JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 19, No. 4, July–August 1996

Efficient Computation of Closed-Loop Frequency Response for Large-Order Flexible Systems

Peiman G. Maghami
NASA Langley Research Center, Hampton, Virginia 23681-0001
and
Daniel P. Giesy
Lockheed Martin Engineering and Sciences, Hampton, Virginia 23666

An efficient and robust computational scheme is given for the calculation of the frequency response function of a large-order, flexible system implemented with a linear, time-invariant control system. Advantage is taken of the highly structured sparsity of the system matrix of the plant based on a model of the structure using normal mode coordinates. The computational time per frequency point of the new computational scheme is a linear function of system size, a significant improvement over traditional, full-matrix techniques whose computational times per frequency point range from quadratic to cubic functions of system size. This permits the practical frequency-domain analysis of systems of much larger order than by traditional, full-matrix techniques. Formulations are given for both open- and closed-loop systems. Numerical examples are presented showing the advantages of the present formulation over traditional approaches, both in speed and in accuracy. Using a model with 703 structural modes, a speed-up of almost two orders of magnitude was observed, while accuracy improved by up to 5 decimal places.

Introduction

CONTROL of flexible systems has received significant attention in the literature. To date, numerous techniques, algorithms, and procedures have been developed for design of controllers for such systems ranging from spacecraft and satellites to aircraft, ships, machines, etc. These flexible systems, which are generally infinite-dimensional, are typically modeled using a finite number of generalized coordinates or modes. Control of flexible systems may become difficult, depending on the number, location, relative proximity, and inherent damping of these modes. The response of the system to a given disturbance/excitation generally depends on modal properties (amplitude, frequency, and damping) and the amplitude and phase content of the disturbance/excitation. In general, two techniques, time-domain analysis and frequency-domain analysis, have been developed and extensively used to analyze and characterize the input/output behavior of linear time-invariant systems including flexible systems. In frequency-domain analysis, frequency response functions (defined as transfer function matrices from inputs to outputs of the system) have typically been used (usually in the form of magnitude and phase or Bode plots) in the analysis of linear systems, as well as in designing controllers for such systems. In general, frequency response functions of the open-loop system are used to evaluate the performance of the open-loop system, and to identify and quantify needed performance and/or stability improvements at various frequency bands. The closed-loop frequency response functions are, typically, needed to ensure that desired performance and stability have been achieved by the control system. Moreover, frequency-domain specifications such as peak magnitude, bandwidth, roll-off rate, etc., are often used in characterizing the desired behavior of the system in the frequency domain (this is known as loop shaping).

In general, the order of the flexible system (as defined by the number of modes retained in the model) for which open-loop and/or closed-loop analysis is performed depends on the application considered. For example, if the closed-loop response of a spacecraft

Peiman G. Maghami received his D.Sc. degree from the George Washington University, Washington, DC, in 1984. For the period 1984–1985, he was a research assistant professor at the George Washington University. During 1985–1987, he served as a National Research Council research associate at NASA Langley Research Center, Hampton, VA. He served as a research assistant professor at Old Dominion University, Norfolk, VA, from 1987 to 1989. Since 1989, he has been with NASA Langley Research Center, working in the Guidance and Control Branch as a senior research engineer. His research areas of interest include control of flexible space structure, control of multibody systems, multidisciplinary design and optimization, and stochastic system theory and analysis.

Daniel Giesy received his B.A. and M.A. from Ohio State University in 1960 and his Ph.D. in 1964 from the University of Wisconsin–Madison, majoring in mathematics. He then taught mathematics at the University of Southern California, Western Michigan University, and Norfolk State University. Since 1977 he has provided technical support services under contract to NASA Langley Research Center, where he is now associated with the Guidance and Control Branch. He has designed algorithms and written software for control system design and analysis, specializing in optimization and numerical linear algebra.

Received Feb. 20, 1995; revision received Feb. 20, 1996; accepted for publication March 7, 1996.

with a low-bandwidth attitude control system is of interest, then a small set of modes would be sufficient to capture the low-frequency closed-loop behavior of the system. On the other hand, if the response of the flexible system is desired over a large frequency range or if the control system considered has a high bandwidth, then a large set of modes (in the hundreds or thousands) may be necessary to capture the true response of the system.

However, the current techniques for obtaining frequency response functions, although able to deal with small- or medium-size systems, have problems in handling large-order systems. A straightforward calculation of the frequency response function matrix at a single frequency point that is based on the definition of the transfer function has a computational cost that is a cubic function of the system size. If this calculation must be repeated for many frequency points, Laub[1,2] presents a technique that has a better average cost. This technique performs an initial orthogonal transformation of the system that reduces the system response matrix to Hessenberg form. This initial transformation has a computational cost that is a cubic function of the system size. This technique can then calculate the frequency response function matrix at each frequency point at a cost that is a quadratic function of the system size. For very large systems (many hundreds of modes or more), however, even this is too slow, and a better method is needed.

To this end, this paper describes a novel and efficient technique for the computation of closed-loop frequency response functions of large-order flexible systems. The proposed technique is computationally robust and accurate. It takes advantage of the sparsity of the flexible systems in normal mode coordinates and reduces the computational cost from a quadratic function of the order of the system to a linear function. Formulations are given for both open- and closed-loop systems. Numerical examples are presented showing the advantages of the present formulation over traditional approaches, both in speed and in accuracy.

Mathematical Formulation

Second-Order Modal Equations

The dynamics of a typical linear, time-invariant flexible system may be written in a second-order form as

$$M\ddot{x} + D\dot{x} + Kx = Hu + H_d w$$

$$y = C_p x + C_r \dot{x}$$

$$y^{pr} = C_p^{pr} x + C_r^{pr} \dot{x} + C_a^{pr} \ddot{x}$$

where M, D, and K are the $n \times n$ mass, damping, and stiffness matrices, respectively; x is the $n \times 1$ position/attitude vector; u is the $m \times 1$ control input vector; w is the $r \times 1$ disturbance vector; H is the $n \times m$ control input influence matrix; and H_d is the $n \times r$ disturbance influence matrix. The vectors y and y^{pr} are, respectively, the $q \times 1$ measurements output vector and the $l \times 1$ vector of performance outputs; C_p and C_r are $q \times n$ measurement output influence matrices; and C_p^{pr}, C_r^{pr}, and C_a^{pr} are $l \times n$ performance output influence matrices.

If the second-order system is transformed into normal mode coordinates, and p of the normal modes are retained to capture the relevant dynamics of the structure, then the system equations may be written in a modal form as

$$\bar{M}\ddot{q} + \bar{D}\dot{q} + \bar{K}q = \bar{H}u + \bar{H}_d w$$

$$y = \bar{C}_p q + \bar{C}_r \dot{q}$$

$$y^{pr} = \bar{C}_p^{pr} q + \bar{C}_r^{pr} \dot{q} + \bar{C}_a^{pr} \ddot{q}$$

where $\bar{M}$, $\bar{D}$, and $\bar{K}$ are the $p \times p$ modal mass, damping, and stiffness matrices, respectively; q is the $p \times 1$ vector of modal coordinates; and $\bar{H}$ and $\bar{H}_d$ are the $p \times m$ control input and the $p \times r$ disturbance influence matrices in modal coordinates, respectively. The matrices $\bar{C}_p$ and $\bar{C}_r$ are $q \times p$ measurement output influence matrices in modal coordinates; and $\bar{C}_p^{pr}$, $\bar{C}_r^{pr}$, and $\bar{C}_a^{pr}$ are $l \times p$ performance output influence matrices in modal coordinates.

It is assumed that the mode shapes are normalized with respect to the mass matrix, and modal damping is assumed. This means that $\bar{M} = I_p$, $\bar{D} = \text{diag}\{2\zeta_1\omega_1, 2\zeta_2\omega_2, \ldots, 2\zeta_p\omega_p\}$, and $\bar{K} = \text{diag}\{\omega_1^2, \omega_2^2, \ldots, \omega_p^2\}$, where I_p is the identity matrix of order p; and ω_i and ζ_i are the open-loop frequencies and damping ratios.

The control input and disturbance influence matrices are given by

$$\bar{H} = \Phi^T H, \qquad \bar{H}_d = \Phi^T H_d$$

The measurement and performance output influence matrices are given by

$$\bar{C}_p = C_p \Phi, \qquad \bar{C}_r = C_r \Phi$$

$$\bar{C}_p^{pr} = C_p^{pr} \Phi, \qquad \bar{C}_r^{pr} = C_r^{pr} \Phi, \qquad \bar{C}_a^{pr} = C_a^{pr} \Phi$$

The columns of matrix Φ are the p retained mode shapes

$$\Phi = [\phi_1 \quad \phi_2 \quad \cdots \quad \phi_p]$$

The second-order modal equations may be rewritten in a first-order form as

$$\dot{x}_s = A_s x_s + B_s u + B_d w$$

$$y = C x_s \tag{1}$$

$$y^{pr} = C_1^{pr} x_s + C_2^{pr} \dot{x}_s$$

The vector x_s is the plant state vector whose components are

$$x_s = \begin{Bmatrix} q_1 \\ \dot{q}_1 \\ q_2 \\ \dot{q}_2 \\ \vdots \\ \vdots \\ q_p \\ \dot{q}_p \end{Bmatrix}$$

and the vectors y and y^{pr} are the same plant measurement and performance outputs, respectively. The matrix A_s is the plant state matrix and has the form

$$A_s = \begin{bmatrix} A_s^1 & 0 & \cdots & 0 \\ 0 & A_s^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \vdots & A_s^p \end{bmatrix} \tag{2}$$

where

$$A_s^i = \begin{bmatrix} 0 & 1 \\ -\omega_i^2 & -2\zeta_i\omega_i \end{bmatrix} \tag{3}$$

The matrix B_s is the control input influence matrix, formed by setting its odd-numbered rows to zeros and using the rows of $\bar{H}$ for its even-numbered rows:

$$B_s = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ \bar{H}_{11} & \bar{H}_{12} & \cdots & \cdots & \bar{H}_{1m} \\ 0 & 0 & \cdots & \cdots & 0 \\ \bar{H}_{21} & \bar{H}_{22} & \cdots & \cdots & \bar{H}_{2m} \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 0 \\ \bar{H}_{p1} & \bar{H}_{p2} & \cdots & \cdots & \bar{H}_{pm} \end{bmatrix} \tag{4}$$

in which $\bar{H}_{ij}$, for example, represents the (i, j) element of matrix $\bar{H}$. The matrix B_d is formed from $\bar{H}_d$ in the same manner.

The measurement output influence matrix C is defined by setting the odd-numbered columns of C to the columns of $\bar{C}_p$ and the even-numbered columns of C to the columns of $\bar{C}_r$:

$$C = \begin{bmatrix} \bar{C}_p(1,1) & \bar{C}_r(1,1) & \bar{C}_p(1,2) & \bar{C}_r(1,2) & \cdots & \cdots & \bar{C}_p(1,p) & \bar{C}_r(1,p) \\ \bar{C}_p(2,1) & \bar{C}_r(2,1) & \bar{C}_p(2,2) & \bar{C}_r(2,2) & \cdots & \cdots & \bar{C}_p(2,p) & \bar{C}_r(2,p) \\ \vdots & \vdots & \vdots & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots \\ \bar{C}_p(q,1) & \bar{C}_r(q,1) & \bar{C}_p(q,2) & \bar{C}_r(q,2) & \cdots & \cdots & \bar{C}_p(q,p) & \bar{C}_r(q,p) \end{bmatrix}$$

where $\bar{C}_p(i, j)$ and $\bar{C}_r(i, j)$ denote the (i, j) elements of matrices $\bar{C}_p$ and $\bar{C}_r$, respectively. C_1^{pr} is defined from $\bar{C}_p^{pr}$ and $\bar{C}_r^{pr}$ in the same fashion. C_2^{pr} is defined by setting the odd-numbered columns of C_2^{pr} to zeros and the even-numbered columns of C_2^{pr} to the columns of $\bar{C}_a^{pr}$.

By substituting the first equation of Eqs. (1) into the third, the acceleration term can be replaced by feedthrough:

$$\begin{aligned} \dot{\boldsymbol{x}}_s &= A_s \boldsymbol{x}_s + B_s \boldsymbol{u} + B_d \boldsymbol{w} \\ \boldsymbol{y} &= C \boldsymbol{x}_s \\ \boldsymbol{y}^{pr} &= C^{pr} \boldsymbol{x}_s + D_u^{pr} \boldsymbol{u} + D_w^{pr} \boldsymbol{w} \end{aligned} \tag{5}$$

The performance output influence matrix is given by

$$C^{pr} = C_1^{pr} + C_2^{pr} A_s$$

whereas the performance feedthrough matrices are

$$D_u^{pr} = C_2^{pr} B_s, \qquad D_w^{pr} = C_2^{pr} B_d$$

Notice that if there is no performance acceleration output ($C_a^{pr} = 0$), then $\bar{C}_a^{pr} = 0$ and $C_2^{pr} = 0$, and so both feedthrough matrices D_u^{pr} and D_w^{pr} are zero.

Control System Equations

In this paper it is assumed that the structure is controlled by a linear time-invariant control system. The model of a linear time-invariant control system for a typical flexible structure may be written as

$$\begin{aligned} \dot{\boldsymbol{x}}_c &= A_c \boldsymbol{x}_c + B_c \boldsymbol{y} \\ \boldsymbol{u} &= -C_c \boldsymbol{x}_c \end{aligned} \tag{6}$$

where $\boldsymbol{x}_c$ denotes the $k \times 1$ vector of control system states; A_c, B_c, and C_c represent the $k \times k$ control system state matrix, the $k \times q$ input influence matrix, and the $m \times k$ output influence matrix, respectively; and $\boldsymbol{y}$ is the measurement output vector, which was defined in the preceding section.

Frequency-Domain Equations

For the open-loop plant, $\boldsymbol{y}$ and $\boldsymbol{u}$ of Eq. (5) are nonexistent, and so the equations reduce to

$$\begin{aligned} \dot{\boldsymbol{x}}_s &= A_s \boldsymbol{x}_s + B_d \boldsymbol{w} \\ \boldsymbol{y}^{pr} &= C^{pr} \boldsymbol{x}_s + D_w^{pr} \boldsymbol{w} \end{aligned}$$

The open-loop transfer function from the disturbances $\boldsymbol{w}$ to the performance output $\boldsymbol{y}^{pr}$ is defined for all complex s not in the spectrum of A_s and is given by

$$T(s) = C^{pr}(sI - A_s)^{-1} B_d + D_w^{pr} \tag{7}$$

The closed-loop system is more complicated. Using Eqs. (5) and (6), the closed-loop dynamics of the controlled structure may be written as

$$\begin{aligned} \dot{\boldsymbol{x}} &= \tilde{A} \boldsymbol{x} + \tilde{B}_d \boldsymbol{w} \\ \boldsymbol{y}^{pr} &= \tilde{C}^{pr} \boldsymbol{x} + D_w^{pr} \boldsymbol{w} \end{aligned} \tag{8}$$

where $\boldsymbol{x}$ represents the closed-loop state vector defined as

$$\boldsymbol{x} = \begin{Bmatrix} \boldsymbol{x}_s \\ \boldsymbol{x}_c \end{Bmatrix} \tag{9}$$

$\tilde{A}$, $\tilde{B}$, and $\tilde{C}^{pr}$ are, respectively, the closed-loop state matrix, disturbance influence matrix, and output influence matrix. These matrices are given by

$$\tilde{A} = \begin{bmatrix} A_s & -B_s C_c \\ B_c C & A_c \end{bmatrix}, \qquad \tilde{B}_d = \begin{bmatrix} B_d \\ 0 \end{bmatrix}$$
$$\tilde{C}^{pr} = [C^{pr} \quad -D_u^{pr} C_c] \tag{10}$$

The closed-loop transfer function from the disturbances $\boldsymbol{w}$ to the performance output $\boldsymbol{y}^{pr}$ is defined for all complex s not in the spectrum of $\tilde{A}$ and is given by

$$\tilde{T}(s) = \tilde{C}^{pr}(sI - \tilde{A})^{-1} \tilde{B}_d + D_w^{pr} \tag{11}$$

Open-Loop Calculation

The algorithm presented here for calculation of the frequency response function of an open-loop structural system seems to be a part of engineering folklore. It is presented here for completeness and because it is a building block for the closed-loop algorithm to follow.

Assume s is not in the spectrum of A_s. From Eqs. (2) and (3), it follows that $(sI - A_s)^{-1}$ is block diagonal, with the ith block being

$$\left(sI_2 - A_s^i\right)^{-1} = \left\{ \frac{1}{\left(s^2 + 2\zeta_i \omega_i s + \omega_i^2\right)} \right\} \begin{bmatrix} s + 2\zeta_i \omega_i & 1 \\ -\omega_i^2 & s \end{bmatrix}$$
$$i = 1, 2, \ldots, p$$

where I_2 denotes the 2×2 identity matrix. Furthermore [see the discussion following Eq. (4)], the odd-numbered rows of B_d are zero. If the row i of B_d is denoted by $b_d^{(i)}$, if $Q(s)$ is used to represent $(sI - A_s)^{-1} B_d$, and if $Q(s)$ is partitioned as

$$Q(s) = \begin{bmatrix} Q_1(s) \\ Q_2(s) \\ \vdots \\ Q_p(s) \end{bmatrix} \tag{12}$$

where each partition matrix $Q_i(s)$ is a $2 \times r$ matrix, then Q is calculated using the formula

$$Q_i(s) = \left\{ \frac{1}{\left(s^2 + 2\zeta_i \omega_i s + \omega_i^2\right)} \right\} \begin{bmatrix} b_d^{(2i)} \\ s b_d^{(2i)} \end{bmatrix}, \qquad i = 1, 2, \ldots, p \tag{13}$$

The open-loop transfer function calculation is completed in a straightforward manner:

$$T(s) = C^{pr} Q(s) + D_w^{pr}$$

If desired, the gain and phase angle data (Bode plot data) may then be computed directly from the frequency response function matrix. If there are no acceleration performance measurements, then the feedforward term is zero and the software may bypass the step where D_w^{pr} is added in.

The standard, full-matrix way to calculate $Q(s)$ would involve first performing an LU decomposition of $sI - A_s$ (i.e., factoring $sI - A_s$ into lower triangular and upper triangular factors) followed by a backward and then forward solve of the triangular systems of equations using the columns of B_d as right-hand sides. The floating point operation (FLOP) count for this is $\mathcal{O}(p^3) + \mathcal{O}(p^2 r)$, and so $T(s)$ is computed in $\mathcal{O}(p^3) + \mathcal{O}(p^2 r) + \mathcal{O}(plr)$ FLOPs. Thus, in the typical case where system size is much larger than the number of disturbances, the calculation time per frequency point is a cubic function of system size.

If this calculation must be repeated for many values of s (a typical scenario), the technique of Laub[1,2] has a better average FLOP count. An initial $\mathcal{O}(p^3)$ orthogonal transformation must be done once; then for each s, the $Q(s)$ is calculated in $\mathcal{O}(p^2 r)$ FLOPs, so that $T(s)$ is computed in $\mathcal{O}(p^2 r) + \mathcal{O}(plr)$ FLOPs. Thus, if the number of frequencies for which this calculation must be repeated is on the order of the system size, or larger, the calculation time per frequency point is a quadratic function of system size.

When the calculation is done as in Eqs. (12) and (13), the FLOP count is $\mathcal{O}(pr)$, and so $T(s)$ is computed in $\mathcal{O}(plr)$ FLOPs. Thus, the calculation time per frequency point is a linear function of system size. This represents a substantial savings, particularly when a large number of modes is necessary to capture the dynamics of the system.

Closed-Loop Calculation

The closed-loop dynamics of the controlled system are given in Eqs. (8–10).

Observing the closed-loop state matrix $\tilde{A}$, it is obvious the block diagonal form of the open-loop plant has been destroyed by the coupling generated by the feedback connection of plant and the control system. However, the initial sparsity of the open-loop state matrix A_s is still intact. Now, the sparsity of the open-loop state matrix is exploited to develop an efficient method for the computation of the closed-loop frequency response function matrix of the controlled flexible structure. If sparsity is not exploited and many structural modes are modeled, it follows from Eq. (11) that a large computational effort would be required to calculate the closed-loop frequency response function matrix, since this would involve the computation of the matrix term $(sI - \tilde{A})^{-1}\tilde{B}_d$ with $s = j\omega$ for all desired frequency values ω.

In the following it is assumed that s is not in the spectrum of $\tilde{A}$ (necessary for the transfer function even to be defined) and it is further assumed that s is not in the spectrum of A_s. This further assumption is needed to enable some algebraic manipulation, and should not adversely affect the applicability of the following results. On the one hand, since A_s is the plant state matrix for a linear model of a flexible structure, its eigenvalues occur either at 0 (corresponding to rigid-body modes) or in the left-half plane (corresponding to damped flexible modes). On the other hand, it is anticipated that these results will be used to compute $\tilde{T}(s)$ for $s = j\omega$ with $\omega > 0$. Thus, excluding the eigenvalues of A_s from the domain of applicability of these results does not impact the anticipated usage.

The matrix term $(sI - \tilde{A})$ in Eq. (11) may be written as

$$(sI - \tilde{A}) = \begin{bmatrix} sI_s - A_s & B_s C_c \\ -B_c C & sI_c - A_c \end{bmatrix} \equiv \begin{bmatrix} E_{11}(s) & E_{12} \\ E_{21} & E_{22}(s) \end{bmatrix} \tag{14}$$

where I_s, and I_c are identity matrices of orders equal to the size of the plant state vector and the controller state vector, respectively. Introduce the following notation:

$$\begin{bmatrix} X_{11}(s) & X_{12}(s) \\ X_{21}(s) & X_{22}(s) \end{bmatrix} \equiv (sI - \tilde{A})^{-1} = \begin{bmatrix} E_{11}(s) & E_{12} \\ E_{21} & E_{22}(s) \end{bmatrix}^{-1} \tag{15}$$

The assumptions that have been made about s ensure that the inverses in Eq. (15) exist, as does E_{11}^{-1}. Rewrite Eq. (15) as

$$\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix} = \begin{bmatrix} I_s & 0 \\ 0 & I_c \end{bmatrix} \tag{16}$$

Expanding the lower left block of Eq. (16) and solving for X_{21} gives $X_{21} = -X_{22} E_{21} E_{11}^{-1}$. Expanding the lower right block of Eq. (16), substituting the previous expression for X_{21}, and factoring gives $X_{22}\Delta = I_c$, where $\Delta = E_{22} - E_{21} E_{11}^{-1} E_{12}$. This demonstrates that Δ is invertible and justifies the application of the block matrix inversion formula given in Ref. 3 to find the inverse of the block matrix in Eq. (15):

$$\begin{aligned} \Delta &= E_{22} - E_{21} E_{11}^{-1} E_{12} \\ X_{11} &= E_{11}^{-1} + E_{11}^{-1} E_{12} \Delta^{-1} E_{21} E_{11}^{-1} \\ X_{12} &= -E_{11}^{-1} E_{12} \Delta^{-1} \\ X_{21} &= -\Delta^{-1} E_{21} E_{11}^{-1} \\ X_{22} &= \Delta^{-1} \end{aligned} \tag{17}$$

Using Eqs. (10) and (15) in Eq. (11), the closed-loop transfer function from the disturbances to the performance outputs is reduced to

$$\begin{aligned} \tilde{T}(s) &= \begin{bmatrix} C^{pr} & -D_u^{pr} C_c \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} B_d \\ 0 \end{bmatrix} + D_w^{pr} \\ &= C^{pr} X_{11} B_d - D_u^{pr} C_c X_{21} B_d + D_w^{pr} \end{aligned}$$

Using Eq. (17), this becomes

$$\begin{aligned} \tilde{T} &= C^{pr} E_{11}^{-1} B_d + C^{pr} E_{11}^{-1} E_{12} \Delta^{-1} E_{21} E_{11}^{-1} B_d \\ &\quad + D_u^{pr} C_c \Delta^{-1} E_{21} E_{11}^{-1} B_d + D_w^{pr} \end{aligned}$$

Using Eq. (14), replace E_{12} and E_{21} in 1) the expression for Δ in Eq. (17) and 2) the preceding equation. This produces

$$\begin{aligned} \Delta &= E_{22} + B_c C E_{11}^{-1} B_s C_c \\ \tilde{T} &= C^{pr} E_{11}^{-1} B_d - C^{pr} E_{11}^{-1} B_s \left\{ C_c \Delta^{-1} B_c C E_{11}^{-1} B_d \right\} \\ &\quad - D_u^{pr} \left\{ C_c \Delta^{-1} B_c C E_{11}^{-1} B_d \right\} + D_w^{pr} \end{aligned} \tag{18}$$

In this form, the following computational efficiencies are observed:

1) Since $E_{11} = (sI - A_s)$ and B_s shares with B_d the property of having zeros in the odd-numbered rows, the terms $E_{11}^{-1} B_s$ and $E_{11}^{-1} B_d$ can be computed using the techniques presented for efficient computation of the open-loop transfer function [see Eq. (13)].

2) The computation of $\Delta^{-1} B_c$ must be done as a full-matrix computation, but since Δ is of the same order as the control system, which is usually small compared to the order of the analysis model of the plant, it should not be very costly to compute.

3) Common subexpressions, such as those mentioned in the preceding items and those enclosed in braces in Eq. (18) are computed once per frequency, saved, and reused.

4) The expected shapes of the matrices and the exploitation of common subexpressions make it advisable not to precompute some of the matrix products in Eq. (18) that are independent of the frequency parameter s [if X is a tall, skinny matrix and Y is a short, wide matrix, so that $W = XY$ is both tall and wide, and if z is a vector, then calculating $X(Yz)$ is cheaper than calculating Wz].

5) If there are no acceleration sensors in the performance outputs, so that the feedforward matrices are zero, the software may bypass the second line of the $\tilde{T}$ calculation in Eq. (18).

Now, using Eq. (18) to calculate $\tilde{T}(s)$, the frequency response function matrix of the closed-loop system is evaluated for various values of $s = j\omega$, with ω taking on the user-specified frequency values. The closed-loop gain and phase plots (Bode plots) may then be computed directly from the frequency response function matrix, if desired.

The closed-loop system matrix $\tilde{A}$ [Eq. (10)] has order $2p + k$. As in the discussion of the open-loop calculation, if $\tilde{T}(s)$ is calculated as in Eq. (11) using standard full-matrix techniques, the computation takes $\mathcal{O}[(2p + k)^3] + \mathcal{O}[(2p + k)^2 r] + \mathcal{O}[(2p + k)lr]$ FLOPs per frequency point. Using the technique of Refs. 1 and 2, if the number of frequency points for which the calculation is to be repeated

is on the order of $2p + k$ or more, then $\tilde{T}(s)$ can be calculated in $\mathcal{O}[(2p+k)^2 r] + \mathcal{O}[(2p+k)lr]$ FLOPs per frequency point. Again, these are cubic and quadratic functions, respectively, of the system size. By counting FLOPs resulting from subroutine calls and DO loops in the Fortran software used to implement the calculation of Eq. (18), it is determined that $\tilde{T}(s)$ can be calculated in

$$\mathcal{O}[p(lr + rq + qm + rm)] + \mathcal{O}[k^3 + k^2 r + k(rq + qm + rm)] + \mathcal{O}(lrm)$$

FLOPs per frequency point. Again, this is a linear function of system size.

In future work, it is expected that the $\mathcal{O}(k^3)$ in this last expression, which comes from performing an LU decomposition on the matrix Δ, will be reduced to $\mathcal{O}(k^2)$. This will be based on applying the technique of Laub[1,2] to E_{22} and making use of the observation that the other term in the definition of Δ is, in the expected applications, of low rank.

Software Implementation

The evaluation of the open- and closed-loop transfer function has been implemented using MATLAB function M-file [MATLAB, a product of The MathWorks, Inc., is "a technical computing environment for high-performance numeric computation and visualization" (Ref. 4)] and a Fortran 77 code, which is then accessed through MATLAB using the MEX-file external interface facility. The M-files contain straightforward implementations of the calculations presented in the preceding two subsections.

The Fortran source code for the MEX-file uses the BLAS (Basic Linear Algebra Subprograms[5–11]) to perform vector–vector, vector–matrix, and matrix–matrix operations. In addition, LAPACK[12] subroutine ZGESV, a complex double precision linear equation solver, is used to calculate $\Delta^{-1} B_c$.

Numerical Examples

A number of numerical examples are presented to demonstrate the efficiency and accuracy of the algorithm presented in this paper compared to two standard full-matrix methods of calculating the frequency response function of a closed-loop system.

EOS-AM-1 Spacecraft Model

The data come from a model for the EOS-AM-1 spacecraft used in a jitter reduction study.[13] The structural model contains 703 modes for a potential 1406 plant states. There are six rigid-body modes and flexible modes ranging from 1.24 to 1564 rad/s. The six measurement outputs are the spacecraft's roll, roll rate, pitch, pitch rate, yaw, and yaw rate measurements at the spacecraft navigational unit. Actuators consist of x-, y-, and z-axis torquers. The control system has 39 states. Up to 10 channels of disturbance input and 27 channels of performance measurement output were used. Each case was run using position measurements at each output, resulting in no feedforward term, and run using acceleration measurements at each output, resulting in a feedforward term being present.

All algorithms used in this timing study are intended to be used to calculate the frequency response matrix at multiple frequency points, so that the frequency response may be plotted (as, e.g., Bode plots). They all have some calculations that are done once per entry into the algorithm and other calculations that are done once for every frequency point. To take this into account, all cases were run over a range of 200 frequency points and most of them were rerun using 2000 points (the exceptions were the cases that would have required 5+ days of CPU time to complete). In all cases, the points were logarithmically distributed between frequencies of 0.01 and 10,000 rad/s.

Software Used in Timing Studies

In this study, two software realizations of the closed-loop frequency response function calculations are compared to two software realizations of previously available algorithms.

The present algorithm is programmed both as a MATLAB function M-file and as Fortran 77 code, which is then accessed through MATLAB using the MEX-file external interface facility. These will be called, respectively, the new M-code and the new MEX-code.

One of the programs used for comparison makes use of the algorithm in Refs. 1 and 2. The Fortran code in Ref. 2 is in single precision; Laub's own double precision Fortran code is imbedded in the software package FREQ[14] and was used here. This test code is purely in Fortran 77. This will be called the old Fortran code.

Preliminary testing indicated that to get a reasonably well-conditioned matrix for the $sI - \tilde{A}$ expression in the Laub code it was necessary to exercise the built-in option of balancing the $\tilde{A}$ matrix. The unbalanced matrix was particularly ill conditioned at low frequencies. This can be attributed to the presence of the rigid-body (0 frequency) modes. In the Laub code, balancing was coupled with the extraction of the eigenvalues of $\tilde{A}$. As Laub wrote this code, the same value of the input flag that signaled the code to balance the $\tilde{A}$ matrix also signaled the code to extract its eigenvalues. For purposes of timing tests here, the Laub code was modified so that the portion that extracts eigenvalues was bypassed.

The other program used for comparison is the MathWorks M-file freqrc.m, an undocumented utility routine in their "Robust Control Toolbox" software package,[15] which calculates (to quote the program preamble comments) "Continuous complex frequency response (MIMO)." This will be called the old M-code. Once again, to achieve reasonable accuracy, it was necessary to balance $\tilde{A}$. This was done using MATLAB built-in routine BALANCE.

Timing Comparisons

The executions times of the four test codes are compared on 12 representative problems. Three different plant sizes were used: a small plant with 1 input, 1 output, and 61 states (24 structural and 39 controller states); a medium plant with 3 inputs, 5 outputs, and 221 states (184 structural and 39 controller states); and a large plant with 10 inputs, 27 outputs, and 1445 states (1406 structural and 39 controller states). For each plant size, two plants were used: one with no feedforward term (i.e., no acceleration outputs were used as performance outputs) and one with feedforward. The frequency response function of each of these 6 plants was calculated at a short (200 values) vector of frequency values and at a long (2000 values) vector of frequency values.

Particularly on the larger plants, the new algorithm performs dramatically faster than the older programs. This should not be taken as an indictment of the older technology. The older technology was designed to apply to an arbitrary plant, whereas the new algorithm takes full advantage of the particular pattern of sparsity that results from using the modal model of a flexible structure. On the other hand, when the new technology is applicable, it enables analysis of structures of much larger order than would be practical or even possible with the older technology.

Table 1 gives the time in seconds to calculate the frequency response function using each of the 4 test routines for each of these 12 cases (except that the old M-code does not attempt the two largest cases).

One conclusion to be drawn from this table is that the timing values returned by the system timing software are not totally consistent with each other. The first three software packages in that table all check up front to see whether feedforward is present. The bulk of the code is executed whether feedforward is present or not. If feedforward is present, additional code is executed, which should take additional time. But in 9 of 18 cases, the table shows the feedforward case taking less time than the one without.

There are still significant trends to be observed in these timing data. The new MEX-code is significantly faster than the M-code; in the more important larger cases, about three times as fast. This justifies the effort of rendering the algorithm in Fortran and writing the interface necessary to access it through MATLAB.

Comparing the new MEX-code, which is Fortran based, with the old Fortran code shows that for the small system, the old code more than holds its own. This is not unexpected, since in the small system, the controller dominates the count of states. Thus, there is relatively little of the sparsity from the structural part of the plant of which the new MEX-code may take advantage. But even in the medium-size case, the new code is four to seven times as fast as the

Table 1 Time to calculate frequency response function

System size[a]	Freq. vector length	Feed-forward present	New MEX-code	New M-code, s	Old Fortran code, s	Old M-code, s
1/1 24 + 39	200	no	1.98	11.18	1.26	7.57
1/1 24 + 39	200	yes	2.47	13.50	1.10	7.62
1/1 24 + 39	2000	no	24.78	108.87	10.08	71.18
1/1 24 + 39	2000	yes	19.60	135.20	10.36	71.92
3/5 184 + 39	200	no	5.03	18.53	29.84	287.93
3/5 184 + 39	200	yes	3.85	14.18	27.50	290.25
3/5 184 + 39	2000	no	46.08	147.00	213.69	2,818.78
3/5 184 + 39	2000	yes	35.77	188.93	200.09	2,830.32
10/27 1406 + 39	200	no	60.40	231.25	5,714.94	49,846.62
10/27 1406 + 39	200	yes	73.77	227.40	5,756.99	49,199.20
10/27 1406 + 39	2000	no	591.50	1922.75	24,928.01	——
10/27 1406 + 39	2000	yes	615.73	1897.22	24,929.14	——

[a]In m/n in the first line, m is the number of inputs and n is the number of outputs. In $m + n$ in the second line, m is the number of structural states and n is the number of controller states.

old. This is getting near the size at which conventional numerical analytic wisdom would place the limits of applicability of the old, full-matrix based, technique. Since the time for the old Fortran code is expected to grow quadratically with the number of system states whereas that of the new technique is expected to grow only linearly, it is not surprising that the difference between them in the largest example is so great.

In what is called here the old M-code, MathWorks actually used M-files for outer loop logic control to drive a built-in function, LTIFR. This function calculates the matrix G whose columns are $[s(i)I - A]^{-1}b$ where s is a vector of complex numbers (set in the present application to $j\omega$, where $j^2 = -1$ and ω is a vector of frequencies) and b is a column vector (set in this application to a column of the B matrix). The on-line help for LTIFR states that it "implements, in high speed" what the user could calculate by looping through the elements of s and building G one column at a time. Despite this, it is only competitive on the smallest system, and then only against the new M-code, which utilizes MATLAB built-in functions only at the more primitive level of basic matrix operations.

All of the algorithms tested do have some once-per-entry calculations in addition to the calculations that occur once per frequency value. Thus, the time for the 2000 point calculations should never be more than 10 times that for the 200 point calculations. In Table 1, there are several exceptions to this. This reinforces the earlier remark that the numbers returned by the computer system timing routines are, at best, approximate. However, from looking at the largest case, it can be reasonably concluded that the once per entry overhead is fairly small in both realizations of the new algorithm while being substantial in the old Fortran code, at least for large systems. This is expected, since for a system of order n (all other parameters being held fixed), the once per entry overhead in the old Fortran code includes the initial reduction, which takes $\mathcal{O}(n^3)$ FLOPs, whereas the per frequency point calculation takes $\mathcal{O}(n^2)$ FLOPs.

Accuracy

No formal error analysis has been performed on the new algorithm. There is, however, numerical evidence to support the thesis that the new algorithm is more accurate than the older techniques, particularly when applied to larger systems.

Outputs from the four algorithm realizations were compared. For each frequency value, individual entries in the frequency response matrices computed by the four codes were compared using a symmetric relative error: The discrepancy between complex numbers z and ω (not both 0) was measured by

$$\delta(z, w) = \frac{|z - w|}{0.5(|z| + |w|)}$$

This error measure ranges from a minimum of 0 to a maximum of 2. A value of $\delta(z, w)$ near 10^{-n} indicates that z and w agree to about n decimal places whereas $\delta(z, w) > 0.1$ indicates anything from rough approximation (near 0.1) to no correlation (bigger than, say, 1). For each fixed frequency, the worst discrepancy over all possible input–output pairs was observed.

The size of the discrepancy between the frequency response function matrices computed by these codes was observed to depend not only on which two of the codes were being compared but also on the size of the system, the frequency, and whether or not feedforward was present. It would take too much space to present details of these comparisons. However, some general statements can be made.

For each of the test problems (corresponding to one row of Table 1), the results produced by the four codes were compared two by two. The overall best agreement between any pair of calculations came from comparing the outputs of the new MEX-code and the new M-code. At worst, these agree to about seven decimal places. This generally improves with reduction of system size or increase in frequency so that best agreement is within machine accuracy. No other pairing, either between one of the new codes and one of the old or between the two old codes, ever showed noticeably better agreement, and in general the agreement was much worse. It frequently occurred that the results from comparing the two new codes showed that the agreement of their computations was better than that of any other pairing by at least two decimal places. In the largest system, this advantage could increase to five decimal places, particularly for small frequencies or when no feedforward was present.

Thus, two dissimilar implementations of the new algorithm produce results in good agreement. When a parallel process is applied to the older algorithm, the two dissimilar implementations produce results that are not in as good agreement, either with each other or with those of the new algorithm.

To provide further evidence that the results of the new algorithm are the more accurate, the old Fortran code was translated to quadruple precision from its native double precision and was run (at a time penalty of about 32 times) on the medium-sized problem using no

feedforward and 200 frequency points. The output from this showed the same degree of agreement with the output from the new codes as they showed with each other.

These results combine to indicate strongly that the new algorithm provides more accurate results than those previously available. There are theoretical grounds for expecting this. The old way requires the solution of linear systems with the coefficient matrix $sI - \tilde{A}$, which is usually of large order. It also had conditioning problems that balancing ameliorated but did not totally eliminate.

In the new algorithm, the coefficient matrices involved in the solution of linear systems are Δ, which has the same order as the controller, and, for $i = 1, \ldots, p$, $sI_2 - A_s^i$, which is of order 2. Particularly when dealing with a large-order structural model, the coefficient matrices used by the new algorithm are much smaller than the matrix $sI - \tilde{A}$ used by the old, and so there is much less opportunity for roundoff error. Any conditioning problems coming from the interaction of the frequency represented by $s = j\omega$ and the ith structural mode in the old method is isolated in the new method to calculating the denominator term $\omega_i^2 - \omega^2 + 2j\zeta_i\omega_i\omega$ in Eq. (13); this only gives numerical problems if ω is so close to ω_i that truncation occurs in forming the difference and ζ_i is so small that the (small) real part $\omega_i^2 - \omega^2$ is a significant part of the whole term.

Summary

An efficient and novel procedure has been developed for the calculation of the frequency response function of a large-order, flexible system implemented with a linear, time-invariant control system. The procedure takes advantage of the highly structured sparsity of the system matrices of the plant in normal mode coordinates. This reduces the computational cost from a quadratic function of the order of the system to a linear function, thereby permitting the practical frequency analysis of systems of much larger order than by traditional, full-matrix means. Formulations have been given for both open- and closed-loop systems. Numerical examples were presented wherein the advantages of the present formulation over traditional approaches, both in speed and in accuracy, have been demonstrated. When exercised on the largest systems, the new MEX-code was about 40 times as fast as the old Fortran code when many frequency points were used. The advantage increased to a factor of about 80 or better when the calculation involved few frequency points. In this latter case, the new M-code was over 200 times as fast as the old M-code.

References

[1]Laub, A. J., "Efficient Multivariable Frequency Response Computations," *IEEE Transactions on Automatic Control*, Vol. AC-26, No. 2, 1981, pp. 407, 408.

[2]Laub, A. J., "ALGORITHM 640—Efficient Calculation of Frequency Response Matrices from State Space Models," *ACM Transactions on Mathematical Software*, Vol. 12, No. 1, 1986, pp. 26–33.

[3]Ogata, K., *Modern Control Engineering*, 2nd ed., Prentice–Hall, Englewood Cliffs, NJ, 1990, p. 898.

[4]Anon., "MATLAB High-Performance Numeric Computation and Visualization Software, Reference Guide," MathWorks, Inc., Natick, MA, Aug. 1992, p. i.

[5]Lawson, C. L., Hanson, R. J., Kincaid, D. R., and Krogh, F. T., "Basic Linear Algebra Subprograms for Fortran Usage," *ACM Transactions on Mathematical Software*, Vol. 5, No. 3, 1979, pp. 308–323.

[6]Lawson, C. L., Hanson, R. J., Kincaid, D. R., and Krogh, F. T., "Algorithm 539: Basic Linear Algebra Subprograms for Fortran Usage [f1]," *ACM Transactions on Mathematical Software*, Vol. 5, No. 3, 1979, pp. 324, 325.

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

[8]Dongarra, J. J., Du Croz, J., Hammarling, S., and Hanson, R. J., "An Extended Set of Fortran Basic Linear Algebra Subprograms," *ACM Transactions on Mathematical Software*, Vol. 14, No. 1, 1988, pp. 1–17.

[9]Dongarra, J. J., Du Croz, J., Hammarling, S., and Hanson, R. J., "Algorithm 656: An Extended Set of Fortran Basic Linear Algebra Subprograms," *ACM Transactions on Mathematical Software*, Vol. 14, No. 1, 1988, pp. 18–32.

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

[11]Dongarra, J. J., Du Croz, J., Duff, I. S., and Hammarling, S., "Algorithm 679: A Set of Level 3 Basic Linear Algebra Subprograms," *ACM Transactions on Mathematical Software*, Vol. 16, No. 1, 1990, pp. 18–28.

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

[13]Belvin, W. K., Maghami, P. G., Tanner, C. E., Kenny, S. P., and Cooley, V. M., "Evaluation of CSI Enhancements for Jitter Reduction on the EOS AM-1 Observatory," *Dynamics and Control of Large Structures, Proceedings of the 9th VPI&SU Symposium*, edited by L. Meirovitch, Virginia Polytechnic Inst. and State Univ., Blacksburg, VA, 1993, pp. 255–266.

[14]Giesy, D. P., and Armstrong, E. S., "FREQ: A Computational Package for Multivariable System Loop-Shaping Procedures," NASA TM 4127, National Technical Information Service, Springfield, VA, Sept. 1989.

[15]Anon., "Robust Control Toolbox," MathWorks, Inc., Natick, MA, Aug. 1992.