JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 18, No. 3, May–June 1995

Efficient Computation of the Eigensystem Realization Algorithm

L. D. Peterson
University of Colorado, Boulder, Colorado 80309-0429

The computational efficiency of an algebraically equivalent alternative formulation of the eigensystem realization algorithm is investigated. This alternative can process large data sets much faster than the form of the eigensystem realization algorithm that is ordinarily used. This is primarily due to the partial eigenvalue decomposition of the inner or outer product of the Hankel matrix and its transpose, whichever is smaller, in place of the full singular value decomposition of the Hankel matrix itself or of the Hankel matrix square. By computing only the largest subset of the Hankel matrix product eigenvalues, low-order models can be realized from very large data sets. The primary benefit of this is improved convergence on the model pole locations for practical modal identification problems using hundreds of sensors to identify hundreds of modal resonances. This paper investigates the computational speed increase over the original algorithm on experimental data and assesses the magnitude of model errors introduced by squaring the Hankel matrix. In addition, key computational strategies are presented for efficient implementation of the algorithm on modern computer architectures.

Nomenclature

$\{A, B, C, D\}$	= discrete model state-space matrices
D_q	= inner product of the Hankel matrix, $H_{qd}(0)H_{qd}^T(0)$
D_q^*	= partition of D_q, $D_q(n_y + 1 : qn_y, 1 : (q-1)n_y)$
$H_{qd}(k-1)$	= Hankel matrix with q block rows and d block columns at time $k-1$
k	= sample index
$M(k)$	= Markov parameters
n_u	= number of inputs
n_x	= number of states in the model
n_y	= number of measurements
$\tilde{P}_q, S_{qd}, \tilde{Q}_q$	= spectral factors of the Hankel matrix
$\tilde{P}_q, \tilde{S}_{qd}, \tilde{Q}_q$	= spectral Factors of the D_q matrix
q, d, r, s	= block dimensions of Hankel matrices
$u(k)$	= vector of applied forces
V_q	= observability factor of the Hankel matrix
W_d	= controllability factor of the Hankel matrix
$x(k)$	= state vector
$y(k)$	= vector of measured responses

Superscripts

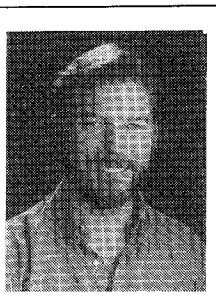
T	= matrix transpose
$+$	= pseudo-inverse

Introduction

SYSTEMATIC modal identification methods such as the eigensystem realization algorithm (ERA)[1] have advanced the complexity of structures that can be modeled by experimental measurements. Using a theoretical result from system realization theory, ERA fits a minimal order discrete state-space model of the modal dynamics to measured impulse response data. Modal frequencies, damping ratios, and vectors can be extracted from the state-space model and can then be used to revise a finite element model, to detect changes in the structural dynamics due to damage, or to design a control system for vibration damping. ERA has become a recognized and applied modal data analysis procedure.

A major advantage of ERA over other system realization methods is that theoretically it finds the smallest order model that fits the data within a given accuracy. This is accomplished through the singular value decomposition (SVD) of a large Hankel matrix formed from impulse response measurements. By discarding all but the largest singular values of the Hankel matrix, the ERA user can separate measurement components that are due to true modal dynamics from components that are due to measurement noise or weakly excited modes.[2] This makes ERA an effective method for analyzing data from structures with repeated or closely spaced modal frequencies, with or without noise in the data.

Modern modal analysis experiments, however, can task the practical limits of ERA. It is becoming commonplace for data from a single experiment to be collected from hundreds of output measurements for many applied inputs. In addition, the need for precise models of complex structures are requiring the identification of several hundred modal resonances. Users of ERA may be familiar with the fact that the poles of such modally complex structures will converge only if several thousand time samples are included in the ERA Hankel matrix to overspecify the realization problem. It would not be unusual for this type of system identification problem to require



Lee D. Peterson received his S.B., S.M., and Ph.D. degrees in Aeronautics and Astronautics from the Massachusetts Institute of Technology in 1982, 1983, and 1987 respectively. From 1987 to 1989 he was a Member of the Technical Staff at Sandia National Laboratories, and from 1989 to 1991 he was an Assistant Professor of Aeronautics and Astronautics at Purdue University. Since 1991, he has been an Assistant Professor in Aerospace Engineering Sciences at the University of Colorado. His research interests include experimental modal analysis and precision deployable spacecraft structures. He is a Senior Member of AIAA and an Associate Member of ASME.

Received April 8, 1993; revision received March 12,1994; accepted for publication June 22, 1994.

Hankel matrices with dimensions on the order of several thousand rows by several tens of thousands of columns. For a Hankel matrix this size, the computational and memory requirements of the original ERA procedure become impractical, and so the algorithm itself becomes inapplicable.

The purpose of this paper is to investigate the numerical and computational performance of an alternative formulation of the ERA procedure. This algebraically equivalent procedure is of interest because it potentially increases both the practical size and speed of ERA so as to accommodate large data sets that would otherwise be infeasible. For example, a data set with thousands of Hankel matrix rows and tens of thousands of Hankel matrix columns can be processed in the same time the original ERA would identify a data set a fraction of the size. Conversely, small data sets can be analyzed in only a few seconds, making near-real-time modal identification possible using ERA.

The key to this capability is an algebraically equivalent form of ERA that reduces the computational effort of the procedure. The basic form of this alternative formulation can be found in several ways: possibly as a special case of the ERA/Data Correlations (ERA/DC)[4] algorithm, or as the Hankel approximation to a Q-Markov covariance equivalent realization, referred to as HQMC.[3] Both are equivalent. The fact that the special form of ERA/DC potentially required fewer computations than ERA was suggested by the developers of that algorithm.[4] Because of this relation to other algorithms, this paper makes no claim about the introduction of a wholly new approach. Instead, the purposes of this paper are to show a new and particularly efficient computational variation of the algorithm, to measure and evaluate its performance on real modal data, to examine numerical and computational issues that affect its practical application, and to discuss efficient implementation procedures that maximize its performance.

The primary advantage of the efficient ERA examined in this paper is that it reduces the size and computational effort of the singular value decomposition problem in either ERA or ERA/DC. ERA computes the full singular value decomposition of the rectangular Hankel matrix, whereas ERA/DC (or HQMC) computes the full singular value decomposition of the symmetric inner product of the Hankel matrix. In contrast, the approach presented here computes only the partial eigendecomposition of the symmetric inner or outer product of the Hankel matrix and its transpose, whichever results in the smaller dimension matrix. Because the Hankel matrix product or its transpose product is symmetric positive semi definite, its singular value decomposition can be found using symmetric eigenanalysis algorithms. The eigenvalues of the Hankel matrix product are the square of the nonzero singular values of the Hankel matrix. This makes it possible to use partial eigensolvers in place of the full singular value decomposition. The advantage of this is that, even though the Hankel matrix might have several thousand nonzero singular values, it is very rare that more than several hundred modes would be included in a reasonable model. This paper shows that partial eigensolvers can find the largest hundred or so singular values of the symmetric Hankel inner or outer matrix product much faster than a singular value decomposition algorithm can find the complete nonzero spectral decomposition of either the Hankel matrix, as used in ERA, or the Hankel matrix square, as used in ERA/DC.

In view of this, this Paper provides two main contributions. The first is a presentation and investigation of the use of the partial eigensolution of the symmetric Hankel matrix inner or outer product in place of the full singular value decomposition of the full rectangular Hankel matrix. The speed increment and memory efficiency are measured for real modal test data as a function of the Hankel matrix dimensions. These are compared for the original ERA, the faster ERA with a full Hankel eigendecomposition (which is equivalent to ERA/DC), and the faster ERA with a partial Hankel eigendecomposition. The second contribution is an investigation of whether the efficiency gain will come at a sacrifice in numerical accuracy of the poles that are retained in the model. The paper demonstrates that the real parts of identified system poles are biased towards more negative values by the squaring of the Hankel matrix in an amount approximately the order of the machine numerical precision. Consequently, the significant increase in speed over the original ERA computation will be at a negligible loss in accuracy.

The paper is organized as follows. The first section reviews the system realization approach to modal analysis and introduces the relevant notation. The second and third sections present ERA and its efficient formulation, which for the purpose of this paper is referred to as FastERA. In addition, a transpose dual formulation is presented to be used in the case where the corresponding Hankel matrix is tall rather than wide. The fourth section presents and discusses effective numerical procedures that lead to increased computational efficiency in the algorithm. Optimized Fortran based code, using the BLAS[8] and LAPACK[9] replacements to EISPACK[7] and LINPACK,[10] is the basis for this comparison. The execution speed and numerical precision of these procedures on experimental data is presented and discussed in the last section.

System Realization Problem

Most modern modal analysis methods first find a discrete time model of the modal dynamics and then transform this model into the continuous time domain for extracting the modal frequencies, damping ratios, and vectors. Finding this discrete time model is the objective of system realization. This section reviews the fundamental formulations of system realization and introduces notation that is necessary in the following sections.

Any linear sampled data system can be modeled by the following state-space realization:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{aligned} \tag{1}$$

Starting from zero initial conditions, the response of this model to any vector of inputs $\boldsymbol{u}(k)$ is a discrete convolution sum:

$$\boldsymbol{y}(k) = \sum_{i=1}^{k} \boldsymbol{M}(k-i)\boldsymbol{u}(i) \tag{2}$$

in which the Markov parameters are related to the system matrices by

$$\boldsymbol{M}(k) = \begin{cases} \boldsymbol{D}, & k = 0 \\ \boldsymbol{C}\boldsymbol{A}^{k-1}\boldsymbol{B}, & k > 0 \end{cases} \tag{3}$$

The Markov parameters are the same as the response of the system to a unit sample input. For instance, the jth column of $\boldsymbol{M}(k)$ is the response to a unit sample applied to the jth input, with all other inputs held to zero. These quantities can be measured by any of a number of established methods, as they are the inverse Fourier transforms of the measured frequency response functions from each input to each output. A common method for determining the frequency response functions is the averaged cross-correlation formulation found in Bendat and Piersol.[5] Another approach might consider the OKID algorithm.[11]

The system realization problem is stated as follows. Given the sequence of Markov parameters,

$$\{\boldsymbol{M}(k), 0 \leq k < \infty\}$$

find the state-space matrices for the model

$$\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}\}$$

by maximizing some measure of the model's accuracy. The relevant issues in realization include the uniqueness of the model, the minimality of the order of the model (the dimension of $\boldsymbol{A}$), the effect of noise, the choice of cost function in the minimization, and the computational burden. Here we restrict the discussion to the latter consideration.

It is useful to define several matrices that are common to the methods discussed later. The Hankel matrix is defined to be

$$H_{rs}(k-1) = \begin{bmatrix} M(k) & M(k+1) & \cdots & M(k+s-1) \\ M(k+1) & M(k+2) & \cdots & M(k+s) \\ \vdots & \vdots & \ddots & \ddots \\ M(k+r-1) & M(k+r) & \cdots & M(k+r+s-2) \end{bmatrix} \quad (4)$$

The significance of this matrix is that in the limit of r and s becoming very large, the numerical rank of this matrix is the smallest order model that can reproduce the Markov parameters in the data. From the preceding definitions, we can relate this matrix to the unknown model state-space matrices. In particular,

$$H_{rs}(k-1) = V_r A^{k-1} W_s \quad (5)$$

in which

$$V_r = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{r-1} \end{bmatrix} \qquad W_s = [B \quad AB \quad \cdots \quad A^{s-1}B] \quad (6)$$

These matrices are also significant to the system identification process. In particular, V_r can be used to determine the observability of the modes and W_s can be used to determine the controllability of the modes.

Eigensystem Realization Algorithm (ERA) and Its Computational Procedure

The eigensystem realization algorithm (ERA)[1] is based on the shift of the Hankel matrix from time $k = 1$ to $k = 2$. From Eqs. (4) and (5), the Hankel matrix at $k = 1$ is

$$H_{rs}(0) = \begin{bmatrix} M(1) & M(2) & \cdots & M(s) \\ M(2) & M(3) & \cdots & M(s+1) \\ \vdots & \vdots & \ddots & \vdots \\ M(r) & M(r+1) & \cdots & M(r+s-1) \end{bmatrix} = V_r W_s \quad (7)$$

and the Hankel matrix at $k = 2$ is

$$H_{rs}(1) = \begin{bmatrix} M(2) & M(3) & \cdots & M(s+1) \\ M(3) & M(4) & \cdots & M(s+2) \\ \vdots & \vdots & \ddots & \vdots \\ M(r+1) & M(r+2) & \cdots & M(r+s) \end{bmatrix} = V_r A W_s \quad (8)$$

A least-squares solution of Eq. (8) for A is

$$A = V_r^+ H_{rs}(1) W_s^+ \quad (9)$$

The key to ERA is that the observability and controllability factors V_r and W_s can be computed from $H_{rs}(0)$ by its decomposition into its singular vectors:

$$H_{rs}(0) = \tilde{P}_r S_{rs} \tilde{Q}_s^T \quad (10)$$

Then,

$$V_r = \tilde{P}_r S_{rs}^{\frac{1}{2}}, \qquad W_s = S_{rs}^{\frac{1}{2}} \tilde{Q}_s^T \quad (11)$$

and

$$V_r^+ = S_{rs}^{-\frac{1}{2}} \tilde{P}_r^T, \qquad W_s^+ = \tilde{Q}_s S_{rs}^{-\frac{1}{2}} \quad (12)$$

And so the A matrix is given by

$$A = S_{rs}^{-\frac{1}{2}} \tilde{P}_r^T H_{rs}(1) \tilde{Q}_s S_{rs}^{-\frac{1}{2}} \quad (13)$$

The following procedure is used to compute the ERA realization from a given set of Markov parameters:

1) Form the Hankel matrix $H_{rs}(0)$:

$$H_{rs}(0) = \begin{bmatrix} M(1) & M(2) & \cdots & M(s) \\ M(2) & M(3) & \cdots & M(s+1) \\ \vdots & \vdots & \ddots & \vdots \\ M(r) & M(r+1) & \cdots & M(r+s-1) \end{bmatrix}$$

2) Decompose $H_{rs}(0)$ into its singular value factors:

$$H_{rs}(0) = \tilde{P}_r S_{rs} \tilde{Q}_s^T$$

3) Retain only the n_x nonzero singular values S_{rs} and corresponding singular vectors.

4) Form $H_{rs}(1)$:

$$H_{rs}(1) = \begin{bmatrix} M(2) & M(3) & \cdots & M(s+1) \\ M(3) & M(4) & \cdots & M(s+2) \\ \vdots & \vdots & \ddots & \vdots \\ M(r+1) & M(r+2) & \cdots & M(r+s) \end{bmatrix}$$

5) Solve for A:

$$A = S_{rs}^{-\frac{1}{2}} \tilde{P}_r^T H_{rs}(1) \tilde{Q}_s S_{rs}^{-\frac{1}{2}}$$

6) Solve for B:

$$B = S_{rs}^{\frac{1}{2}} \tilde{Q}_s^T (1 : n_x, 1 : n_u)$$

7) Solve for C:

$$C = \tilde{P}_s (1 : n_y, 1 : n_x) S_{rs}^{\frac{1}{2}}$$

8) Solve for D:

$$D = M(0)$$

Efficient Formulation of ERA

This section describes the alternative formulation of ERA that finds the realization from the partial decomposition of the Hankel matrix product instead of the Hankel matrix itself. For the purpose of this discussion, this alternative is referred to as FastERA. The first subsection shows the computation for the A matrix, which is found using shifts in the partitions of the observability matrix. The next subsection describes an efficient method for finding the Hankel matrix product. The third subsection relates FastERA to ERA, and the fourth subsection provides a step-by-step procedure for fastERA.

A-Matrix Computation

First we define the following data matrix D_q to be the square of the Hankel matrix:

$$D_q = H_{qd}(0) H_{qd}(0)^T \quad (14)$$

From Eq. (7), this is the same as

$$D_q = V_q W_d W_d^T V_q^T \quad (15)$$

We can, without loss of generality, presume a coordinate basis for the state-space model for which the square of the controllability factor is identity:

$$W_d W_d^T = I \quad (16)$$

which means

$$D_q = V_q V_q^T \quad (17)$$

The A matrix for FastERA can be found by examining the time shift of the observability matrix V_q. From Eq. (6), the first $q-1$ row blocks of V_q are

$$V_q^{(1)} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{q-2} \end{bmatrix} \tag{18}$$

and the last $q-1$ row blocks of V_q are

$$V_q^{(2)} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{q-1} \end{bmatrix} \tag{19}$$

These $q-1$ row block partitions of V_q are related by the following shift relation:

$$V_q^{(1)} A = V_q^{(2)} \tag{20}$$

The critical step is to realize that V_q can be computed from a square root factorization or eigenanalysis of D_q:

$$D_q = P_q P_q^T = \tilde{P}_q \tilde{S}_q \tilde{P}_q^T \tag{21}$$

To find the A matrix using Eq. (21), partition P_q into row blocks, each of dimension $n_y \times n_x$:

$$P_q = \begin{bmatrix} \tilde{P}_0 \\ \tilde{P}_1 \\ \vdots \\ \tilde{P}_{q-1} \end{bmatrix} \tag{22}$$

which means that the partitions of V_q can be associated with partitions of P_q:

$$\begin{aligned} V_q^{(1)} = P_q^{(1)} &= \begin{bmatrix} \tilde{P}_0 \\ \tilde{P}_1 \\ \vdots \\ \tilde{P}_{q-2} \end{bmatrix} \\ V_q^{(2)} = P_q^{(2)} &= \begin{bmatrix} \tilde{P}_1 \\ \tilde{P}_2 \\ \vdots \\ \tilde{P}_{q-1} \end{bmatrix} \end{aligned} \tag{23}$$

so that

$$A = \left[P_q^{(1)}\right]^+ P_q^{(2)} \tag{24}$$

This is incidentally the same realization found for the Hankel Q-Markov COVER,[3,6] which is known to asymptotically approach a realization that matches the first q number of Markov parameters and Covariance parameters. It can also be shown that the corresponding A is stable when very wide Hankel matrices (large d) are used. It is also equivalent to a special case of the ERA/DC algorithm[4] in which the data correlation matrix includes only a single Hankel block. As shown later, it is consequently a simple coordinate scaling of the ERA realization, when special values of q and d are used.

Relation to ERA

This section shows how the preceding realization (24) actually corresponds to the ERA realization (13). First, consider how the pseudo-inverse of $P_q^{(1)}$ is formed. Since $P_q^{(1)}$ is the upper partition of P_q, D_q can be partitioned as

$$\begin{aligned} D_q = P_q P_q^T &= \begin{bmatrix} P_q^{(1)} \\ \tilde{P}_q \end{bmatrix} \left\{ \left[P_q^{(1)}\right]^T \quad \tilde{P}_q^T \right\} \\ &= \begin{Bmatrix} P_q^{(1)}\left[P_q^{(1)}\right]^T & P_q^{(1)}\tilde{P}_q^T \\ \tilde{P}_q\left[P_q^{(1)}\right]^T & \tilde{P}_q\tilde{P}_q^T \end{Bmatrix} \end{aligned} \tag{25}$$

This means that $P_q^{(1)}$ is the square root factorization of the upper left $q-1 \times q-1$ partition of D_q, which is D_{q-1}. In terms of appropriate Hankel matrices, this means

$$P_q^{(1)}\left[P_q^{(1)}\right]^T = D_{q-1} = H_{q-1d}(0) H_{q-1d}(0)^T \tag{26}$$

Consequently, D_{q-1} can be factored instead of D_q to obtain $P_q^{(1)}$:

$$P_q^{(1)} = P_{q-1} = \tilde{P}_{q-1} \tilde{S}_{q-1}^{\frac{1}{2}} \tag{27}$$

Then, the pseudo-inverse of $P_q^{(1)}$ is

$$\left[P_q^{(1)}\right]^+ = \tilde{S}_{q-1}^{-\frac{1}{2}} \tilde{P}_{q-1}^T \tag{28}$$

This result is useful only insofar that $P_q^{(2)}$ can be formed without decomposing D_q. To see how this is done, recognize that the D_q matrix can also be partitioned as

$$\begin{aligned} D_q = P_q P_q^T &= \begin{bmatrix} \tilde{P}_0 \\ P_q^{(2)} \end{bmatrix} \left\{ \left[P_q^{(1)}\right]^T \quad \tilde{P}_q^T \right\} \\ &= \begin{Bmatrix} \tilde{P}_0\left[P_q^{(1)}\right]^T & \tilde{P}_0\tilde{P}_q^T \\ P_q^{(2)}\left[P_q^{(1)}\right]^T & P_q^{(2)}\tilde{P}_q^T \end{Bmatrix} \end{aligned} \tag{29}$$

Define the lower left partition of D_q to be

$$D_q^* = P_q^{(2)}\left[P_q^{(1)}\right]^T \tag{30}$$

Then, if this partition has been computed, then $P_q^{(2)}$ can be formed from $P_q^{(1)}$ by

$$P_q^{(2)} = D_q^*\left[P_q^{(1)}\right]^{+T} = D_q^* \tilde{P}_{q-1} \tilde{S}_{q-1}^{-\frac{1}{2}} \tag{31}$$

Substituting this result and Eq. (28) into Eq. (24), the A matrix is given by

$$\begin{aligned} A &= \left[P_q^{(1)}\right]^+ P_q^{(2)} \\ &= \tilde{S}_{q-1}^{-\frac{1}{2}} \tilde{P}_{q-1}^T D_q^* \tilde{P}_{q-1} \tilde{S}_{q-1}^{-\frac{1}{2}} \end{aligned} \tag{32}$$

This form is the actual computation used in the numerical performance tests described later, but one additional step is required to show how this relates to Eq. (13).

Since

$$D_q = H_{qd}(0) H_{qd}(0)^T \tag{33}$$

and since $H_{qd}(0)$ can be partitioned as

$$H_{qd}(0) = \begin{bmatrix} H_{q-1d}(0) \\ \tilde{H}_{qd} \end{bmatrix} = \begin{bmatrix} \tilde{H}_{1d} \\ H_{q-1d}(1) \end{bmatrix} \tag{34}$$

the data matrix D_q can be partitioned as follows:

$$\begin{aligned} D_q &= \begin{bmatrix} H_{q-1d}(0) H_{q-1d}(0)^T & H_{q-1d}(0)\tilde{H}_{qd}^T \\ \tilde{H}_{qd} H_{q-1d}(0)^T & \tilde{H}_{qd}\tilde{H}_{qd}^T \end{bmatrix} \\ &= \begin{bmatrix} \tilde{H}_{1d} H_{q-1d}(0)^T & \tilde{H}_{1d}(0)\tilde{H}_{qd}^T \\ H_{q-1d}(1) H_{q-1d}(0)^T & H_{q-1d}(1)\tilde{H}_{qd}^T \end{bmatrix} \end{aligned} \tag{35}$$

The upper left partition of the first result, when used with Eqs. (21) and (10), leads to the following:

$$\begin{aligned} \boldsymbol{D}_{q-1} &= \tilde{\boldsymbol{P}}_{q-1}\bar{\boldsymbol{S}}_{q-1}\tilde{\boldsymbol{P}}_{q-1}^T = \boldsymbol{H}_{q-1d}(0)\boldsymbol{H}_{q-1d}(0)^T \\ &= \tilde{\boldsymbol{P}}_{q-1}\boldsymbol{S}_{q-1d}\tilde{\boldsymbol{Q}}_d^T\tilde{\boldsymbol{Q}}_d\boldsymbol{S}_{q-1d}\tilde{\boldsymbol{P}}_{q-1}^T \\ &= \tilde{\boldsymbol{P}}_{q-1}\boldsymbol{S}_{q-1}^2\tilde{\boldsymbol{P}}_{q-1}^T \end{aligned} \tag{36}$$

which means that the ERA singular values $\boldsymbol{S}_{rs}$ are related to the FastERA singular values $\bar{\boldsymbol{S}}_{q-1}$ by

$$\boldsymbol{S}_{rs} = \bar{\boldsymbol{S}}_{q-1}^{\frac{1}{2}} \tag{37}$$

when the following values of r and s are used:

$$\begin{aligned} r &\rightarrow q-1 \\ s &\rightarrow d \end{aligned} \tag{38}$$

The lower left partition of the second result in Eq. (35), when used with Eqs. (32) and (37), leads to the following expression for the $\boldsymbol{A}$ matrix:

$$\begin{aligned} \boldsymbol{A} &= \bar{\boldsymbol{S}}_{q-1}^{-\frac{1}{2}}\tilde{\boldsymbol{P}}_{q-1}^T\boldsymbol{D}_q^*\tilde{\boldsymbol{P}}_{q-1}\bar{\boldsymbol{S}}_{q-1}^{-\frac{1}{2}} \\ &= \bar{\boldsymbol{S}}_{q-1}^{-\frac{1}{2}}\tilde{\boldsymbol{P}}_{q-1}^T\boldsymbol{H}_{q-1d}(1)\boldsymbol{H}_{q-1d}^T(0)\tilde{\boldsymbol{P}}_{q-1}\bar{\boldsymbol{S}}_{q-1}^{-\frac{1}{2}} \\ &= \bar{\boldsymbol{S}}_{q-1}^{-\frac{1}{2}}\tilde{\boldsymbol{P}}_{q-1}^T\boldsymbol{H}_{q-1d}(1)\tilde{\boldsymbol{Q}}_d\boldsymbol{S}_{q-1d}\tilde{\boldsymbol{P}}_{q-1}^T\tilde{\boldsymbol{P}}_{q-1}\bar{\boldsymbol{S}}_{q-1}^{-\frac{1}{2}} \\ &= \boldsymbol{S}_{q-1}^{-\frac{1}{2}}\boldsymbol{S}_{q-1}^{-\frac{1}{2}}\tilde{\boldsymbol{P}}_{q-1}^T\boldsymbol{H}_{q-1d}(1)\tilde{\boldsymbol{Q}}_d\boldsymbol{S}_{q-1}^{-\frac{1}{2}}\boldsymbol{S}_{q-1}^{\frac{1}{2}} \end{aligned} \tag{39}$$

Comparing this with Eq. (13) implies that the realizations from FastERA and ERA are identical to within a diagonal coordinate transformation, when appropriate values of r and s are used as indicated in Eq. (38).

Formation of the Hankel Matrix Product

One objection to the decomposition of the Hankel matrix product is that the computation required to form the product mitigates any advantage over the SVD in decomposing a smaller matrix. To see why this is not the case, note that the product can be written in terms of Markov parameters as

$$\begin{aligned} \boldsymbol{D}_q &= \boldsymbol{H}_{qd}(0)\boldsymbol{H}_{qd}(0)^T \\ &= \begin{bmatrix} \boldsymbol{M}(1) & \boldsymbol{M}(2) & \cdots & \boldsymbol{M}(d) \\ \boldsymbol{M}(2) & \boldsymbol{M}(3) & \cdots & \boldsymbol{M}(d+1) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{M}(q) & \boldsymbol{M}(q+1) & \cdots & \boldsymbol{M}(q+d-1) \end{bmatrix} \\ &\times \begin{bmatrix} \boldsymbol{M}(1)^T & \boldsymbol{M}(2)^T & \cdots & \boldsymbol{M}(q)^T \\ \boldsymbol{M}(2)^T & \boldsymbol{M}(3)^T & \cdots & \boldsymbol{M}(q+1)^T \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{M}(d)^T & \boldsymbol{M}(d+1)^T & \cdots & \boldsymbol{M}(q+d-1)^T \end{bmatrix} \end{aligned} \tag{40}$$

One would expect that this would require order q^2d multiplications and additions, which, when added to the time required to decompose $\boldsymbol{D}_q$, would equal or exceed the time required to decompose $\boldsymbol{H}_{qd}(0)$ in ERA. This is not the case because of the following recursion that can be used to form $\boldsymbol{D}_q$. If $\boldsymbol{D}_q$ is partitioned as follows,

$$\boldsymbol{D}_q = \begin{bmatrix} \bar{\boldsymbol{D}}_q(1,1) & \bar{\boldsymbol{D}}_q(1,2) & \cdots & \bar{\boldsymbol{D}}_q(1,q) \\ \bar{\boldsymbol{D}}_q(1,2)^T & \bar{\boldsymbol{D}}_q(2,2) & \cdots & \bar{\boldsymbol{D}}_q(1,q) \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\boldsymbol{D}}_q(1,q)^T & \bar{\boldsymbol{D}}_q(2,q)^T & \cdots & \bar{\boldsymbol{D}}_q(1,q) \end{bmatrix} \tag{41}$$

then the $n_y \times n_y(i,\ j)$ partition of $\boldsymbol{D}_q$ is

$$\bar{\boldsymbol{D}}_q(i,j) = \sum_{k=i}^{d+i-1} \boldsymbol{M}(k)\boldsymbol{M}(k+j-i)^T \tag{42}$$

Notice that the $(i+1,\ j+1)$ partition of $\boldsymbol{D}_q$ is

$$\begin{aligned} \bar{\boldsymbol{D}}_q(i+1,\ j+1) &= \sum_{k=i+1}^{d+i} \boldsymbol{M}(k)\boldsymbol{M}(k+j-i)^T \\ &= \sum_{k=i}^{d+i-1} \boldsymbol{M}(k)\boldsymbol{M}(k+j-i)^T \\ &+ \boldsymbol{M}(d+i)\boldsymbol{M}(d+j)^T - \boldsymbol{M}(i)\boldsymbol{M}(j)^T \\ &= \bar{\boldsymbol{D}}_q(i,j) + \boldsymbol{M}(d+i)\boldsymbol{M}(d+j)^T - \boldsymbol{M}(i)\boldsymbol{M}(j)^T \end{aligned} \tag{43}$$

This recursion can be used to generate subsequent rows of $\boldsymbol{D}_q$ if the first row has been computed from Eq. (40). This recursion was also pointed out by Ref. 16.

Transpose Formulation of FastERA

Decomposition of the inner product of the Hankel matrix is appropriate only when qn_y is less than dn_u. Most often, however, the number of measurements far exceeds the number of inputs. In this case, forming the product

$$\boldsymbol{D}_q = \boldsymbol{H}_{qd}(0)\boldsymbol{H}_{qd}(0)^T \tag{44}$$

to obtain the factors of the Hankel matrix is less efficient than forming

$$\hat{\boldsymbol{D}}_q = \boldsymbol{H}_{qd}(0)^T\boldsymbol{H}_{qd}(0) \tag{45}$$

Many of the singular values computed from $\boldsymbol{D}_q$ are zero (and need not be computed) when qn_y is more than dn_u. However, in this case $\hat{\boldsymbol{D}}_q$ has no zero singular values if the Hankel matrix is full column rank, and its spectral decomposition is therefore more efficient than that of $\boldsymbol{D}_q$.

A dual to the preceding algorithm can be formulated using $\hat{\boldsymbol{D}}_q$ in place of $\boldsymbol{D}_q$. Define the transpose Markov parameters to be

$$\hat{\boldsymbol{M}}(k) = \boldsymbol{M}(k)^T \tag{46}$$

Suppose we find a realization of this transposed data sequence:

$$\begin{aligned} \hat{\boldsymbol{x}}(k+1) &= \hat{\boldsymbol{A}}\hat{\boldsymbol{x}}(k) + \hat{\boldsymbol{B}}\hat{\boldsymbol{u}}(k) \\ \hat{\boldsymbol{y}}(k) &= \hat{\boldsymbol{C}}\hat{\boldsymbol{x}}(k) + \hat{\boldsymbol{D}}\hat{\boldsymbol{u}}(k) \end{aligned} \tag{47}$$

The transpose Markov parameters can be related to the state-space matrices by

$$\hat{\boldsymbol{M}}(k) = \begin{cases} \hat{\boldsymbol{D}}, & k=0 \\ \hat{\boldsymbol{C}}\hat{\boldsymbol{A}}^{k-1}\hat{\boldsymbol{B}}, & k>0 \end{cases} \tag{48}$$

Using Eqs. (3) and (46), this is the same as

$$\boldsymbol{M}(k)^T = \begin{cases} \boldsymbol{D}^T, & k=0 \\ \boldsymbol{B}^T[\boldsymbol{A}^{k-1}]^T\boldsymbol{C}^T, & k>0 \end{cases} \tag{49}$$

This means that the two realizations are related by

$$\begin{aligned} \boldsymbol{A} &= \hat{\boldsymbol{A}}^T & \boldsymbol{B} &= \hat{\boldsymbol{C}}^T \\ \boldsymbol{C} &= \hat{\boldsymbol{B}}^T & \boldsymbol{D} &= \hat{\boldsymbol{D}}^T \end{aligned} \tag{50}$$

FastERA Computational Procedure

To summarize what is here meant by the FastERA procedure, the following steps are itemized:

1) For given ERA values of r and s, set

$$q = r+1$$

$$d = s$$

2) Compute the first row of the $\boldsymbol{D}_q$ matrix using

$$\bar{\boldsymbol{D}}_q(1, j) = \sum_{k=1}^{d} \boldsymbol{M}(k)\boldsymbol{M}(k + j - 1)^T$$

3) Form the remainder of the upper triangular part of the $\boldsymbol{D}_q$ matrix using

$$\bar{\boldsymbol{D}}_q(1, j) = \bar{\boldsymbol{D}}_q(i - 1, j - 1) + \boldsymbol{M}(d + i - 1)$$

$$\times \boldsymbol{M}(d + j - 1)^T - \boldsymbol{M}(i - 1)\boldsymbol{M}(j - 1)^T$$

The remainder of $\boldsymbol{D}_q$ is known from symmetry.

4) Determine the n_x largest eigenvalues $\boldsymbol{S}_{q-1}$ of the upper left $q - 1 \times q - 1$ partition of $\boldsymbol{D}_q$:

$$\boldsymbol{D}_{q-1} = \tilde{\boldsymbol{P}}_{q-1}\bar{\boldsymbol{S}}_{q-1}\tilde{\boldsymbol{P}}_{q-1}^T$$

5) Form the pseudo-inverse of $\boldsymbol{P}_q^{(1)}$:

$$\left[\boldsymbol{P}_q^{(1)}\right]^T = \bar{\boldsymbol{S}}_{q-1}^{-\frac{1}{2}}\tilde{\boldsymbol{P}}_{q-1}^T$$

6) Form $\boldsymbol{P}_q^{(2)}$:

$$\boldsymbol{P}_q^{(2)} = \boldsymbol{D}_q^*\left[\boldsymbol{P}_q^{(1)}\right]^{+T}$$

7) Form $\boldsymbol{A}$:

$$\boldsymbol{A} = \left[\boldsymbol{P}_q^{(1)}\right]^{+}\boldsymbol{P}_q^{(2)}$$

8) Form $\boldsymbol{B}$:

$$\boldsymbol{B} = \left[\boldsymbol{P}_q^{(1)}\right]^{+}\begin{bmatrix}\boldsymbol{M}(1)\\ \vdots \\ \boldsymbol{M}(q)\end{bmatrix}$$

9) Form $\boldsymbol{C}$:

$$\boldsymbol{C} = \boldsymbol{P}_q^{(1)}(1{:}n_y, 1{:}n_x)$$

$$= \left[\boldsymbol{P}_q^{(1)}\right]^{+T}\bar{\boldsymbol{S}}_{q-1}$$

10) Form $\boldsymbol{D}$:

$$\boldsymbol{D} = \boldsymbol{M}(0)$$

11) If $q\,n_y > d\,n_u$, the following steps are followed instead of steps 1–10:

Form the transpose Markov parameters from the ordinary Markov parameters:

$$\hat{\boldsymbol{M}}(k) = \boldsymbol{M}(k)^T$$

Perform steps 2–10 with

$$\boldsymbol{M}(k) = \hat{\boldsymbol{M}}(k)$$

$$q = s + 1$$

$$d = r$$

Transpose the state-space model:

$$\boldsymbol{A} = \hat{\boldsymbol{A}}^T \qquad \boldsymbol{B} = \hat{\boldsymbol{C}}^T$$

$$\boldsymbol{C} = \hat{\boldsymbol{B}}^T \qquad \boldsymbol{D} = \hat{\boldsymbol{D}}^T$$

Computational Implementation

This section discusses several key issues in the successful implementation of the FastERA algorithm. The first section discusses numerical roundoff effects in the squaring of the Hankel matrix, the second section discusses the partial symmetric eigendecomposition, and the third section discusses memory efficiency issues.

Numerical Precision Effects

A major numerical concern with the decomposition of $\boldsymbol{D}_q$ instead of $\boldsymbol{H}_{qd}(0)$ is that the process of forming $\boldsymbol{D}_q$ from multiplying the Hankel matrices reduces its condition number. To see this, suppose that one of the Hankel matrix singular values is perturbed by a small amount, so that the numerically computed singular values are

$$S_i \longrightarrow S_i + \varepsilon \tag{51}$$

where ε is the numerical precision of the machine used in the computation. If these are squared, then

$$\bar{S}_i \approx S_i^2 \longrightarrow S_i^2 + 2\varepsilon S_i + \varepsilon^2 \tag{52}$$

in which case the relative roundoff error is $2(\varepsilon/S_i) + (\varepsilon^2/S_i^2)$. In contrast, if the numerically computed singular values of the $\boldsymbol{D}_q$ matrix are perturbed by machine precision, then

$$\bar{S}_i \longrightarrow S_i^2 + \varepsilon \tag{53}$$

so that the relative roundoff error is ε/S_i^2. This implies that the relative roundoff error in the computed singular values of the $\boldsymbol{D}_q$ matrix becomes significantly degraded in comparison with the corresponding singular values of the Hankel matrix when the magnitude of the singular value becomes comparable to machine precision.

This means that numerical roundoff effects only the smallest singular values. However, in most modal data analysis, only the largest singular values are retained. Therefore, numerical roundoff in the formation of $\boldsymbol{D}_q$ should not be an issue in modal analysis, unless it increases the number of iterations required in the numerical solution. This will be evaluated by application to experimental test data later. For a more complete discussion of the effect of squaring on the smallest singular values, see Ref. 12.

The potential for roundoff effects leads to a decision to use double precision (32 bit) accuracy in the implementation of both the ERA and the FastERA algorithms for the comparison study. Whether this effects computational efficiency depends on the computer. For instance, double precision executes half as fast as single precision on a SUN Sparc 2 but actually executes faster than single precision on an IBM RS/6000.

Partial Singular Value Decomposition

As mentioned in the introduction, the use of $\boldsymbol{D}_q$ to obtain the Hankel matrix spectral decomposition makes it possible to consider computing only a partial number of the singular values. For instance, if one is interested in only the strongest 100 modes of the structure, then no more that 200 singular values would ever be retained, even if the dimension of $\boldsymbol{D}_q$ is as high as several thousand. It is not numerically necessary to find all of the singular values of $\boldsymbol{D}_q$ to compute the largest subspace of $\boldsymbol{D}_q$.

The ordinary singular value decomposition (SVD) algorithm,[7] which would be used to compute the SVD of the Hankel matrix or $\boldsymbol{D}_q$, is a variant of the QR algorithm.[13] It requires a computation of all of the singular values before the singular vectors can be computed. Since $\boldsymbol{D}_q$ is symmetric, an eigensolver can be used in place of an SVD solver, because the eigenvalues and vectors of $\boldsymbol{D}_q$ are the same as its singular values. Several algorithms available in general linear algebra software packages[7,9] can solve selected eigenvalues and vectors of symmetric matrices. Two methods are generally used to specify the desired eigenvalues: either by an index range or by a numerical range. A bisection algorithm is then used to find the eigenvalues, and an inverse iteration algorithm is typically used to find the corresponding eigenvectors.[7,9,14] In the studies presented next, the index range specification is used. It should be noted, however, that users may find the numerical range option useful in problems where the noise singular value level is known a priori.

Table 1 Comparison of singular value decomposition times (s) for typical Hankel matrix sizes

Size $H_{qd}(0)$	EISPACK	LAPACK	Speed ratio
144 × 600	7.3	3.8	1.9
304 × 600	62.9	27.0	2.3
464 × 600	182.1	97.0	1.9
624 × 600	346.9	241.6	1.4
784 × 600	451.8	263.1	1.72

Partial eigensolution should not be used indiscriminately; it is not always faster than finding the full decomposition. The full SVD algorithm will compute its decomposition in the same time that as many as one third of the eigenvalues can be calculated using a partial eigensolver. This means that the full SVD should be used whenever the fraction of the number of computed eigenvalues exceeds approximately one-third. This will also be evaluated next in the application to experimental data.

Efficient Use of Computer Memory

A final consideration is the efficient use of computer memory. This affects performance in two ways. First, the amount of process memory will generally constrain the practical size of the $\boldsymbol{D}_q$ matrix or $\boldsymbol{H}_{qd}(0)$ that can be decomposed on a given computer. Second, most modern computer architectures have several levels of memory, each successively faster but smaller in capacity. Consequently, the algorithm execution is improved by structuring the computations to keep as much necessary data in fast memory as long as possible.

To minimize the amount of memory required, the Fortran program developed for this study used the SUPES library[15] to dynamically allocate and de-allocate data arrays. This library uses C routines to make system-dependent memory allocation calls and compresses memory as necessary to optimize the total space available. In addition, the $\boldsymbol{D}_q$ matrix is stored as a packed symmetric array of $n(n+1)/2$ instead of n elements.

To maximize the efficient availability of data in core memory on modern RISC and parallel architectures, the recently developed LAPACK[9] replacement for EISPACK[7] and LINPACK[10] was used for both the ERA and FastERA implementations. This library is based on the BLAS 3[8] matrix-matrix multiplication routines. The EISPACK and LINPACK routines were based on the BLAS 1 vector-vector multiplication routines, which do not optimize memory usage on modern architectures. To show this effect, Table 1 presents the decomposition times for a range of Hankel matrix sizes for both EISPACK and LAPACK implementations of algebraically identical algorithms. These results were measured on an IBM RS/6000 model 550, for which the BLAS routines have been coded in assembly language for maximum efficiency on the given machine. To note the effect of using the assembly coded BLAS instead of compiled BLAS, similar decompositions on an RS/6000 were measured to be approximately 16 times faster than on a SUN Sparc 2, which uses compiled code, even though the benchmark performance of the RS/6000 is only about 4 times faster than the SUN Sparc 2. Clearly, using assembly coded and memory optimized BLAS routines greatly affects computational speed.

Application to Experimental Data

This section compares the computational efficiency of the algorithms presented earlier by applying them to experimental modal test data. The use of experimental data instead of simulated data for this comparison is critical to the comparison. The SVD and eigensolution algorithms are iterative, and their convergence depends on the numerical conditioning of the data. Because this is difficult to simulate, timing and accuracy comparisons that use simulated data are inherently suspect.

The first section describes the experimental test structure and how the data was collected, and the last three sections present and compare the timing, memory usage, and numerical precision of the algorithms.

Table 2 Computation time breakdown (s) for a typical data analysis case (2016 Hankel columns, 1000 Hankel rows, 30 states)

Task	ERA	FastERA/full SVD	FastERA/partial SVD
Data matrix formation	1.02	3.03	3.05
Data matrix decomposition	809.35	312.41	48.53
ABCD formation	2.29	1.16	0.99
Error analysis	0.45	2.68	2.69
Total	813.11	319.28	55.26

Experimental Test Structure and Test Procedure

Data were collected from a 1:10 dynamic scale model of an orbiting interferometric imaging telescope. This structure has a triangular base, for which each leg consists of 11 bays constructed from a 1-ft-long pyramidal truss network. A fourth leg extends vertically to above the center of the triangular base to form an image collecting plane. The structure was suspended at three points by nylon wires to simulate a free-free condition. The bounce mode of the suspended structure was approximately 1 Hz, and the first bending mode was 10 Hz. The damping levels observed in this structure were typically 0.5% critical or less. This structure had approximately 200–300 modes below 300 Hz.

Pseudorandom burst force inputs were applied using a 50-lb modal shaker in three orthogonal directions from each corner of the structure. The force load path passed from the shaker to a moment isolation stinger that was fixed to a force transducer. Vibration measurements were measured at 31 locations by piezoelectric accelerometers mounted to the nodes of the structure. The accelerometer signals were filtered at 300 Hz and amplified by a programmable signal conditioner. The amplification gain factor was set to use the entire dynamic range of each sensor for a given measurement. The signals were simultaneously digitized at a rate of 1000 Hz by a 32-channel 12-bit-analog-to digital converter. A total of 93 frequency response functions were measured, averaged over 300 repetitions, and then converted into Markov parameter estimates for subsequent analysis. In each ERA or FastERA analysis, the desired model order was fixed at 30 states.

Timing Comparisons

Table 2 and Figs. 1–3 plot the total required CPU times as a function of varying row or column dimension. These times are the total CPU time for a complete analysis, including 1) the system identification algorithm, 2) the solution for the eigenmodes of the state-space model, and 3) the generation of modal amplitude coherences[2] from the data and the model. This was necessary because although the FastERA realization step might be significantly faster, this is only meaningful in the context of the total analysis time.

Table 2 shows a typical time breakdown for ERA, FastERA with a full SVD, and FastERA with a partial SVD. ERA forms its data

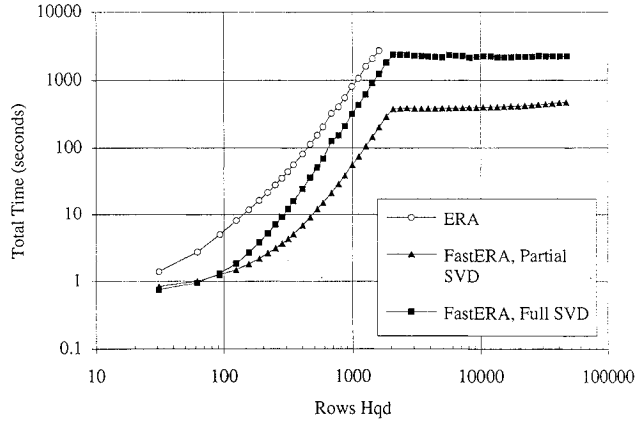


Fig. 1 Variation of total computational time with number of Hankel matrix rows with the number of columns fixed at 2016. ERA exhausts memory for large rows sizes.

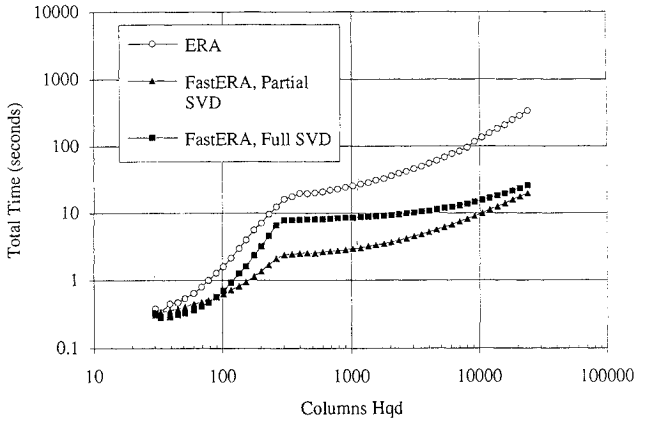


Fig. 2 Variation of total computation time with number of Hankel matrix columns with the number of rows fixed at 2015. The FastERA/partial SVD variant is approximately 16 times faster than ERA at large matrix sizes.

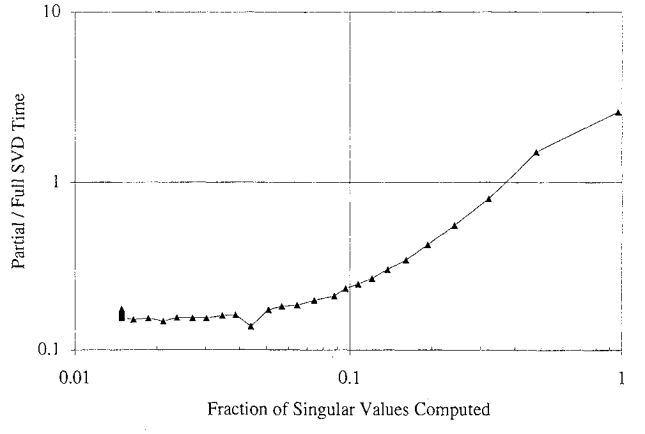


Fig. 3 Partial SVD computational time as a fraction of full SVD computational time. When more than approximately 35% of the singular values are computed, the full SVD is more efficient.

matrix (the Hankel matrix) very quickly by simply copying Markov parameters to memory. Note, however, that the corresponding $\boldsymbol{D}_q$ matrix is formed in just a few more seconds by FastERA. The greatest difference is seen in the decomposition times. The ERA solution required 2.6 times the full SVD solution for FastERA and 16 times the partial SVD solution for FastERA. The matrix formations are also faster for the FastERA algorithms because the matrix multiplication $\boldsymbol{H}_{qd}(1)\tilde{\boldsymbol{Q}}_d^T$ in ERA is implicit in FastERA. The modal error analysis, however, requires significantly more time in the FastERA algorithms. This is because the formation of the modal amplitude coherence and extended modal amplitude coherence values requires the $\tilde{\boldsymbol{Q}}_d^T$ singular vectors. Whereas ERA finds this as part of the decomposition, FastERA requires additional time to solve

$$\tilde{\boldsymbol{Q}}_d^T = \tilde{\boldsymbol{S}}_q^{-\frac{1}{2}} \tilde{\boldsymbol{P}}_q^T \boldsymbol{H}_{qd}(0) \tag{54}$$

Clearly, though, for the single example in Table 2, FastERA with a partial SVD has a significant computational advantage over ERA.

Figures 1 and 2 show how the total CPU time varies with rows and columns of the Hankel matrix. The example from Table 2 is approximately in the center of the range of each plot. Figure 1 varies the rows of $\boldsymbol{H}_{qd}(0)$ with the number of columns fixed at 2016, and Fig. 2 varies the columns of $\boldsymbol{H}_{qd}(0)$ with the number of rows fixed at 2015. Note that these plots use logarithmic scales.

Figure 1 shows that the ERA solution could not extend beyond approximately 1500 rows. This is because the memory requirements of the algorithm exceeded the available memory on the machine used for these comparisons. Above this Hankel matrix size, the FastERA algorithms process data sets larger than what is practical for ERA. This memory limit was not reached for the variation with column size shown in Fig. 2.

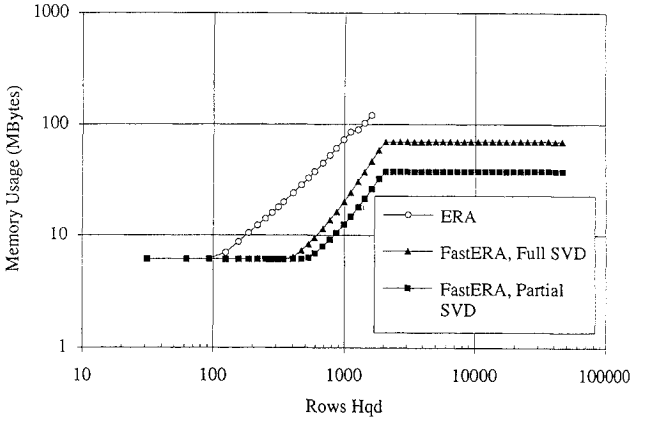


Fig. 4 Variation of computer memory usage with number of Hankel matrix rows with the number of columns fixed at 2016. The FastERA/partial SVD variant has a clear memory advantage over the other options.

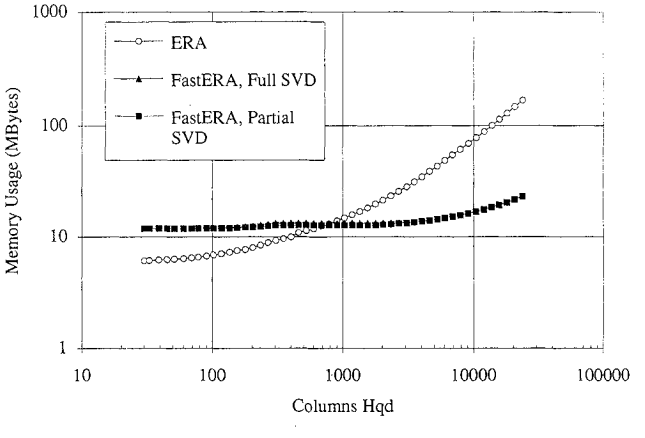


Fig. 5 Variation of computer memory usage with number of Hankel matrix columns with the number of rows fixed at 2015.

Notice that the slope of the CPU trend is reduced when the Hankel matrix size changes from tall to wide in Fig. 1 or from wide to tall in Fig. 2. For the ERA algorithm, this change is due to the recognition of the SVD algorithm that the null space of the Hankel matrix need not be computed. For the FastERA variants, this is due to a switch in the algorithm from decomposition of $\boldsymbol{D}_q$ to $\hat{\boldsymbol{D}}_q$, which is formed from the outer product of the Hankel matrix rather than its inner product. (See the last step in the FastERA procedure presented earlier.)

The third important characteristic revealed in these plots is the relative efficiency of the full and partial SVD solutions. For most cases, the partial SVD, which found only 30 singular vectors, executed faster. At high Hankel matrix sizes, this speed advantage is reduced when compared with the total execution time, which becomes dominated by the error analysis computations. Notice also that at small matrix sizes, where the row or column rank of the Hankel matrix becomes less than 100, the partial SVD actually begins to require more CPU time than the full SVD. This expected result is further detailed in Fig. 3, which shows the execution time ratio for the partial vs full SVD as a function of the fraction of the number of singular values of $\boldsymbol{D}_q$ that are computed. As expected in the previous section, it is more efficient to use the full SVD algorithm when more than approximately 35% of the singular values are to be computed.

Memory Usage Comparison

Figures 4 and 5 show that the partial SVD variant of FastERA uses the least amount of memory of the three options. Figure 4 shows that ERA exhausts machine memory at high matrix sizes, but FastERA will use a constant amount of memory at larger Hankel matrix row size. This is due to the constant size of the $\boldsymbol{D}_q$ matrix using the transpose dual formation. Both FastERA variants can accommodate and process much larger data sets than ERA.

Numerical Precision Effects

To determine the effect, if any, of the numerical procedure on the accuracy of the poles of the realized model, the continuous modal eigenvalues found by FastERA were compared with the corresponding eigenvalues found by ERA. The imaginary parts were found to be different by a randomly distributed amount about zero with a magnitude of approximately of 10^{-16}, which is approximately the machine precision. However, all of the real (damping) components were different by a positive amount of approximately 10^{-12}, meaning that the FastERA damping values are biased towards higher damping when compared with the ERA eigenvalues. For this example data set, however, these numerical biases are relatively small, so the gain in computational efficiency is with an insignificant sacrifice in precision.

Conclusions

This paper investigated the computational advantages of an algebraically equivalent form of the eigensystem realization algorithm. Using a partial eigensolution of the symmetric inner or outer product of the Hankel matrix with itself, it was shown that significant computational and memory advantages are realized for experimental vibration data at negligible loss in numerical precision. This partial eigensolution approach makes practical the determination of modal parameters for complex modal data sets with hundreds of sensors and hundreds of modal resonances.

Further studies should consider additional improvements to the decomposition algorithm. In particular, algorithms based on a Lanczos or similar iterative sweeping procedure can further reduce the amount of storage required for the computation. It may also be possible to further enhance the performance by using disk-based rather than core-based algorithms or algorithms that can operate directly from data. Continuing research is also examining the behavior of the realization algorithms at extreme Hankel matrix sizes, and these will be reported in a future paper.

Acknowledgments

This research was supported in part by National Science Foundation Grant MSS-9007276, by a donation from Hughes Aircraft Company, Space and Communications Group, and by a donation from the Shimizu Corporation. This paper is based on a paper that appeared in the *Proceedings of the 10th International Modal Analysis Conference*, San Diego, California, February 1992.

References

[1]Juang, J.-N., and Pappa, R. S., "An Eigensystem Realization Algorithm (ERA) for Modal Parameter Identification and Model Reduction," *Journal of Guidance, Control, and Dynamics*, Vol. 8, No. 5, 1985, pp. 620–627.

[2]Juang, J.-N., and Pappa, R. S., "Effects of Noise on Modal Parameters Identified by the Eigensystem Realization Algorithm," *Journal of Guidance, Control, and Dynamics*, Vol. 9, No. 3, 1986, pp. 294–303.

[3]Peterson, L. D., and Bullock, S. J., "A Comparison of the Eigensystem Realization Algorithm and the Q-Markov COVER," *Proceedings of the 32nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference* (Baltimore, MD), AIAA, Washington, DC, 1991, pp. 2169–2178.

[4]Juang, J.-N., Cooper, J. E., and Wright, J. R., "An Eigensystem Realization Algorithm Using Data Correlations (ERA/DC) for Modal Parameter Identification," *Control Theory and Advanced Technology*, Vol. 4, No. 1, 1988, pp. 5–14.

[5]Bendat, J. S., and Piersol, A.G., *Random Data: Analysis and Measurement Procedures*, 2nd ed., Wiley, New York, 1986.

[6]King, A. M., Desai, U. B., and Skelton, R. E., "A Generalized Approach to q-Markov Covariance Equivalent Realizations for Discrete Systems," *Automatica*, Vol. 24, No. 4, 1988, pp. 507–515.

[7]Smith, B. T., Boyle, J. M., Dongarra, J. J., Garbow, B. S., Ikebe, Y., Klema, V. C., and Moler, C. B., *Matrix Eigensystem Routines—Eispack Guide*, Springer-Verlag, New York, 1976.

[8]Dongarra, J., Du Croz, S., Duff, I., and Hammerling, S., "A Set of Level 3 Basic Linear Algebra Subprograms," *ACM Transactions on Mathematical Software*, Vol. 16, No. 1, 1990, pp. 1–17.

[9]Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S., and Sorenson, D., *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[10]Dongarra, J. J., Bunch, J. R., Moler, C. B., and Stewart, G. W., *LINPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1979.

[11]Juang, J. N., Phan, M., Horta, L. G., and Longman, R. W., "Identification of Observer/Kalman Filter Markov Parameters: Theory and Experiments," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, AIAA, Washington, DC, 1991, pp. 1195–1207.

[12]Cullum, J., Willoughby, R. A., and Lake, M., "A Lanczos Algorithm for Computing Singular Values and Vectors of Large Matrices," IBM, Research Rept. RC 9166, Yorktown Heights, NY, 1981.

[13]Golub, G. H., and Reinsch, C., "Singular Value Decomposition and Least Squares Solutions," *Numerical Math.*, Vol. 14, 1970, pp. 403–420.

[14]Barth, W., Martin, R. S., and Wilkinson, J. H., "Calculation of the Eigenvalues of a Symmetric Tridiagonal Matrix by the Method of Bisection," *Numerical Math.*, Vol. 9, 1967, pp. 386–393.

[15]Red-Horse, J. R., Mills-Curran, W. C., and Flanagan, D. P., "SUPES Version 2.1: A Software Utilities Package for the Engineering Sciences," Sandia National Laboratories, Rept. SAND90-0247, Albuquerque, NM, May 1990.

[16]Russell, W. C., Ikegami, R., Hunziker, K. S., and Campbell, A. C., "Recursive System Identification of Space Structures," *Proceedings of the USAF/NASA System Identification and Health Monitoring of Precision Space Structures Conference* (Pasadena, CA), March 1990, pp. 245–264.