *time window* can be introduced to downweight the influence of past observation or errors, and accordingly, prevent a possible rapid decrease in $\lambda_{min}(P(m))$.

There are two types of time windows: sliding windows and exponentially-weighted windows, the latter of which is a special case of our work and thus will not be discussed further. When a sliding window approach is used, then a fixed window length $l > n$ is chosen. For each time m in the recursive process, the problem is updated by adding the $(m+1)^{st}$ observation and the effect of the old $(m-l+1)^{st}$ observation is completely removed by downdating (e.g., [2]). This sliding window approach approximately doubles the computational complexity of the basic updating process with exponentially-weighted windows, but is well-known to possess favorable convergence or signal tracking characteristics [3], [4], [12].

RLS algorithms based upon modifying the Cholesky factor $L \equiv R^{-T}$ of $P = (X^TX)^{-1}$ are reviewed in Section 2. Adaptive Lanczos based condition estimation schemes for triangular factor updating and downdating are developed in Section 3. Reports on numerical tests with some ill-conditioned data and some actual signal processing data in RLS computations, along with some final comments, are given in Section 4.

2 RLS Sliding Window Computations

We now consider the sliding window recursive least squares (RLS) computations and proceed to simplify the notation. Updating computations are considered first.

At time $m+1$, we set $y = x(m+1)$. Now, consider the least squares problem (1) and, without loss of generality, assume that the additional data vector y and scalar σ for the equation

$$y^Tw \cdot y_1w_1 + \cdots y_nw_n \approx \sigma$$

are appended to X and s forming

$$\widetilde{X} = \begin{bmatrix} X \\ y^T \end{bmatrix}, \quad \tilde{s} = \begin{bmatrix} s \\ \sigma \end{bmatrix}. \tag{8}$$

One now seeks to solve the modified problem

$$\min \|\tilde{s} - \widetilde{X}\tilde{w}\|_2 \tag{9}$$

for the updated least squares estimate vector $\tilde{w}$. This process is then repeated at each recursive time step.

The process of modifying least squares computations by updating the covariance matrix P has been used in control and signal processing for some time in the context of linear sequential filtering [1], [4], [21], [29]. One begins with estimates for $P = R^{-1}R^{-T}$ (where R is the Cholesky factor of X^TX) and w, and updates R^{-1} to $\tilde{R}^{-1}$ and w to $\tilde{w}$ at each recursive time step. Recently Pan and Plemmons [30] have described a parallel scheme for these computations.

Observe first that with $\tilde{X}$ given in (8), $\tilde{P}^{-1}$ is given by

$$\tilde{P}^{-1} = \tilde{X}^T\tilde{X} = X^TX + yy^T = P^{-1} + yy^T.$$

Consequently, by the Sherman-Morrison formula (see, e.g., [20]),

$$\tilde{P} = P - \frac{1}{1 + y^TPy}Pyy^TP. \tag{10}$$

The updated w is given by

$$\tilde{w} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T \begin{bmatrix} s \\ \sigma \end{bmatrix} = \tilde{P}\tilde{X}^T \begin{bmatrix} s \\ \sigma \end{bmatrix}. \tag{11}$$

By substituting (8) and (10) into (11) and using the representation (2) for w, there results the following basic formula for the update:

$$\tilde{w} = w + \tilde{P}y(\sigma - y^Tw). \tag{12}$$

The vector

$$k \equiv \tilde{P}y \tag{13}$$

is often called the Kalman gain vector (see, e.g., [1], [4], [21]). It weights the predicted residual $\sigma - y^Tw$. Updating schemes based upon applying (10) and (12) directly are called *Conventional RLS Algorithms* [1], [3], [4], [21]. Such approaches can lead to numerical difficulties, e.g., loss of symmetry and/or loss of positive definiteness in P. Better numerical results can be expected when the Cholesky factor $L \equiv R^{-T}$ of $P = (X^TX)^{-1}$, rather than P itself, is updated [4], and we adopt that premise in this paper.

Computational schemes for updating the Cholesky factor R typically employ the application of orthogonal plane rotations Q; to zero the update vector y^T. In particular, orthogonal plane rotations $Q_{i,n+1}$, rotating the i^{th} row into the $(n+1)^{st}$ row, are formed for the reduction

$$Q_{n,n+1} \cdots Q_{1,n+1} \begin{bmatrix} R \\ y^T \end{bmatrix} = \begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix},$$

so that the updated matrix $\tilde{R}$ is upper triangular.

Covariance matrix downdating is considered next. Many of the downdating concepts are similar to those for covariance matrix updating, and will only be summarized for the sake of brevity. We assume that, for stability purposes, the updating step is performed before downdating (see, e.g., [30]). Thus before a downdating step we assume that X has been modified to $\widetilde{X}$ and the observation vector s to $\tilde{s}$, by updating the Cholesky factor $L \equiv R^{-1}$ of $P = (X^TX)^{-1}$ to $\tilde{L}$ and the least squares estimate w to $\tilde{w}$. For simplicity of notation, we replace [X, s, L, w] with the updated $\left[\widetilde{X}, \tilde{s}, \tilde{L}, \tilde{w}\right]$ in some of our discussion of downdating to follow.

The downdating computation in RLS with sliding data windows can be described in the following way. Here the purpose is to remove the effect of an observation on the current least squares vector w; that is, to remove a row z^T from the observation matrix X and a scalar η from the observation vector s, corresponding to the observation

$$z^Tw = z_1w_1 + \dots \; z_nw_n \approx \eta.$$

Assume that z^T is the first row of X, so that X and s are related to the downdated $\widehat{X}$ and $\hat{s}$ by

$$\widetilde{X} = \begin{bmatrix} z^T \\ \widehat{X} \end{bmatrix}, \quad \tilde{s} = \begin{bmatrix} \eta \\ \hat{s} \end{bmatrix}. \tag{14}$$

In this case the modified covariance matrix satisfies

$$\hat{P}^{-1} = \widehat{X}^T\widehat{X} = X^TX + yy^T \quad - zz^T = P^{-1} + yy^T - zz^T,$$

where y is the update vector and z the downdate vector. If R is the updated Cholesky factor of X^TX, it follows that the modified Cholesky factor $\hat{R}$ satisfies $\hat{R}^T\hat{R} = R^TR - zz^T$. Assuming that $\widehat{X}$ retains full column rank, *the covariance downdating problem* is now to use the updated inverse Cholesky factor $L \equiv R^{-1}$ of the updated $P = (X^TX)^{-1}$ to compute the downdated inverse factor $\hat{L}$ and the downdated least squares vector $\hat{w}$. But now the orthogonal rotation schemes do not apply directly, due to the negative sign of zz^T. A computationally efficient scheme based upon the use of hyperbolic transformations H_i rather than orthogonal trigonometric plane rotations Q_i can used to compute the downdated factor. For brevity, we refer the reader to the second edition of [20], Section 12.6.4, for a detailed discussion of the use hyperbolic transformations H_i for downdating. We comment that, for stability reasons, these transformations should be implemented as described by Golub [16] (see also [2]).

The following RLS sliding window scheme for modifying the covariance matrix P by updating R^{-T} to $\tilde{R}^{-T}$ followed by downdating to $\hat{R}^{-T}$ and for computing the corresponding modified least squares estimate vector $\hat{w}$ was given in [30] (see also Morf and Kailath [29]). Here we write $P = L^TL$, where $L \equiv R^{-T}$.

Algorithm 1 (RLS by the Sliding Window Covariance Method) . *Given the current n-dimensional least squares estimate vector w, the current lower triangular factor $L \equiv R^{-T}$ of $P = (X^TX)^{-1}$, the observation $\left[y^T, \sigma_I\right]$ being added, and the the observation $\left[z^T, \eta\right]$ being removed, the algorithm computes the modified factor $\hat{L} \equiv \hat{R}^{-T}$ of $\hat{P}$ and the modified least squares estimate vector $\hat{w}$.*

1. *Form the matrix vector product $a = Ly$.*
2. *Choose orthogonal plane rotations $G_{i,n+1}$, rotating the i^{th} component into the $(n+1)^{st}$ component, to form*

$$G_{n,n+1} \cdots G_{1,n+1} \begin{bmatrix} -a \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \delta \end{bmatrix}, \quad \delta = \sqrt{1 + a^T a}, \tag{15}$$

by reducing -a to 0 from the top down, and form

$$G_{n,n+1} \cdots G_{1,n+1} \begin{bmatrix} L \\ 0^T \end{bmatrix} = \begin{bmatrix} \tilde{L} \\ u^T \end{bmatrix}, \tag{16}$$

preserving the lower triangular form of L in $\tilde{L}$.
3. *Compute*

$$\tilde{w} = w - \frac{1}{\delta} u(\sigma - y^T w). \tag{17}$$

4. *Replace $[L, w] \Leftarrow [E, \tilde{w}]$.*
5. *Form the matrix vector product $b = Lz$.*
6. *Choose hyperbolic rotations $H_{i,n+1}$, rotating the i^{th} component into the $(n+1)^{st}$ component, to form*

$$H_{n,n+1} \cdots H_{1,n+1} \begin{bmatrix} b \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \gamma \end{bmatrix}, \quad \gamma = \sqrt{1 - b^T b}, \tag{18}$$

by reducing b to 0 from the top down, and form

$$H_{n,n+1} \cdots H_{1,n+1} \begin{bmatrix} L \\ 0^T \end{bmatrix} = \begin{bmatrix} \hat{L} \\ v^T \end{bmatrix}, \tag{19}$$

preserving the lower triangular form of L in $\hat{L}$.
7. *Compute*

$$\hat{w} = w - \frac{1}{\gamma} v(\eta - z^T w). \tag{20}$$

8. *Replace $[L, w] \Leftarrow [\hat{L}, \hat{w}]$, input the new observation $[y^T, \sigma]$ to be added, let $[z^T, \eta]$ denote the old observation to be removed and return to Step 1.*

Some comments about the algorithm are in order. First, the rotation parameters in (15) and (18) are computed successively using the components of the vectors *a* and *b*. It follows from [30], that u in the update step and v in the downdate step are scaled forms of the Kalman gain vectors associated with updating and downdating. The algorithm requires up to $5n^2$ + *O(n)* multiplications per time step. The $5n^2$ term comes entirely from updating and downdating *L*. No triangular solves are involved.

Our purpose now is to develop an adaptive scheme for monitoring the spectral condition number of L, and accordingly the covariance matrix P, after each recursive time step. An adaptive Lanczos based condition estimation scheme, which we call *ale,* for monitoring condition numbers of the covariance matrices over time by tracking the extreme singular values and singular vectors of the Cholesky factors is described in the next section.

3 ALE: Adaptive Lanczos Estimator

Our purpose is to describe schemes for adaptively monitoring the extreme singular values in RLS updating methods associated with updating and downdating.

The Lanczos method [19], [25] can be used to reduce a nonsymmetric matrix A to a bidiagonal matrix, that is, if $A \in R^{n \times n}$, the Lanczos method computes orthogonal matrices U and V, such that

$$U^T A V = B \tag{21}$$

where B is a lower bidiagonal matrix. It follows that A and B have the same singular values.

It is well known that the information about the extreme singular values trends to emerge long before the bidiagonalization process is complete [18]. In our application, we are only interested in estimating the extreme singular values. We will employ this fast convergence characteristic of the Lanczos method for only a few iterations to approximate the extreme singular values, hence the condition number.

Our purpose is to monitor the spectral condition number of Cholesky factor during low rank modifications; in particular, after rank-one updating and downdating. We then apply the Lanczos method to an $n \times n$ lower triangular Cholesky matrix L in Algorithm 1 for k iterations, computing orthonormal vectors u_j and v_j, such that

$$U(k)^T L V(k) = B(k) \tag{22}$$

where $V(k) = [u_1, u_2, \ldots, u_k]$, $V(k) = [vi, v_2, \ldots, v_k]$, and $B(k)$ is a $k \times k$ lower bidiagonal matrix with the following form

$$B(k) = \begin{pmatrix} \alpha_1 & & & \\ \beta_1 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_{k-1} & \alpha_k \end{pmatrix}. \tag{23}$$

The maximal singular value of L is approximated by the maximal singular value of $B(k)$, that is, $\sigma_{max}(L) \approx \sigma_{max}(B(k))$, *c.f.,* [19]. The minimal singular value of L *can* be approximated in a similar way. The following algorithm describes the scheme for estimating $\sigma_{max}(L)$,

Algorithm 2 (Lanczos Method) . *Given a lower triangular* matrix $L \in R^{n \times n}$, *and an arbitrary vector u_1, $\|u_1\|_2 = 1$, the algorithm computes orthonormal vectors v_j and u_j and the entries of the bidiagonal* matrix *B(k), such that $\sigma_{max}(L) \approx \sigma_{max}(B(k))$.*

```
1. Input [$L, u_1$].
2. Compute $\alpha_1 = \|L^T u_1\|_2$, and set $v_1 = \frac{1}{\alpha_1} L^T u_1$.
3. For $j = 1, 2, \ldots, (k-1)$
   (a) Compute $\beta_j = \|Lv_j - \alpha_j u_j\|_2$, and set $u_{j+1} = \frac{1}{\beta_j}(Lv_j - \alpha_j u_j)$,
   (b) Compute $\alpha_{j+1} = \|L^T u_{j+1} - \beta_j v_j\|_2$, and set $v_{j+1} = \frac{1}{\alpha_{j+1}}(L^T u_{j+1} - \beta_j v_j)$,
   end for
4. Construct lower bidiagonal matrix B(k), and compute $\sigma_{max}(B(k))$.
```

Note that the Lanczos method is applied for k steps ($k - 1$ iterations in the for loop), and this k may be much smaller than the problem size n. For example, in our numerical experiments, we let $k = 2$, and the algorithm generates a 2×2 lower bidiagonal matrix whose maximal singular value can be computed directly by a quadratic formula. If $k > 2$, one can use Newton's method to compute the maximal singular value in a recursive way.

For approximating the minimal singular value of L, since $\sigma_{min}(L) = \frac{1}{\sigma_{max}(L^{-1})}$, one can apply the previous algorithm to L^{-1} instead of L. Of course, L^{-1} is not computed. In the algorithm, a triangular solution is performed, instead of a matrix-vector multiplication.

In the RLS problem, a rank-one modification to the inverse Cholesky factor, $L \equiv R^{-T}$, where R is the Cholesky factor of $X^T X$, is performed; namely,

$$\tilde{L}^T \tilde{L} = L^T L + \rho r r^T, \tag{24}$$

where $\rho = 1$ or -1. Suppose $L^T L$ in (24) has the eigenvalue-eigenvector decomposition

$$L^T L = Q \Lambda Q^T,$$

where $Q^T Q = I$ and $\Lambda = diag(\lambda_i)$. Then

$$\tilde{L}^T \tilde{L} = Q(\Lambda + \rho z z^T) Q^T$$

with $z = Q^T r$. Therefore

$$\sigma^2(\tilde{L}) = \lambda(\tilde{L}^T \tilde{L}). = \lambda(\Lambda + \rho z z^T) \tag{25}$$

Golub [17] has shown that if λ_i are distinct and $q_i^T r \neq 0$ for all i, then the eigenvalues of $\tilde{L}^T \tilde{L}$ can be computed by solving the secular equation

$$\psi(\lambda) = 1 + \rho \sum_{i=1}^{n} \frac{(q_i^T r)^2}{(\lambda_i - \lambda)}, \tag{26}$$

where q_i is the eigenvector of $L^T L$ corresponding to the eigenvalue λ_i. It is also known (see [9], [10]) that the eigenvectors of $\tilde{L}^T \tilde{L}$ can be calculated by the formula

$$q_i = \frac{Q(\Lambda - \tilde{\lambda}_i I)^{-1} Q^T r}{\|(\Lambda - \tilde{\lambda}_i I)^{-1} Q^T r\|_2}. \tag{27}$$

if $\lambda_i \neq \tilde{\lambda}_i$.

Based on these observations, suppose that at time t, we have the estimates $\lambda_1, q_1, \lambda_n$, q_n, and we want to estimate $\tilde{\lambda}_1, \tilde{q}_1, \tilde{\lambda}_n$, and $\tilde{q}_n$ for time $t+1$ by using these information and the updating vector r. In order to amplify in the direction of q_1 and q_n, assume that

$$r = \xi_1 q_1 + \xi_n q_n. \tag{28}$$

Multiplying both sides by q_1^T *and* q_n^T, *one* has

$$\xi_1 = q_1^T r, \qquad \xi_n = q_n^T r.$$

The secular equation is approximated by

$$\hat{\psi}(\lambda) = 1 + \rho \left[\frac{\xi_1^2}{\lambda_1 - \lambda} + \frac{\xi_n^2}{\lambda_n - \lambda} \right]. \tag{29}$$

Solving this quadratic equation $\hat{\psi}(\lambda) = 0$, we obtain the estimates $\tilde{\lambda}_1$, and $\tilde{\lambda}_n$. Then the corresponding singular vectors $\tilde{q}_1$ and $\tilde{q}_n$ can be approximated by using formula (27). The vectors $\tilde{q}_1$ and $\tilde{q}_n$ will be used as the initial vectors in the Lanczos algorithm. In this way, the Lanczos method becomes adaptive. The discussion is summarized in the following algorithm.

Algorithm 3 (Estimating Initial Vectors to make the Lanczos Algorithm Adaptive)

1. *Input* $[\lambda_1, q_1, \lambda_n, q_n, r]$.
2. *Compute* $\xi_1 = q_1^T r, \qquad \xi_n = q_n^T r$.
3. *Compute*

$$\tilde{\lambda}_1 = \frac{1}{2}\left[\lambda_1 + \lambda_n + \rho(\xi_1^2 + \xi_n^2) + \sqrt{(\lambda_1 - \lambda_n)^2 + (\xi_1^2 + \xi_n^2)^2 + 2\rho(\lambda_1 - \lambda_n)(\xi_1^2 - \xi_n^2)}\,\right],$$

$$\tilde{\lambda}_n = \frac{1}{2}\left[\lambda_1 + \lambda_n + \rho(\xi_1^2 + \xi_n^2) - \sqrt{(\lambda_1 - \lambda_n)^2 + (\xi_1^2 + \xi_n^2)^2 + 2\rho(\lambda_1 - \lambda_n)(\xi_1^2 - \xi_n^2)}\,\right].$$

4. *Compute*

$$\tilde{q}_1 = \gamma_1 \left(\frac{\xi_1}{\lambda_1 - \tilde{\lambda}_1} q_1 + \frac{\xi_n}{\lambda_n - \tilde{\lambda}_1} q_n \right),$$

$$\tilde{q}_n = \gamma_n \left(\frac{\xi_1}{\lambda_1 - \tilde{\lambda}_n} q_1 + \frac{\xi_n}{\lambda_n - \tilde{\lambda}_n} q_n \right),$$

where γ_1 *and* γ_n *are chosen, so that* $\|\tilde{q}_1\|_2 = 1$, *and* $\|\tilde{q}_n\|_2 = 1$.

5. Output $[\tilde{q}_1, \tilde{q}_n]$ for the initial *vectors* in *the Lanczos Algorithm.*

Another approach to making the Lanczos scheme adaptive is to use the last u-vector at current time step as the initial u-vector for the next time step. In numerical experiments we have observed that this scheme often yields good accuracy. Another advantage of this approach is that the computational complexity is reduced by $\frac{1}{2}n^2$ multiplications. For suppose we perform k iterations in the Lanczos algorithm, save vector u_k produced by the algorithm, and set $\tilde{u}_1 = u_k$ for the next time step. At time $t + 1$, we perform orthogonal or hyperbolic transformations on the vector $L^T u_k$. Note that this matrix-vector multiplication has been performed at time t in the Lanczos algorithm. Letting T denote the product of orthogonal or hyperbolic rotations, we have

$$T\begin{bmatrix} L^T u_k \\ r^T u_k \end{bmatrix} = T\begin{bmatrix} L^T \\ r^T \end{bmatrix} u_k = T\begin{bmatrix} L^T \\ r^T \end{bmatrix} \tilde{u}_1 = \begin{bmatrix} \tilde{L}^T \\ 0^T \end{bmatrix} \tilde{u}_1 = \begin{bmatrix} \tilde{L}^T \tilde{u}_1 \\ 0 \end{bmatrix}. \quad (30)$$

Hence one can obtain the vector $\tilde{L}^T \tilde{u}_1$ after the modification process, and set $\tilde{\alpha}_1 = \|\tilde{L}^T \tilde{u}_1\|_2$ without performing the matrix-vector multiplication.

Actually one can combine Algorithm 3 with this approach, so that the adaptive Lanczos estimate algorithm can be described as follows:

Algorithm 4 (Adaptive Lanczos Algorithm) *Given a lower triangular matrix $L \in R^{n \times n}$, the algorithm computes orthonormal vectors v_j and u_j and the entries of the bidiagonal matrix B(k), such that $\sigma_{max}(L) \approx \sigma_{max}(B(k))$*

```
1. Input $[L, u_1]$ or $[L, u_1, L^T u_1]$.
2. Compute $\alpha_1 = \| L^T u_1\|_2$, and set $v_1 = \frac{1}{\alpha_1} L^T u_1$.
3. For $j = 1, 2, \ldots, (k-1)$
    (a) Compute $\beta_j = \|Lv_j - \alpha_j u_j\|_2$, and set $u_{j+1} = \frac{1}{\beta_j}(Lv_j - \alpha_j u_j)$,
    (b) Compute $\alpha_{j+1} = \|L^T u_{j+1} - \beta_j v_j\|_2$, and set $v_{j+1} = \frac{1}{\alpha_{j+1}}(L^T u_{j+1} - \beta_j v_j)$,
   end for
4. Construct bidiagonal matrix B(k), and compute $est(\sigma_{max}(L)) = \sigma_{max}(B(k))$.
5. Apply Algorithm 3, to obtain a better estimation for maximal singular vector for next
   time step, or save $L^T u_k$ from the last iteration in the for loop.
6. Output $[est(\sigma_{max}(L)), u_k]$, or $[est(\sigma_{max}(L)), u_k, L^T u_k]$.
```

The computational complexity of this adaptive Lanczos algorithm is still $(k-1)n^2 + O(n)$ multiplications.

Algorithm 4 must be applied at each update and each downdate in order to track the conditioning of the covariance matrix P, but for simplicity we only reprort estimates after each sliding window time step, i.e., after each downdate. Similar modifications can be made for estimating $\sigma_{min}(\tilde{L})$. 0 nce we obtain the estimates for $\sigma_{max}(L)$, and $\sigma_{min}(L)$, then the spectral condition number of the covariance matrix P is approximated by

$$K_2(P) \approx \left[\frac{est(\sigma_{max}(L))}{est(\sigma_{min}(L))}\right]^2 .$$

AS a side benefit, the recursive estimates of $\lambda_{min}(P)$ and $\lambda_{max}(P)$ also provide approximate bounds for the *power spectral density* of the least squares process in the context of adaptive filtering, as described by Haykin [21], pp. 62-66. The corresponding singular vectors also provide useful information in the context of Pisarenko harmonic decomposition, e.g., [23].

4 Comparisons and Numerical Experiments

In this section we report on some selected experiments designed to compare the performance of the adaptive condition estimation scheme, *ale,* associated with Algorithm 4 with two alternative methods, one due to Pierce and Plemmons [31] and called ace and the other due to Bischof [5] and called ice. The scheme ace is adaptive and ace requires up to $38n$ multiplication per update/downdate sliding window time step, while ice is non-adaptive and requires up to $6n^2 + O(n)$ multications - the same as our adaptive Lanczos estimator *ale* using the secular equation approximations to obtain starting vectors. If the alternative approach of using the last u-vector at current time step as the initial u-vector for the next time step is employed, then the complexity of *ale* is reduced to $4n^2 + O(n)$ multiplications. It follows that ace is cheaper than this version of *ale only* when $n \geq 7$.

Some data provided in by A. Björck, some randomly generated data, and some actual data from signal processing adaptive filtering are considered in our tests and comparisons. The datasets used the tests are described as follows:

- **Random data from A. Björck [8].** This dataset consists of $m = 50$ observations of $n = 5$ components each, randomly generated with a uniform distribution in $(0,1)$. An outlier equal to 10^4 is added to position $(18,3)$. A sliding window length of 8 is used.

- **Hilbert matrix data from Å. Björck [8].** Again $m = 50$ observations of $n = 5$ components each are generated. This time the fist 25 observations consist of the first 5 columns of the Hilbert matrix of dimension 25, and the same rows in reversed order as the last 25 observations. A random perturbation, uniformly distributed in $(0, 10^{-9})$, is added to the 10 middle rows to prevent the sliding window submatrices, which have window length 8, from becoming singular.

- **Random ill-conditioned data from Pierce and Plemmons [32].** Here $m = 100$ observations of $n = 10$ components each is generated with a random distribution in

$(-50, 50)$. Ill-conditioning is forced by multiplying one component of each observation by 10^{-3} to force near column dependency for the sliding window submatrices. The window length is set at 20.

- **Signal processing data from the NC State University Center for Communications and Signal Processing.** This signal processing data is generated from a well-conditioned second-order autoregressive process. A total of $m = 200$ observations of $n = 5$ components each are considered. A sliding window length of 20 is used.

We report only on the performances of *ale* (in comparison to the alternatives ace and ice), in recursively estimating the condition number of the covariance matrix P for each update/downdate step. Our purpose in these tests is to exhibit the reliability and the accuracy of *ale* for recursive least squares computations. All experiments were performed using the Pro-Matlab system [27]. The results of these experiments are given in Figures 1-4. In the graphs, the solid lines represent *ale*, the dashed lines represent ace, and the dotted lines represent *ice*. As can be seen from the figures, *ale* compares very favorably with ace and ice in terms of accurately tracking the condition number, $K_2(P)$, of the covariance matrix over time.

Acknowledgement The authors wish to thank A. Björck in the Mathematics Department at Linkoping University, Sweden, for providing test data. We also wish to thank Avinash Ghimikar and Tulay Adali in the N.C. State University Electrical and Computer Engineering Department for supplying the signal processing data used our tests.

References

[1] S.T. Alexander, **Adaptive Signal Processing,** Springer Verlag, New York (1986).

[2] S.T. Alexander, C.-T. Pan and R.J. Plemmons, *Analysis of a recursive least squares* hyperbolic *rotation scheme for signal processing,* **Lin. Alg. Appl. Special Issue on Electrical Eng.** 98 (1988) 3-40.

[3] K.J. Åström and B. Wittenmark, **Adaptive Control,** Addison-Wesley, Reading MA (1989).

[4] G.J. Bierman, **Factorization Methods for Discrete Sequential Estimation,** Academic Press, NY (1977).

[5] C.H. Bischof, *Incremental* condition *estimation,* **SIAM J. on Matrix Anal. and Applic.** to appear April (1990).

[6] C.H. Bischof and G.M. Shroff, *On updating signal subspaces,* Tech. Rept., Argonne National Lab. September (1989).

[7] Å. Björck, Least Squares Methods, in *Handbook of Numerical Analysis,* P.Ciarlet and J. Lions, eds., Elsevier/North Holland, Amsterdam (1989).

[8] Å. Björck, Accurate downdating *of least squares solutions,* presented at the Annual SIAM Meeting, held at San Diego, CA (1989).

[9] J.R. Bunch, C.P. Nielson, Updating *the* Singular *Vaule Decomposition,* Numer. Math., 31 (1978) 111-129.

[10] J.R. Bunch, C.P. Nielson, and D.C. Sorensen, *Rank-One Modification Of the Symmetric Eigenproblem,* Numer. Math. 31 (1978) 31-48.

[11] J.M. Cioffi, Limited precision effects in *adaptive filtering,* IEEE Trans. on CAS 34 (1987) 821-833.

[12] J.M. Cioffi and T. Kailath, Windowed *fast transversal filters,* adaptive *algorithms* with *normalization,* IEEE Trans. on Acous., Speech and Sig. Proc. 33 (1985) 607-625.

- [13] P. Comon and G.H. Golub, *Tracking a few extreme singular values and vectors* in signal processing, Stanford Univ. Tech. Rept. NA-89-01 (1989).

[14] R.D. DeGroat and R.A. Roberts, *Highly parallel* eigenvector *update methods* with *applications to* signal processing, SPIE Adv. Alg. and Arc. for Signal Proc. 696 (1986) 62-70.

[15] P.E. Gill, G.H. Golub, W. Murray and M.A. Saunders, *Methods for modifying matrix* factorizationa, Math. Comput. 28 (1974) 505-535.

[16] G.H. Golub, Matrix *decompositions and statistical calculations,* in R.C. Milton and J.A. Nelder, Eds., Statistical Computation, Academic Press, New York, (1969) 365-395.

[17] G.H. Golub, Some modified *matrix* eigenvalue *problems,* SIAM Review 15 (1973) 318-334.

[18] G.H. Golub and W. Kahan, *Calculating the singular values and pseudo inverse of a matrix,* SIAM J. Num. Anal. Ser. B2 (1965) 205-224.

[19] G.H. Golub, F.T. Luk and M.L. Overton, *A block Lanczos method for computing the singular values and corresponding* singular *vectors of a matrix,* ACM Transcation on Mathematical Software 7 (1981) 149-169.

[20] G.H. Golub and C. Van Loan, Matrix Computations, Johns Hopkins Press, Baltimore, Second Edition (1989).

[21] S. Haykin, Adaptive Filter Theory, Prentice-Hall, Englewood Cliffs, NJ (1986).

[22] C.S. Henkel and R.J. Plemmons, *Recursive least squares computations on a hypercube multiprocessor* using *the covariance factorization,* SIAM J. Sci. Statist. Comp., to appear (1990).

[23] N.J. Higham, *A survey of* condition *estimators for triangular matrices,* SIAM Review 29 (1987), 575-596.

[24] Y.H. Hu, Adaptive *methods for real time Pisarenko spectrum estimation,* Proc. ICASSP, Tampa, FL (1985).

[25] C. Lauczos, *An* iteration *method for the solution of the eigenvalue problem of linear differential* and integral *operators,* J. Res. Nat. Bur. Stand. 45 (1950) 255-281.

[26] F. Milinazzo, *C.* Zala and I. Barrodale, *On the rate of growth of condition numbers for* convolution matrices, IEEE Trans. Acous., Speech and Sig. Proc. 35 (1987) 471-475.

- [27] C. Moler, J. Little and S. Bangert, Pro-Matlab Users Guide, The Mathworks, Sherbom, MA (1987).

[28] M. Moonen, P. van Dooren and J. Vandewalle, *Parallel updating algorithms, part one: The ordinary* singular *value* decomposition, preprint (1989).

[29] M. Morf and T. Kailath, *Square root algorithms and least squares estimation,* IEEE Trans. on Automatic Control 20 (1975) 487-497.

[30] C.-T. Pan and R.J. Plemmons, *Least squares modifications with* inverse *factorizationa: parallel* implications, Computational and Applied Math. 27 (1989) 109-127.

[31] D.J. Pierce and R.J. Plemmons, *Fast adaptive condition estimation,* preprint (1990).

[32] D.J. Pierce and R.J. Plemmons, *Fast* adaptive *condition estimation for RLS in signal* processing, *I: Exponential* weighting, submitted (1990).

[33] D.J. Pierce and R.J. Plemmons, *Fast* adaptive *condition estimation, II: Sliding win*dows *for RLS* in signal processing, in preparation (1990).

[34] R.J. Plemmons, *Recursive least squares computations,* Proc. Inter. Conf. on Math. in Networks and Syst., Birkhauser Boston, Inc., to appear (1990).

[35] S. Sastry and M. Bodson, Adaptive Control: Stability, Convergence, and Robustness, Prentice-Hall, Englewood Cliffs, NJ, (1989).

[36] R. C. Thompson, ***The Behavior** of **Eigenvalues and*** Singular ***Values under Perturbations** of* Restricted ***Rank,*** **Linear Algebra and Its Applications** 13 (1976) 69-78.

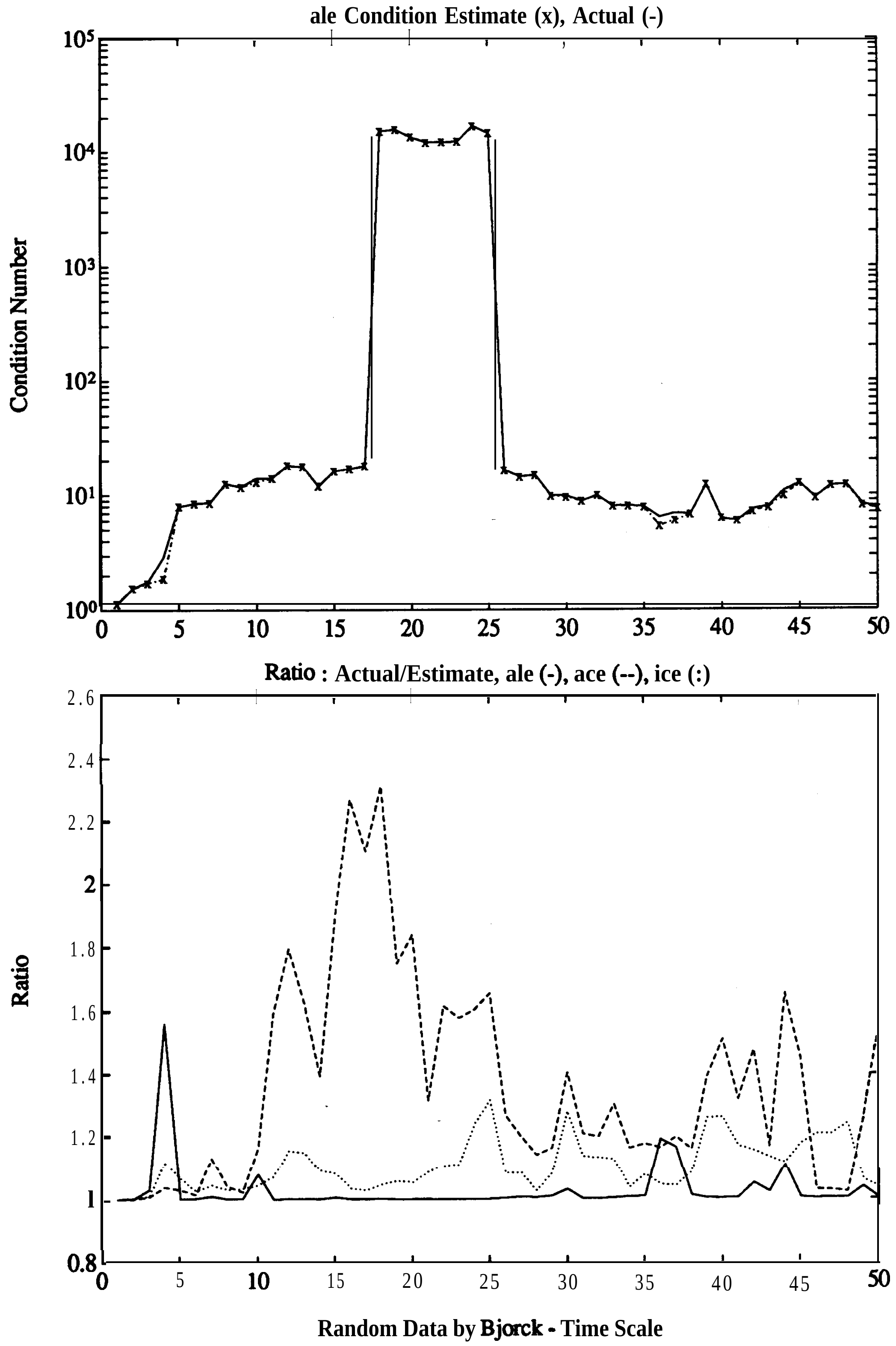
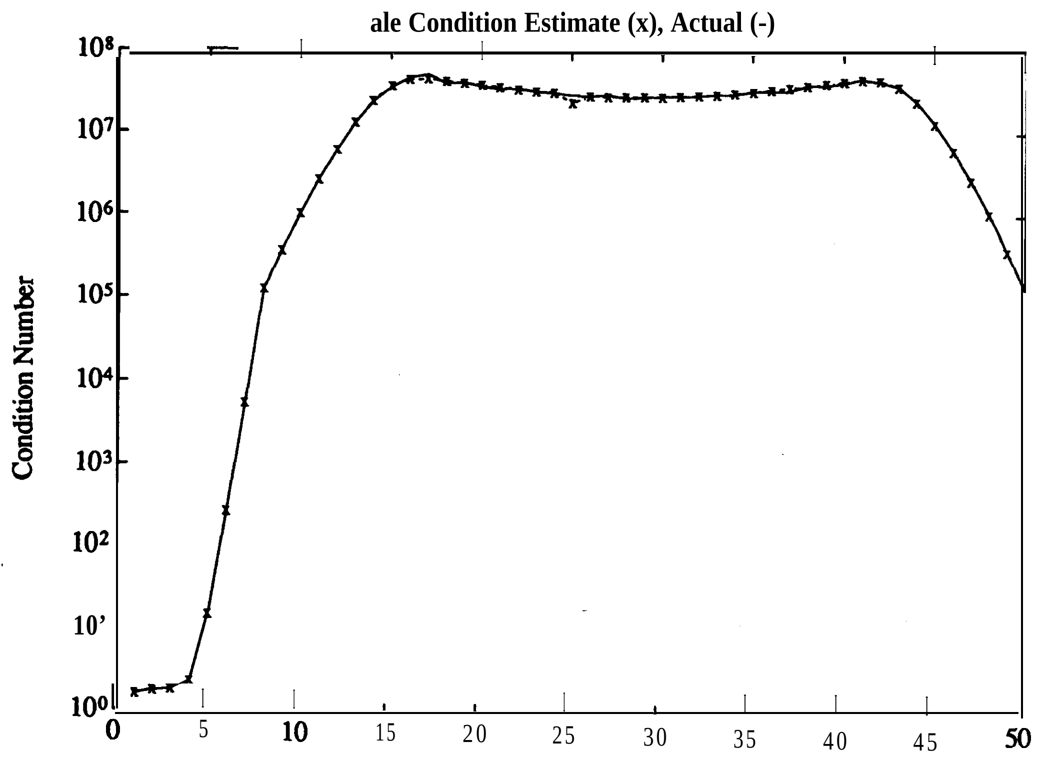


Figure 1: Comparison of ale with ace and ice on random data with outlier from Å. Björck.



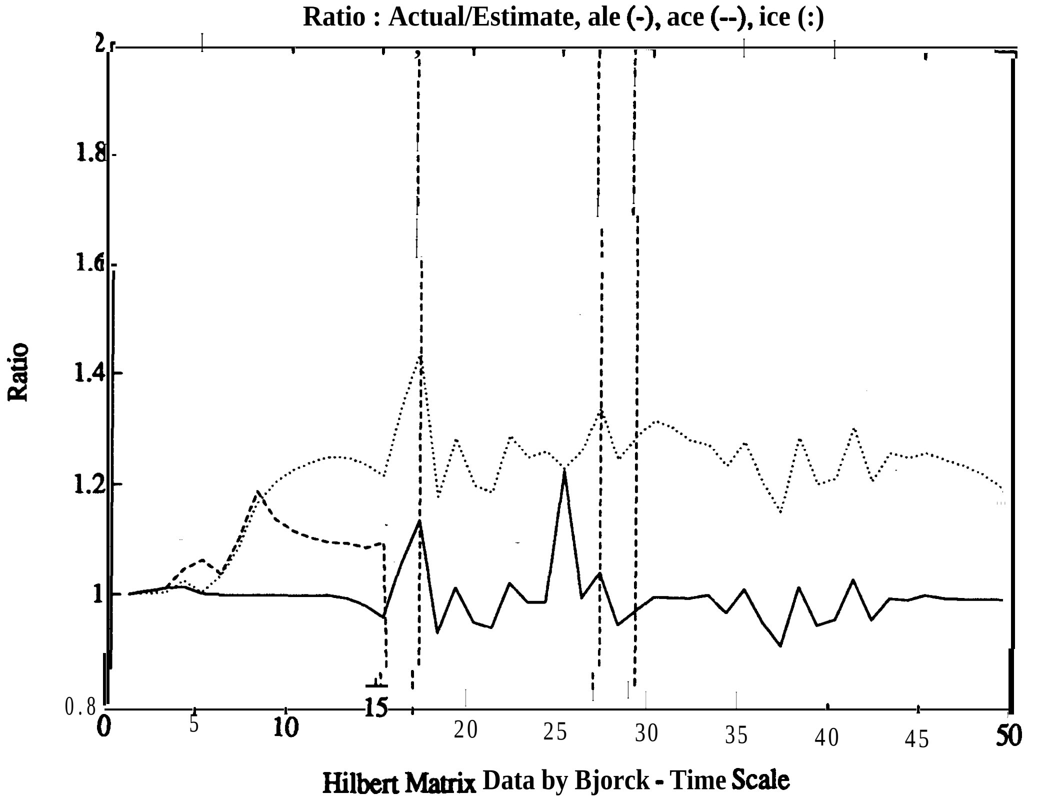


Figure 2: **Comparison of ale with ace and ice on Hilbert matrix data from Å. Björck.**

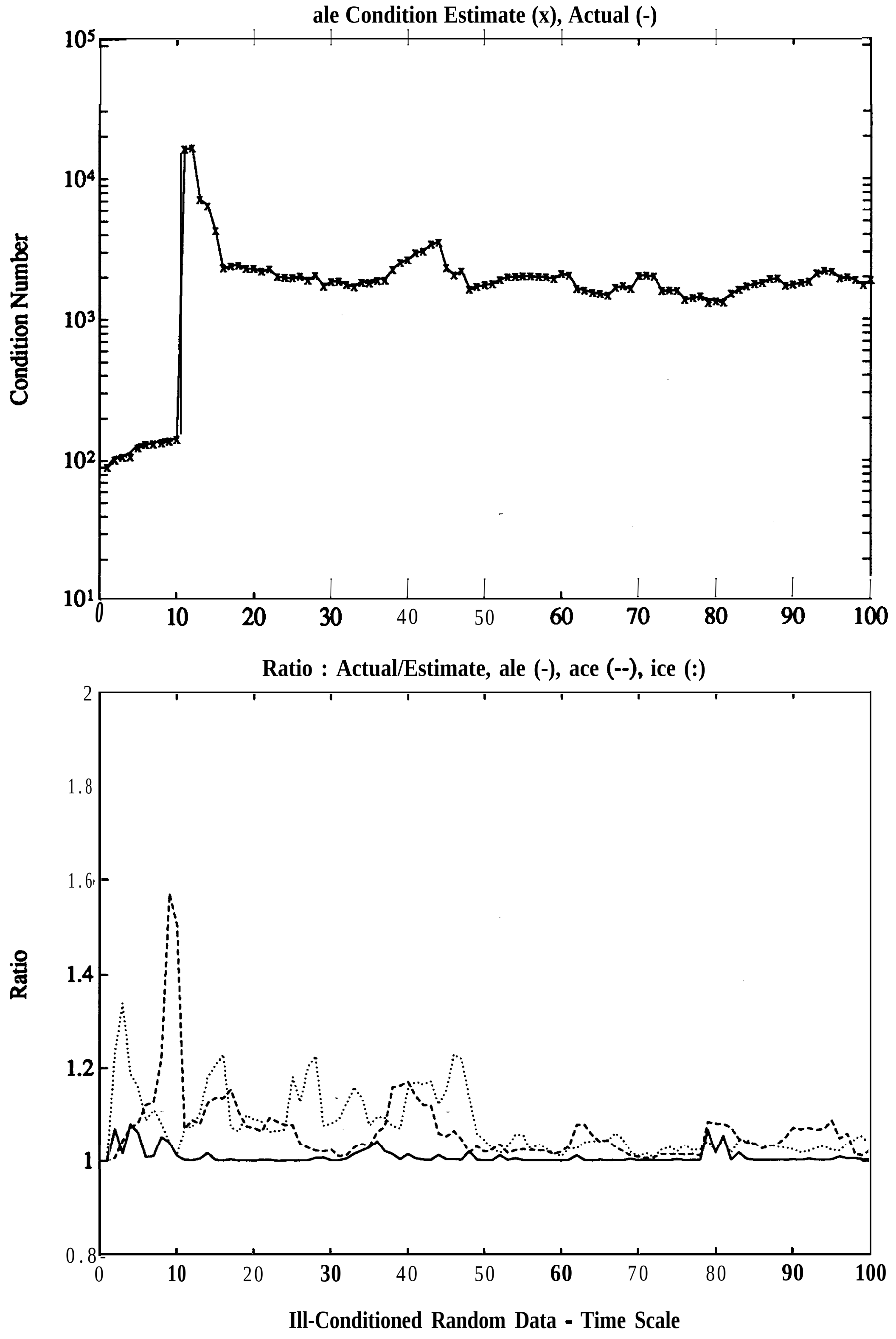


Figure 3: **Comparison of** ale **with ace and ice on random moderately ill-conditioned data from Pierce and Plemmons.**

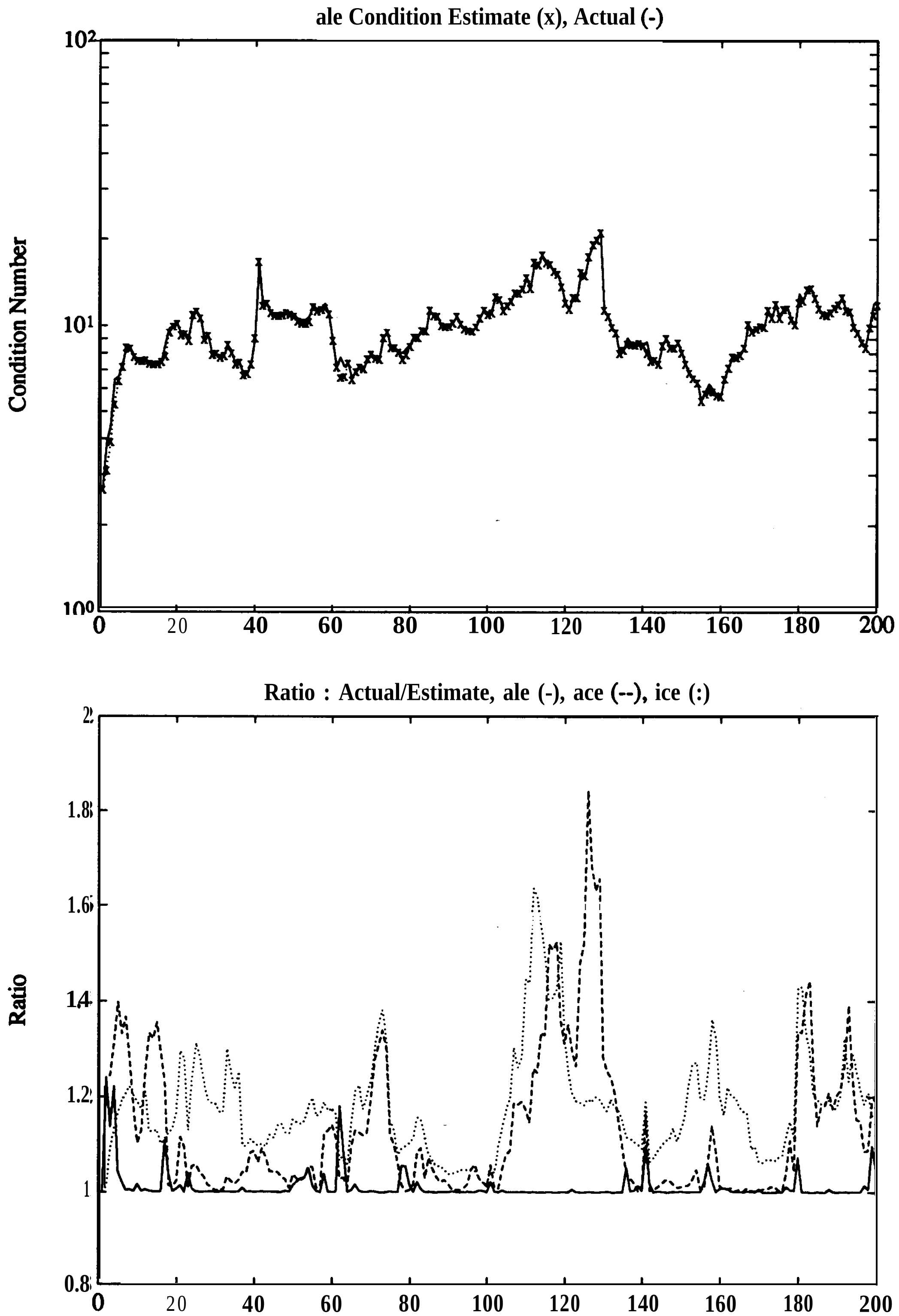


Figure 4: Comparison of ale with ace and *ice* on signal processing data.