

CU-CSSC-91-11

CENTER FOR SPACE STRUCTURES AND CONTROLS

AD-A238 851

ALL FORCE
NOTICE
STINand is
-12

**DEVELOPMENT OF NONLINEAR
TRANSIENT ANALYSIS ALGORITHMS
FOR DYNAMICS AND CONTROL OF
LARGE SPACE STRUCTURES**

DTIC
JUL 23 1991
S C D

by

K. C. Park and C. A. Felippa

May, 1991

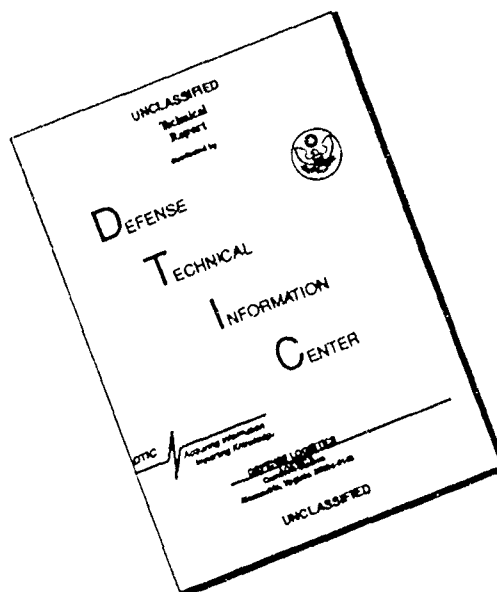
COLLEGE OF ENGINEERING
UNIVERSITY OF COLORADO
CAMPUS BOX 429
BOULDER, COLORADO 80309

91-05847



91 7 22 046

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE May 1991	3. REPORT TYPE AND DATES COVERED Final Report 10/1/87 to 12/31/90	
4. TITLE AND SUBTITLE Development of Nonlinear Transient Analysis Algorithms for Dynamics and Control of Large Space Structures (le)			5. FUNDING NUMBERS Grant F49620-87-C-0074	
6. AUTHOR(S) K. C. Park and C. A. Felippa				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Center for Space Structures and Controls Campus Box 429 University of Colorado Boulder, CO 80309			8. PERFORMING ORGANIZATION REPORT NUMBER CU-CSSC-91-11	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR (Dr. Spencer Wu) Bolling Air Force Base Washington, DC 20332-6448			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Available to the Public			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This is a final report on the research project supported by the Air Force Office of Scientific Research under Grant F49620-87-C-0074, entitled <i>Development of Nonlinear Transient Analysis Algorithms for Dynamics and Control of Large Space Structures</i> , which covered the period of 01 October 1987 to 31 December 1990. The objectives of the research have been: (1) the development of improved dynamic modeling techniques that can capture essential response components encompassing an adequate frequency spectrum in large space structures including dynamic localizations, and (2) the development of efficient algorithms for solving the combined structural and control dynamics equations by taking advantage of computer hardware and software advances such as concurrent computers.				
14. SUBJECT TERMS Space Structures Nonlinear Transient Analysis Dynamics and Control			15. NUMBER OF PAGES 17 (1000 (approx))	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to *stay within the lines* to meet optical scanning requirements.

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract	PR - Project
G - Grant	TA - Task
PE - Program Element	WU - Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."

DOE - See authorities.

NASA - See Handbook NHB 2200.2.

NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - Leave blank.

DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.

NASA - Leave blank.

NTIS - Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

A-1

K. C. Park and C. A. Felippa

COLLEGE OF ENGINEERING
UNIVERSITY OF COLORADO
CAMPUS BOX 429
BOULDER, COLORADO 80309

Final Report

on

**Development of Nonlinear Transient Analysis Algorithms
for
Dynamics and Control of Large Space Structures**

K.C. PARK AND C. A. FELIPPA

*Department of Aerospace Engineering Sciences and
Center for Space Structures and Controls
University of Colorado, Boulder, CO 80309-0429*

May 1991

Report No. CU-CSSC-91-11

Final Report on Grant F49620-87-C-0074, funded
by the Air Force Office of Scientific Research (AFOSR)

TABLE OF CONTENTS

SUMMARY

ENCLOSED PhD THESES AND PAPERS

W. K. Belvin

Simulation and Interdisciplinary Design Methodology
for Control-Structure Interaction Systems

Ph.D. Thesis, Center for Space Structures and Controls
University of Colorado, Report No. CU-CSSC-89-10
July, 1989.

J. D. Downer

A Computational Procedure for the Dynamics of
Flexible Beams Within Multibody Systems,
Ph.D. Thesis, Center for Space Structures and Controls
University of Colorado, Report No. CU-CSSC-90-27
November, 1990.

J. C. Chiou

Constant Treatment Techniques and Parallel
Algorithms for Multibody Dynamic Analysis
Ph.D. Thesis, Center for Space Structures and Controls
University of Colorado, Report No. CU-CSSC-90-26
November, 1990.

L. A. Crivelli

A Total-Lagrangian Beam Element for Analysis of
Nonlinear Space Structures
Ph.D. Thesis, Center for Space Structures and Controls
University of Colorado, Report No. CU-CSSC-91-07
April, 1991.

K. F. Alvin and K. C. Park

Implementation of A Partitioned Algorithm for
Simulations of Large CSI Problems
Center for Space Structures and Controls
University of Colorado, Report No. CU-CSSC-91-04

K. C. Park, J. D. Downer, J. C. Chiou and C. Farhat
A Modular Multibody Analysis Capability for
High-Precision, Active Control and Real-Time Applications
to appear in *International Journal for Numerical
Methods in Engineering*, 1991.

C. A. Felippa and L. A. Crivelli
The Core-Congruential Formulation of
Geometrically Nonlinear TL Finite Elements
Center for Space Structures and Controls
University of Colorado, Report No. CU-CSSC-91-01
To appear in *Nonlinear Computational Mechanics:
The State of the Art*, Springer-Verlag, Berlin, 1991.

K. C. Park, and W. K. Belvin
A Partitioned Solution Procedure for Control-Structure
Interaction Simulations
J. Guidance, Control and Dynamics, 14(1), Jan.-Feb. 1991, 59-67.

K. C. Park, K. F. Alvin and W. K. Belvin
Parallel Computations and Control of Adaptive Structures
in: *Intelligent Structures*, ed. by K. C. Chong, S. C. Liu
and J. C. Li, Elsevier Science Publishers, London, 1990, 439-458.

W. K. Belvin and K. C. Park,
Structural Tailoring and Feedback Control Synthesis:
An Interdisciplinary Approach
J. Guidance, Control, and Dynamics, 13(3),
May -June 1990, 424-429.

W. K. Belvin and K. C. Park
Computer Implementation of Analysis and Optimization
Procedures for Control-Structure Interaction Problems
Proc. the 1990
AIAA Dynamics Specialist Conference, Paper No. AIAA-90-1194
Long Beach, Calif., 5-6 April 1990.

K. C. Park and W. K. Belvin

Second-Order Discrete Kalman Filtering Equations
for Control-Structure Interaction Simulations

Proc. AIAA Guidance, Control and Navigation Conference
Paper No. AIAA- 90-3387-CP, Portland, Ore., August 20-22
1990, pp.653-660.

Belvin, W. K. and K. C. Park

On the State Estimation of Structures with Second
Order Observers

Proc. the 30th Structures, Dynamics and Materials
Conference, AIAA Paper No. 89-1241, April 3-5, 1989.

K. C. Park and W. K. Belvin

Stability and Implementation of Partitioned CSI
CSI Solution Procedures

Proc. the 30th Structures, Dynamics and Materials
Conference, AIAA Paper No. 89-1238, April 3-5, 1989.

C. Farhat and L. Crivelli

A General Approach to Nonlinear FE Computations
on Shared Memory Multiprocessors

Computer Methods in Applied Mechanics and
Engineering, Vol. 72, pp. 153-172, 1989.

SUMMARY

This is a final report on the research project supported by the Air Force Office of Scientific Research under Grant F49620-87-C-0074, entitled *Development of Nonlinear Transient Analysis Algorithms for Dynamics and Control of Large Space Structures*, which covered the period of 01 October 1987 to 31 December 1990. The objectives of the research have been: (1) the development of improved dynamic modeling techniques that can capture essential response components encompassing an adequate frequency spectrum in large space structures including dynamic localizations, and (2) the development of efficient algorithms for solving the combined structural and control dynamics equations by taking advantage of computer hardware and software advances such as concurrent computers. We now summarize the accomplishments of the project.

I. Research Accomplishments

1. Modeling Techniques of Space Truss Structures and Solution Techniques:

Three-dimensional beam models have been formulated for arbitrarily large rigid/flexible motions by two complementary approaches: total Lagrangian and convected coordinate representations. These formulations have been demonstrated through numerical implementations for beams going through large bending and torsional motions, dynamic spinning, and deployment problems. Emphasis of these formulation was to provide an efficient interface with control algorithms for simulating maneuvering and reorientations of flexible space structures undergoing large rotations. Specific accomplishments include: 1) an efficient and accurate algorithm for the computations of angular orientations; 2) a convected rigid-body preserving procedure for the computations of strain energy when the beam is subjected to large motions; 3) an element-independent formulation for the total-Lagrangian derivation of material stiffness, geometric stiffness, and internal force vector for an accurate prediction of bifurcation and post-bifurcation states of the beam.

2. Algorithms for Control-Structure Interaction Analysis:

Computational algorithms for control-structure interaction problems have been developed, which exploit the second-order differential equations of motion for structures whose discrete coefficient matrices are of N -by- N sparse symmetric form where N is the size of the equations of motion. The present approach is in contrast to the widely adopted canonical first-order differential equations of $2N$ -by- $2N$ form, which requires a considerably more computational effort than the N -by- N second-order models. The developed algorithms include: 1) a partitioned procedure that advances the solution of the structural responses separately from that of the control forces, thus preserving the software modularity of both the structural analysis and control force computation software modules; 2) a suboptimal control gain synthesizer based on the physical matrix forms of the second-order structural dynamics equations; 3) a second-order discrete Kalman filtering solver that is necessary for the synthesis of dynamically compensated control laws; 4) a strategy for simultaneous optimization of the structural tailoring and control laws.

II. Manpower Training

The project has enabled, along with two other related research grants, four graduate students to complete their PhD theses. Their work and their post-degree activities are briefly described below.

W. Keith Belvin

Degree Received and Date: Ph.D. (August 1989)

Thesis Title: *Simulation and Interdisciplinary Design*

Methodology for Control-Structure Interaction Systems

Post-Degree Appointment: Aerospace Engineer,

NASA Langley Research Center

Thesis Synopsis:

Computational procedures for preliminary design, synthesis, and simulation of control-structure interaction systems are developed from a second-order systems viewpoint. First, a partitioned solution procedure is developed for the solution of the equations of motion with feedback controls. The computational stability and implementation aspects of the partitioned computational procedure for control-structure interaction analysis is presented. Implementation issues for enhancing both accuracy and the computational stability margin and modular programming of the procedure are addressed.

Second, a computational procedure that takes advantage of the structure of second-order differential equations is developed for state estimation. It is shown that the reconstruction of system states through second-order observers is computationally more attractive than existing first-order observer models if certain mildly restrictive conditions are satisfied. Hence, the present second-order observers enable the analyst to utilize larger observer models than have been possible with first-order observer models.

The third issue addressed is the integrated design of controlled structures. A method for optimization of the controlled structural system using only structural tailoring is presented. Optimal linear quadratic regulator control theory is used and the closed-loop performance is expressed in terms of structural parameters. Also, a software testbed has been designed and implemented for evaluating new formulations and algorithms for controlled structural analysis and design. The testbed fully exploits existing sparse matrix structural analysis techniques, modular control synthesis capabilities and versatile optimization methods to alleviate unnecessary structural and control calculations. The testbed has been used to demonstrate the present partitioned procedures, the performance of second-order observers and the effectiveness of structural tailoring for improved dynamics and control performance on a number of realistic spacecraft applications.

Janice D. Downer

Degree Received and Date: Ph.D. (December 1990)

Thesis Title: *A Computational Procedure for the Dynamics of Flexible Beams Within Multibody Systems*

Post-Degree Appointment:

Assistant Professor (beginning September 1991)

University of Wisconsin, Madison, WI

Thesis Synopsis:

This dissertation is concerned with the dynamic analysis of three-dimensional elastic beams which experience large rotational and large deformational motions. To this end, the beam motion is modeled using an inertial reference for the translational displacements and a body-fixed reference for the rotational quantities. Finite strain rod theories are then defined in conjunction with the beam kinematic description which account for the effects of stretching, bending, torsion, and transverse shear deformations. A convected coordinate representation of the Cauchy stress tensor and a conjugate strain definition is introduced to model the beam deformation. Due to the inertial reference of the beam kinematics and the convected reference of the beam stresses, the present formulation is easily interfaced with general multibody dynamics methodologies as well as software modules for active control simulations.

The numerical treatment of the beam formulation is considered in detail. A procedure to compute the beam internal force is derived from the continuum formulation. The procedure is proven to be invariant to arbitrary rigid motions of the beam while accurately modeling the beam strain. To treat the beam dynamics, a two-stage modification of the central difference algorithm is presented to integrate the translational coordinates and the angular velocity vector. The angular orientation is then obtained from the application of an implicit integration algorithm to the Euler parameter/angular velocity kinematical relation. The combined developments of the objective internal force computation with the dynamic solution procedures result in the computational preservation of total energy for undamped systems.

The present methodology is also extended to model the dynamics of deployment/retrieval of the flexible members. A moving spatial grid corresponding to the configuration of a deployed rigid beam is employed as a reference for the dynamic variables. A transient integration scheme which accurately accounts for the deforming spatial grid is derived from a space-time finite element discretization of a Hamiltonian variational statement. The computational results of this general deforming finite element beam formulation are compared to reported results for a planar inverse-spaghetti problem.

Jin-Chern Chiou

Degree Received and Date: Ph.D. (December 1990)

Thesis Title: *Constant Treatment Techniques and Parallel Algorithms for Multibody Dynamic Analysis*

Post-Degree Appointment: Research Associate,
University of Colorado

Thesis Synopsis:

In the distinct fields of space dynamics, mechanism, and robotics, engineers have effectively modeled their disciplinary problems using a set of linked bodies. In this regard, computational procedures for the kinematic and dynamic analysis of three dimensional multibody dynamic (MBD) systems are developed. To derive the equations of motion for MBD systems, d'Alembert's principle of virtual work augmented by the Lagrange multipliers technique are employed. The resulting dynamical equations, which are characterized as differential-algebraic equations (DAEs), not only possess programming modularity but also can be automatically generated.

The existing computational schemes that have been used to solve DAEs have suffered from the drawbacks such as constraint violations during the process of time integration, degraded constraint force solution involving ill-conditioned matrix, the large computation expense in determining system dependent and independent variables, and inefficient to

apply to the parallel algorithms. To alleviate these drawbacks, two robust and efficient computational schemes which involve constraint stabilization and constraint elimination are developed to effectively overcome these drawbacks. A explicit-implicit solution procedure that is based on the proposed computational schemes is also developed to integrate the equations of motion and obtain accurate solutions. The developed solution procedure can be readily adopted into the parallel computation environment necessary to achieve real-time simulation. In this regard, a Schur complement based parallel preconditioned conjugate gradient numerical algorithm is employed to speed up the whole computational process. By transforming the proposed computational schemes into the arrowhead matrix, we have successfully altered the dynamic equations into the Schur complement form so that proposed parallel algorithm can be equally applied. By utilizing current existing parallel computers, this algorithm has dramatically reduce the computer run time for some testing MBD problems.

Luis A. Crivelli

Degree Received and Date: Ph.D. (May 1991)

Thesis Title: *A Total-Lagrangian Beam Element*

for Analysis of Nonlinear Space Structures

Post-Degree Appointment: Research Associate,

University of Colorado

Thesis Synopsis:

A total-Lagrangian formulation for the large-displacement large-rotation static and dynamic analysis of beams is presented. The Timoshenko beam theory is used, in which cross-section rotations are independent of the motion of the neutral axis. The beam motion is referred to an inertial reference frame and it is split into a translational component — the motion of the neutral axis— and a cross-section rotational component. Specialized forms of the Green-Lagrange strain and the second Piola-Kirchhoff stress are obtained. The formulation uses a large deformation measure, and it accounts for bending-stretching and bending-torsional coupling effects without introducing special provisions. A kinematically-independent formulation is developed which allows writing the core components of the finite element matrices independently of the parametrization chosen for the large rotations. The corresponding representation of these core matrices in terms of a given parametrization is obtained through simple matrix transformations. Results are presented for the rotational vector representation. A physical interpretation of the matrix equations is also given, which allows us to obtain a corotational formulation through a-posteriori simplifications of the governing equations. The conceptual equivalence of large deformation theories with

corotational theories is also shown. The formulation is implemented into a stand-alone C-code software module where new ideas for finite element data structure and organization are being tested. Provisions for implementation on shared memory parallel computers have also been taken. This formulation appears suitable for static and dynamic modeling of large space structures, limit and post-critical analysis, simulation of large rotational maneuvering problems as well as control-structure interaction problems.

III. Impact of the Project

The project has had an impact on the following activities.

1. Preliminary Simultaneous Design of Controls and Structures: The use of physically based second-order control laws developed during the project period has enabled the preliminary designer to tailor the structural parameters as an intrinsic function of control effort or to size up/down the control parameters as an apparent function of physical sizes, thus allowing simultaneous design of the control system and the structure in terms of physical parameters. This is in contrast to current control design practices that frequently lead to control gains that cannot be translated into corresponding mechanical parameters, which makes it difficult to realize a simultaneous control/structural trade-off studies. The preliminary design technique developed during the project has been extensively applied at NASA Langley Research Center in the design of an evolutionary earth-observing platform, and incorporated in further research by researchers at UCLA, VPI/SU and Purdue, among others.

2. Second-Order Simulation Methods for Control-Structure Interactions: In order to achieve the simulation of control-structure interactions (CSI) in terms of second-order solution methods, three related algorithms have been developed. The first is a control law synthesis based on second-order form as discussed above so that widely used solution methods for the structural equations of motion can be applied for CSI simulations. The second algorithm is a partitioned solution procedure that solves the control laws cast in a first-order temporal differential form in a separate solution module so that the structural responses can be obtained by widely available second-order structural analyzers. The third algorithm is a discrete second-order Kalman filtering solver that can be used in a dynamically compensated feedback loop in conjunction with the first two algorithms. Because of substantial computational advantages that the second-order CSI algorithms can offer, our research results have motivated leading researchers such as Drs. J. N. Juang and Richard Longmont to develop their feedback laws based on the pole-zero placement techniques that can be directly implementable on the second-order form. In addition, the

present second-order algorithms constitute a key kernel in the development of efficient parallel simulation algorithms as discussed later in this section.

3. High-Fidelity Flexible Multibody Beams: The physically based control laws and CSI algorithms discussed above are in general applicable to linear or linearized second-order structural systems. The flexible multibody beam capability developed during the project can accommodate both large rigid and flexible motions without resorting to conventional assumed modes and their truncated approximations. As control laws for large three-dimensional rigid/flexible beam maneuverings are problem-dependent and scarcely available, emphasis has been placed instead on the development of high-fidelity, robustness aspects of the present beam models, instead. Applications of the present capabilities to cable hocking, three-dimensional double motions and similar problems indicate that the present three-dimensional beam capabilities accurately predict the cable hocking phenomena, one of the major failure modes for long space tethers, and preserve the conservation of energy for large repetitive flexible pendulum motion, an important requirement for robust nonlinear control strategies for reorientations of flexible spacecrafts. The present capability is now being utilized in the long-duration motion prediction (one or multiple-orbit libration perturbations) of spacecrafts, particularly the Space Station due to flexible manipulator movements, the Japanese science experimental spacecraft to be launched in 1994. In addition, plans are under way to utilize the present capability for the assessment of deployment aspects of 20 meter antenna, 20 km tether deployment and retrieval.

4. Parallel Computing for CSI Simulation: The second-order models of the structure and the state estimator, and the stabilized controller solver were implemented on a shared memory architecture so that the resulting software can either be run on a CRAY and Alliant machines. It has been demonstrated that, after a careful compiler optimization and parallelization, a speedup of about 30 to 35 can be obtained, with about 5 to 6 due to parallelization and 2 to 3 due to partial vectorization of the code. The present code has been installed at NASA Langley Center for the simulation of various preliminary CSI designs.

IV. Conclusions and Future Activities

The present project has achieved solid advances in three important areas that directly impact the simulation of dynamics and control of large space structures: 1) a physically based control law synthesis that can be interchanged to structural redesign parameters, and vice versa so that the designer can perform trade-off studies between control effort and structural redesign; 2) efficient second-order based CSI simulation algorithms that can be implemented on a spectrum of computers from a workstation as well as a shared-memory

machine; 3) finite element beam models of high-fidelity to physics of large rotations and suitable for implementation into flexible multibody dynamics simulations.

Of several possibilities, two most worthwhile aspects that can utilize the present research results are: 1) the development of maneuvering laws based on the present multibody capability that does not truncate response modes, and the investigation of the effects of using this more complete structural model on the motion perturbations during long-duration orbital motions; 2) the development of efficient parallel algorithms on local as well as massively parallel computers for the real-time simulation of large-scale flexible multibody-modeled spacecrafts undergoing a simultaneous maneuvering and vibration control. We intend to pursue these two aspects vigorously in the coming years.

The following theses are included separately:

1. **W. Keith Belvin**
Simulation and Interdisciplinary Design Methodology for Control-Structure Interaction Systems
2. **Janice D. Downer**
A Computational Procedure for the Dynamics of Flexible Beams Within Multibody Systems
3. **Jin-Chern Chiou**
Constant Treatment Techniques and Parallel Algorithms for Multibody Dynamic Analysis
4. **Luis A. Crivelli**
A Total-Lagrangian Beam Element for Analysis of Nonlinear Space Structures

**Implementation of a Partitioned Algorithm
for Simulation of Large CSI Problems**

K. F. Alvin and K. C. Park

Department of Aerospace Engineering Science and
Center for Space Structures and Controls
University of Colorado
Boulder, CO 80309-0429

March 1991

Report No. CU-CSSC-91-04

Research sponsored by NASA Langley Research Center
under grant NAG1-1021

Summary

This report summarizes research work on the implementation of a partitioned numerical algorithm for determining the dynamic response of coupled structure/controller/estimator finite-dimensional systems. The partitioned approach leads to a set of coupled first and second-order linear differential equations which are numerically integrated with extrapolation and implicit step methods. The present software implementation, ACSIS, utilizes parallel processing techniques at various levels to optimize performance on a shared-memory concurrent/vector processing system. The current work also generalizes the form of state estimation, whereby the Kalman filtering method is recast in a second-order differential equation equivalent to and possessing the same computational advantages of the structural equations. As part of the present implementation effort, a general procedure for the design of controller and filter gains is also implemented, which utilizes the vibration characteristics of the structure to be solved. Example problems are presented which demonstrate the versatility of the code and computational efficiency of the parallel methods is examined through runtime results for these problems. A user's guide to the ACSIS program, including descriptions of input formats for the structural finite element model data and control system definition, can be found in Appendix A. The procedures and algorithm scripts related to gain design using PRO-Matlab are included in Appendix B. In Appendix C, a stability analysis for the partitioned algorithm is presented which extends previous analysis to include observer dynamics, leading to a clearly definable stability limit. The source code for the parallel implementation of ACSIS is listed in Appendix D.

1.0 Introduction

The present work on the implementation of a partitioned transient analysis algorithm for the simulation of linear Control-Structure Interaction (CSI) problems has concentrated on four major areas. The initial software implementation emphasized the user-friendly aspect and a structural dynamics-oriented interface for experienced practitioners of finite element analysis programs. Another reason for a new implementation of the algorithm was that the initial architecture of the CS3 software testbed developed by Belvin and Park [1-2] had little provision for effective parallelization. CS3 also included extensive links to optimization and optimal control algorithms which were not central to the current work and proved to be a further hinderance. This new software implementation was designated ACSIS, for Accelerated Control-Structure Interaction Simulation, and led to significant improvements in speed for particular problems on conventional serial processors due to simpler and more economical storage of primary variables. ACSIS is also versatile in its usage of a general Timoshenko beam element with pin release capability and shear correction factor adjustment, and control system definition via a single data file. A preliminary Users Guide was developed for ACSIS, and the input formats were made compatible with pre/postprocessing software developed for Sun and Silicon Graphics computers so that

future versions using parallel techniques via element mesh domain decompositions can rely on the same X-Window-based I/O utilities. Table 1 documents runtime comparisons of ACSIS and CS3 on a sample 48 DOF problem simulated on a Sun 3/260 workstation using a floating point accelerator.

2.0 State Estimation via Second-Order Kalman Filtering

One restriction of the CS3 software testbed was the form of dynamic observer equations used in the partitioned algorithm as developed in [2]. However, Belvin and Park [3] showed that a general Kalman filtering type of state estimation was not only possible in the partitioned solution, but could be implemented at a very small additional cost in computations by a slight modification in the way the dynamical equations of the plant are cast into first-order form for filter gain design. The methods employed in [3] have been successfully implemented into ACSIS, thereby enhancing the code's ability to handle a wide variety of state estimation schemes. This is important as the application of optimal control techniques to the state estimator problem leads to this more general form, and as the restrictions of the observer used in CS3 typically meant either discarding part of the observer gain parameters, resulting in a loss in system performance.

ACSYS retains the option of using the more restricted observer form, as well as simple full state feedback (no dynamic compensation). It is clear from examining the respective equations, that for structures with stiffness-proportional damping, the only additional calculations required for the Kalman filter is the multiplication of the L_1 gain matrix by the predicted state estimation error vector γ (see equations 25 of [3]). This is not a significant portion of the computations required at each integration step, as the dimension of γ is small (number of sensors). By far the major costs are for computing an internal force of the form Kq , and backsolving the factored integration matrix for the estimated displacement states. This is verified numerically in the ACSIS results of Table 1 (using a restricted form of observer) and version (A1) of Table 2. The model used in Table 1 is roughly comparable to the 54 dof truss in Table 2, and both show the additional simulation time due to state estimation is roughly the same as an additional transient analysis. Finally, the Kalman filtering equations do not lead to any additional complications in parallel implementation of the overall algorithm as compared to the more restricted second-order observer developed for CS3.

3.0 Parallel Implementation of ACSIS

A primary emphasis in our work dealt with the optimization of the software implementation on a concurrent processing system. The platform chosen (primarily due to availability, initially at CU and later at NASA LaRC) was the Alliant FX/8 shared memory multiprocessor system with 8 parallel processing units and vectorization capabilities. Versions of the software have been ported to the Alliant and compiled using the Concentrix FX/Fortran optimizing compiler, which has available options for automatic vectorization and concurrency of standard, problem-independent parallel computations.

The partitioned CSI algorithm has three primary levels of parallelism in its numerics which can be exploited. At the highest level is the integration of the second-order dynamical plant (structure) and filter (estimator) equations, which as designed are of roughly equivalent size. Through the algorithm, these systems are effectively decoupled and independent at each discrete time step, and thus may be handled in parallel by invoking a compiler directive in the main program, which calls the respective subroutines simultaneously and handles re-synchronization of the execution upon their return.

At a lower level of the algorithm, the structure and state estimator equations exhibit symmetric, second-order forms typical of linear structural dynamics problems. It is well known that computations related to the formation of the internal force vector, Kq , can be re-implemented at an element level [4], which, through decomposition of the element mesh [5], can be handled in parallel within exclusive subdomains. This technique becomes particularly attractive for larger problems on more massively parallel systems; in the present work, the element-by-element (EBE) technique was not as effective as other types of parallelism. It should be noted here that the partitioned algorithm employs implicit integration methods, leading to systems of algebraic equations which are factored and solved using direct, rather than iterative, numerical methods. Therefore, formulation of the internal force vector is needed only in the formation of the known (right hand side) vector for integration of the plant and filter equations. The alternative to EBE computations is multiplication of the relevant displacement vector with the global stiffness matrix stored in profile (skyline) form. To "simulate" the advantages of local memory typical in large-scale parallel processing systems, the computed element stiffnesses were saved in shared memory for the EBE calculations, thus avoiding the need to recompute this data at each integration step. In addition, a low-overhead automatic element domain decomposition was provided for the parallel EBE method.

The final and lowest level of parallelism is obviously that of the basic matrix computations such as addition, multiplication, etc. These numerical operations are inherent in nearly all areas of the software implementation, including problem preprocessing. With the very capable optimizing compiler available on the Alliant system, this parallelism was exploited through vectorization and concurrency of the nominal source code using the FX/Fortran compiler run-time options `-O -DAS -alt`. The performance of the resultant executable code was examined using the Alliant's profiling capabilities, and changes to the nominal

source code, remaining compliant to F77, to maximize the identifiable concurrency and thus enhance the resultant speed. This was particularly useful in the profile matrix/vector multiply operation, whose speed is critical to the overall program performance.

4.0 Problem Descriptions

Three structural dynamics problems were developed for code testing at various levels of complexity. All three problems have the following common features: simulations consisted of 1000 integration steps and employed a stiffness-proportional damping. The damping was not needed for algorithm stability, but to ensure consistency between the examples and because the existence of damping in the plant equations has a strong influence on program speed. The Kalman filter models were of second-order form [2] and equivalent in size to their respective plant models. For controlled simulations, the control system began operating after 100 integration steps, and all gain matrices were full (i.e. all model states influence all actuators and are influenced by all sensors). Additional specific information for the problems follow.

A. Axial Vibration of Elastic Bar (Spring Model)

# Nodes:	3 free, 1 fixed
# Elements:	3
# Degrees of freedom:	3
# Actuators:	1
# Sensors:	1
Disturbance:	Initial displacement

B. Planar Vibration of Space Truss (Truss Model)

# Nodes:	18 free
# Elements:	33
# Degrees of freedom:	54
# Actuators:	4
# Sensors:	6
Disturbance:	Bang-bang type sinusoidal applied force

C. General 3D Vibration of EPS Satellite Reboost (EPS7 Model)

# Nodes:	97 free
# Elements:	256
# Degrees of freedom:	582
# Actuators:	18
# Sensors:	18
Disturbance:	Bang-bang type square wave applied force

As can be seen, the problem sizes are roughly three different orders of magnitude, with corresponding increases in the sizes of the control systems. Appendix B includes routines using Matlab for the design of control and filter gains which were used for the control system design of all three example problems. An illustration of the EPS model is shown in Figure 1.

5.0 Performance Assessment

Table 2 compares CPU runtimes on the Alliant computer (using the UNIX "time" command) for distinct versions of the software. Version (A1) is the nominal F77 program code compiled without any performance-enhancing options, while version (A2) invokes automatic vectorization and concurrency of low-level, problem-independent computations such as vector addition and inner products. The performance improvements are significant, especially for the large EPS7 model, where the speed-up factor is 35-37.

Version (A3) also uses the compiler options from (A2), but in addition has a compiler directive added to the main program which allows the plant and filter integration subroutines to be called in parallel. This does not affect the transient response results as that analysis option bypasses the altered code, but for controlled response there is some effect on performance. If filtering is used, which results in a significant increase in computation, the directive can lead to some increased speed as can be seen for the spring and truss problems. There can also, however, be a reduction in performance as compared to (A2) if the finite amount of processors and vector units are used in a less efficient way. This appears to be the case for the large EPS problem, where the "overhead" introduced by the directive, and its effect on processor assignment, is greater than the improvement generated by the manually-invoked parallel construct.

Table 3 shows CPU run times for ACSIS using E-B-E computations, which, as mentioned previously did not lead to better performance on the example problems using the Alliant system. This appears to be due to the lack of effective vectorization of the individual element computations when forming the internal force via the EBE method. To determine whether the EBE method effectively lead slower speeds through increased numbers of computations, version (A2) (see above) was altered by removing compiler optimization of the profile matrix/vector multiply routine (the alternate method to EBE). The resultant runtimes matched almost exactly with those of version (A4), leading to the conclusion that both methods require roughly the same amount of computations, but differ in how they can be optimized on the Alliant system. The matrix/vector multiply operation, in this environment, can exploit both vectorization and concurrency through the compiler's performance options; this can be examined in the compiler output. The EBE computations, at the element level, do not vectorize because the parts of the global displacement and force vectors being operated on per element are not contiguous. In version (A5), the elements within each subdomain of the mesh are computed in parallel, leading to some performance

improvement over version (A4), but because of the lack of element level vectorization, this version is less efficient than (A2), which uses a compiler-optimized matrix/vector multiply. Finally, version (A6) runs parallel operations at both the structure/observer and element force level, but this amount of process branching tends to degrade performance for the small number of processors available on the Alliant.

Overall, versions (A2) and (A3) provide the best code performance for the hardware available. Parallelizing the observer and structure (A3) leads to mixed results; improvement for the small spring and truss problems, but not for the large EPS model. Element-by-element computations do not improve code performance over compiler optimization via vectorization and concurrency for this platform. Reimplementation of the algorithm lead to a 5:1 improvement over the CS3 testbed software on a serial computer (Table 1). Further optimization of ACSIS on the Alliant FX/8 lead to an additional 30:1 improvement in runtimes for large-order systems such as the 582 dof EPS model. Time history responses of selected variables for the example problems are shown in Figures 2 through 10. For Figures 8 through 10, the u_x, u_y, u_z displacements plotted are located at node 45, which is located at the vertex of the large antenna of the EPS model (see Figure 1).

6.0 Conclusions

The present work has demonstrated the efficiency of a streamlined simulation code for the analysis of large-order CSI systems and the viability of the continuous second-order Kalman filtering equations for state estimation. These methods show the versatility of the partitioned CSI integration algorithm and the promise of its application to real-time simulation. It is evident, however, that the use of element-by-element computational technique requires the development of innovative algorithms for effective implementation on massively parallel processing systems. Future work in this area will include integrating algorithms for on-line system identification and applying these capabilities to the problem of real-time control.

Simulation	CS3	ACSIS
Transient	439.2	98.8
Full State Feedback	688.2	181.5
FSFB with Observer	1156.7	282.3

Table 1: Comparison of Runtime Speeds for CS3 and ACSIS
on a Sun 3/260 System

Model	Problem Type	(A1) Nominal Code	(A2) Compiler Optimized	(A3) Parallel Observer
3 DOF Spring	Transient	6.6	2.1	2.1
	FSFB	8.0	3.3	3.3
	K. Filter	12.3	3.5	3.3
54 DOF Truss	Transient	78.2	5.7	5.6
	FSFB	97.1	9.4	10.2
	K.Filter	170.7	13.0	10.7
582 DOF EPS7	Transient	3506.	98.6	100.3
	FSFB	7040.	190.2	294.5
	K. Filter	n/a	284.2	312.5

Table 2: CPU Results for Versions of ACSIS

Model	Problem Type	(A4)	(A5)	(A6)
		E-B-E Computation	Parallel E-B-E	Parallel Obs. & EBE
3 DOF Spring	Transient	3.8	3.3	3.3
	FSFB	4.9	4.4	4.9
	K. Filter	6.6	5.6	5.0
54 DOF Truss	Transient	31.7	13.0	13.0
	FSFB	35.5	16.9	35.6
	K.Filter	62.6	27.3	36.2
582 DOF EPS7	Transient	391.7	153.9	n/a
	FSFB	485.9	245.9	n/a
	K. Filter	n/a	n/a	n/a

Table 3: CPU Results for ACSIS with EBE Computations

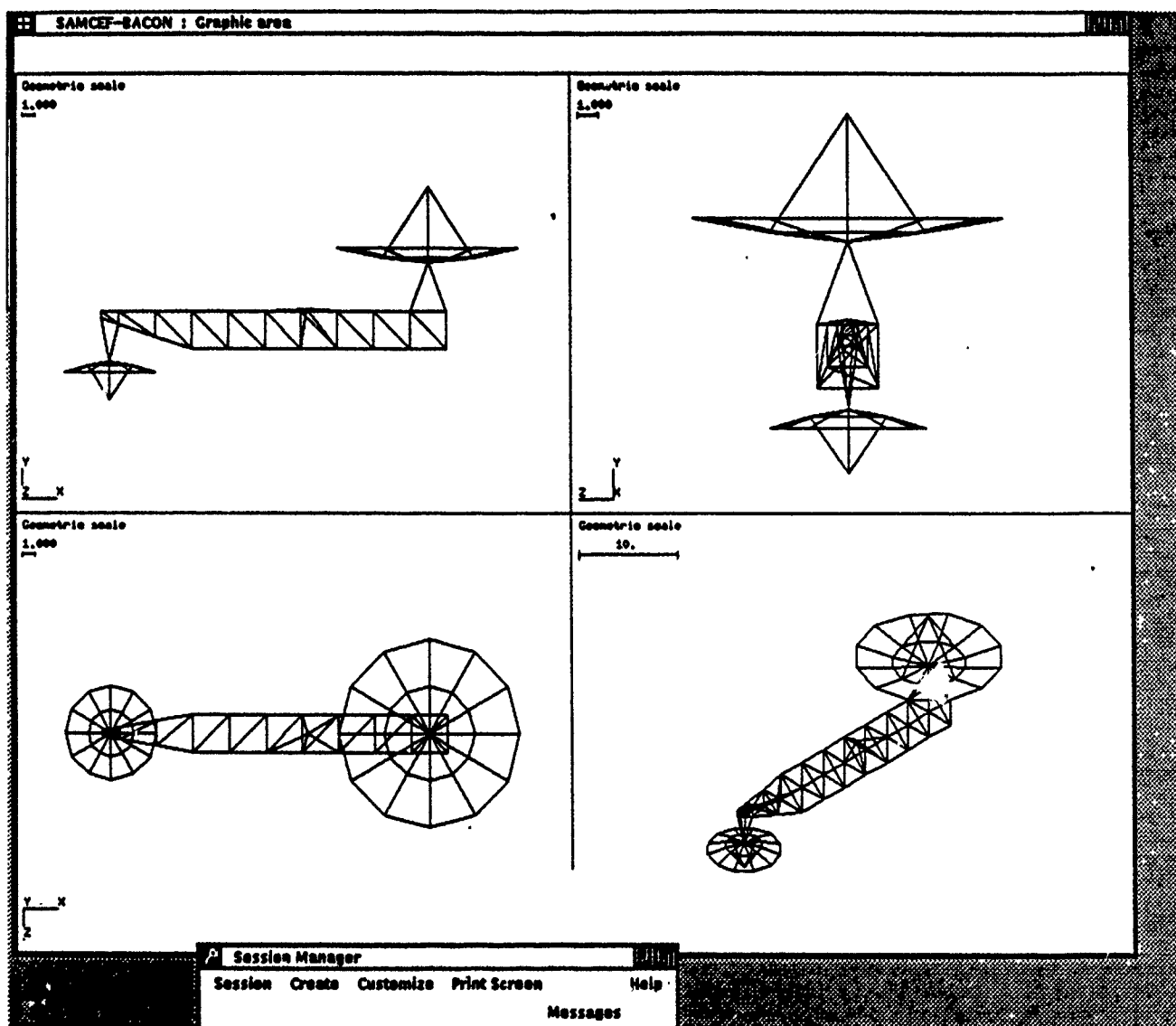


Figure 1: EPS Finite Element Model

Spring Model: Open Loop Transient Response

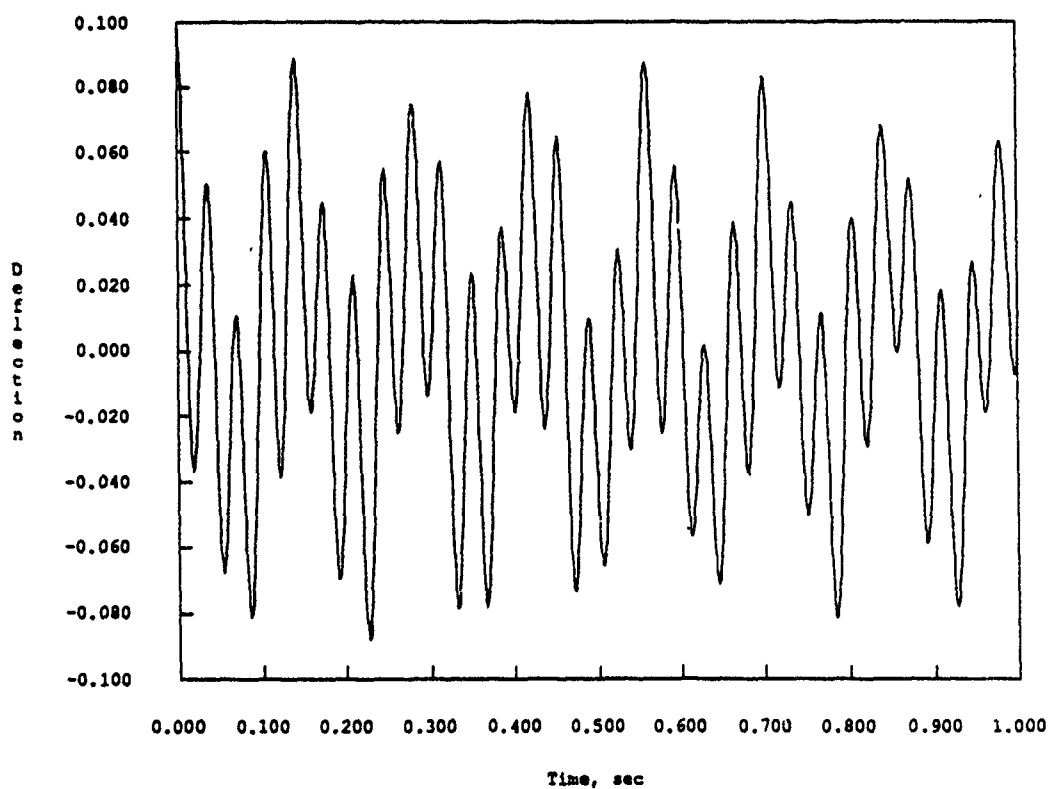
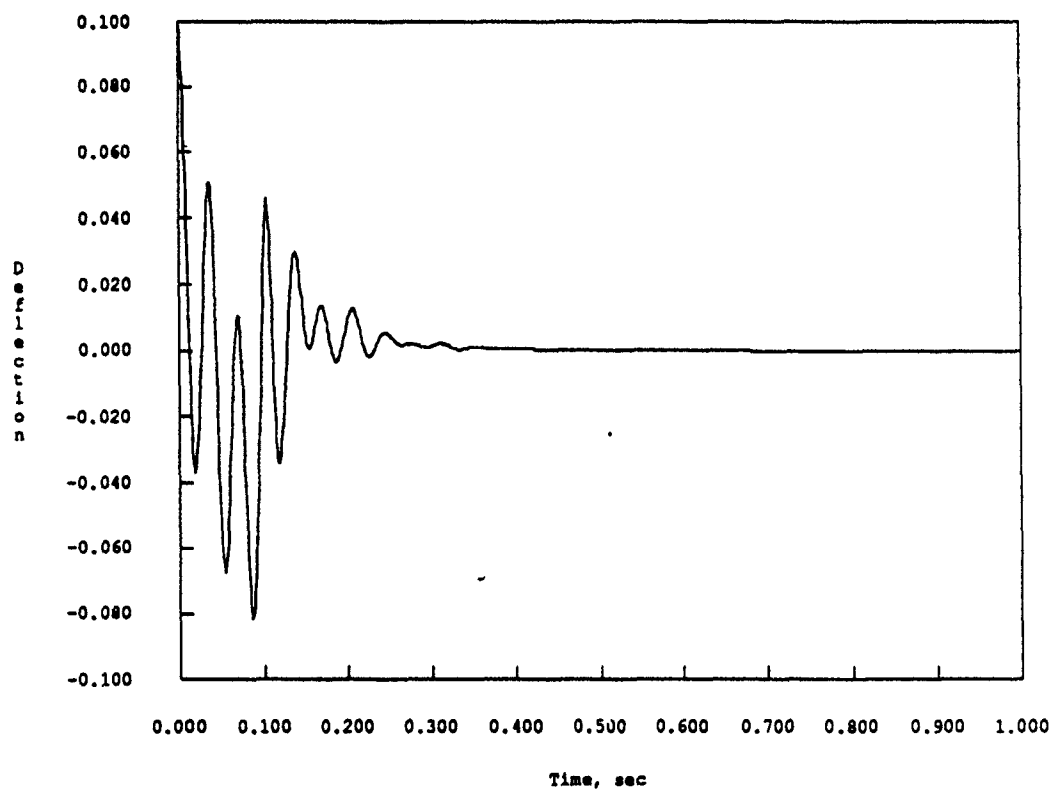


Figure 2: Spring Transient Response

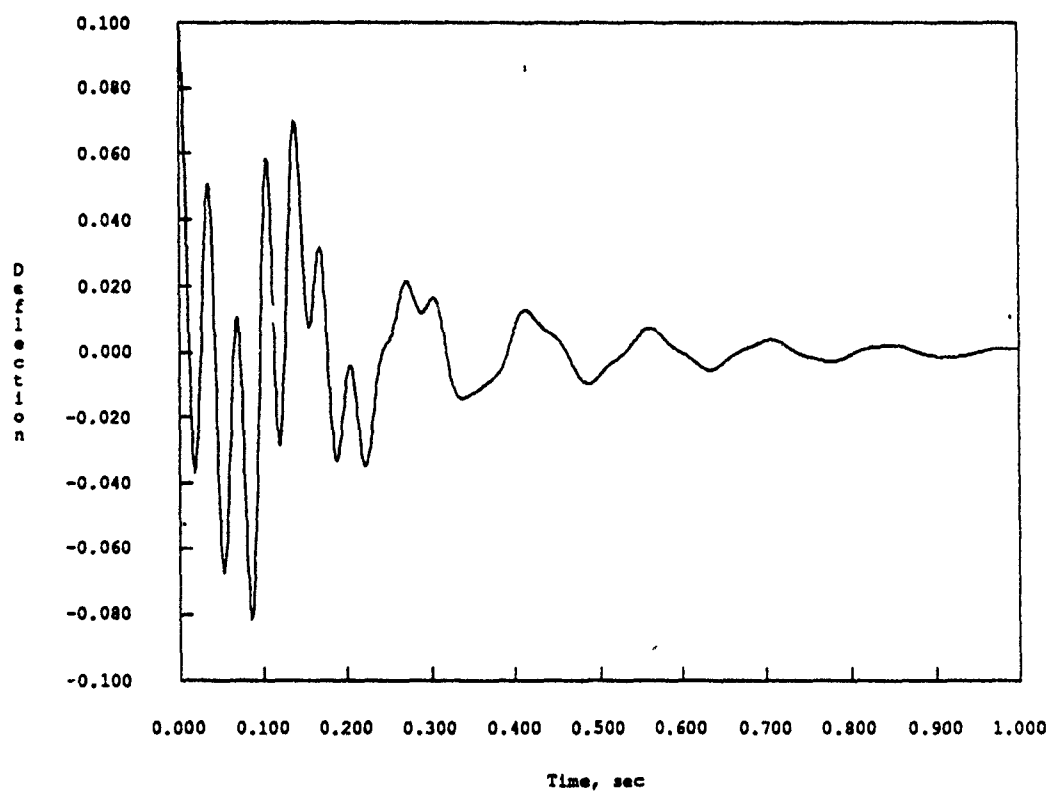
Spring Model: Full State Feedback Response



— Node 3, ux

Figure 3: Spring FSFB Response

Spring Model: Controlled Response w/Kalman Filter



— Node 3, ux

Figure 4: Spring Response w/Filter

Truss Model: Open Loop Transient Response

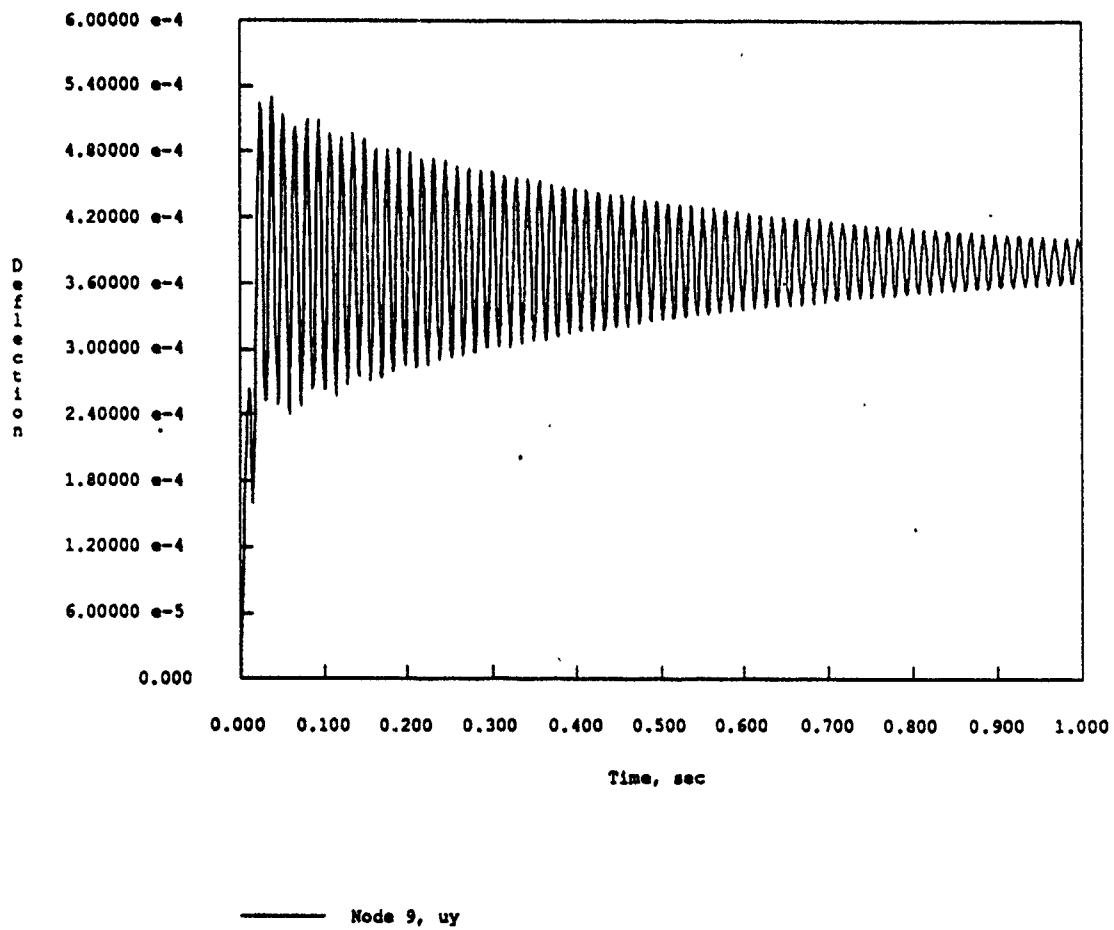
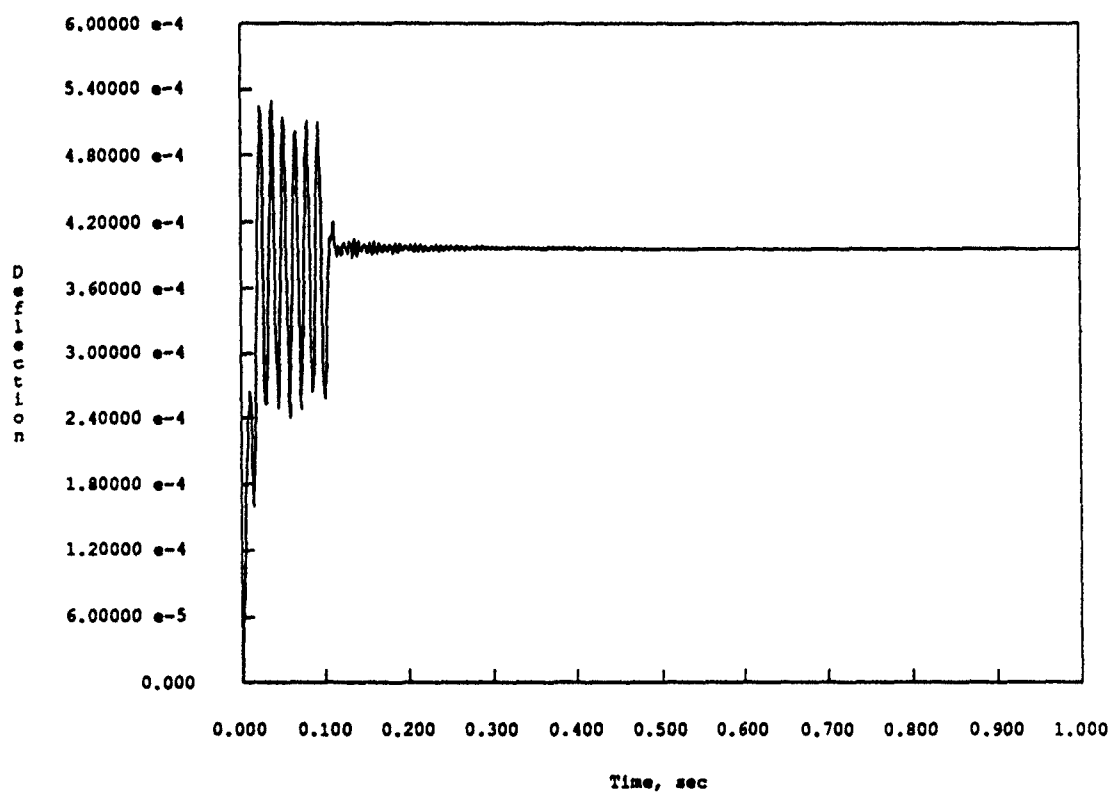


Figure 5: Truss Transient Response

Truss Model: Full State Feedback Response



Node 9, uy

Figure 6: Truss FSFB Response

Truss Model: Controlled Response w/Kalman Filter

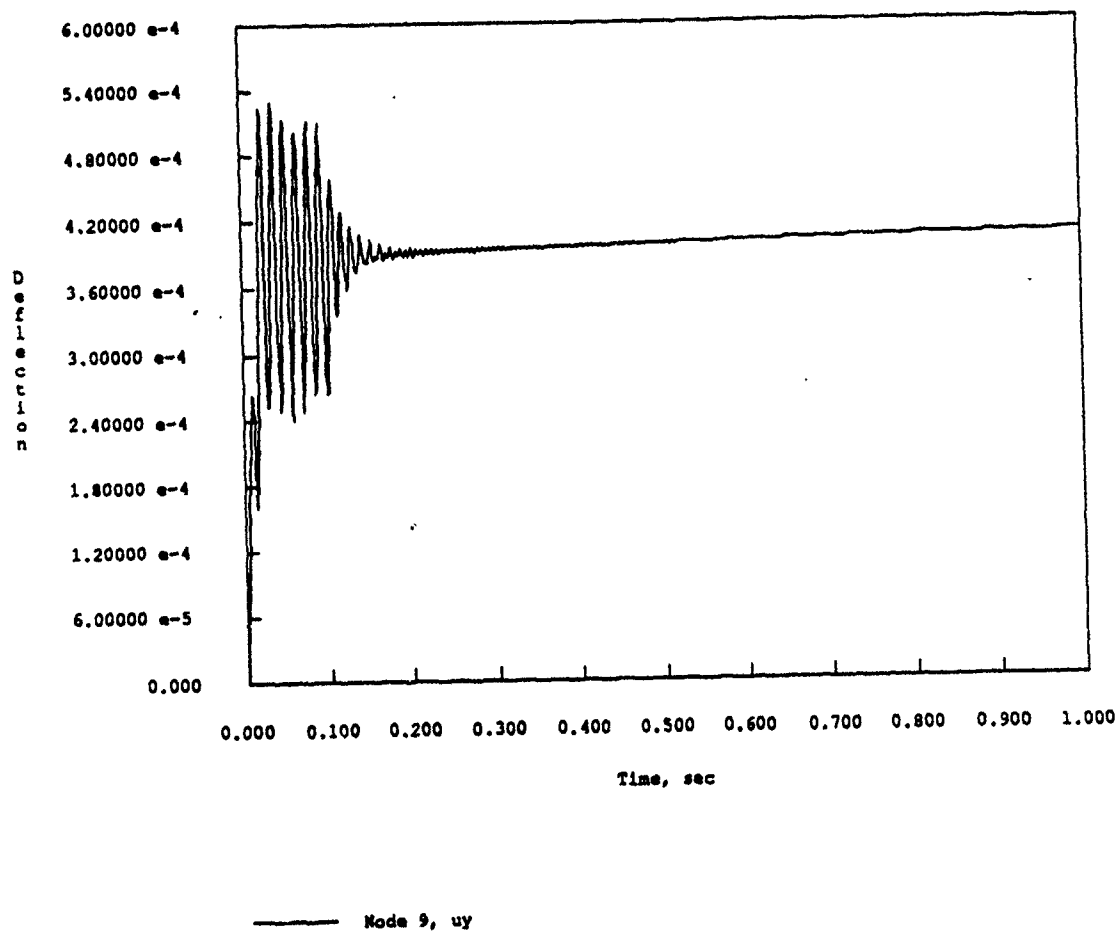


Figure 7: Truss Response w/Filter

EPS7 Model: Open Loop Transient Response

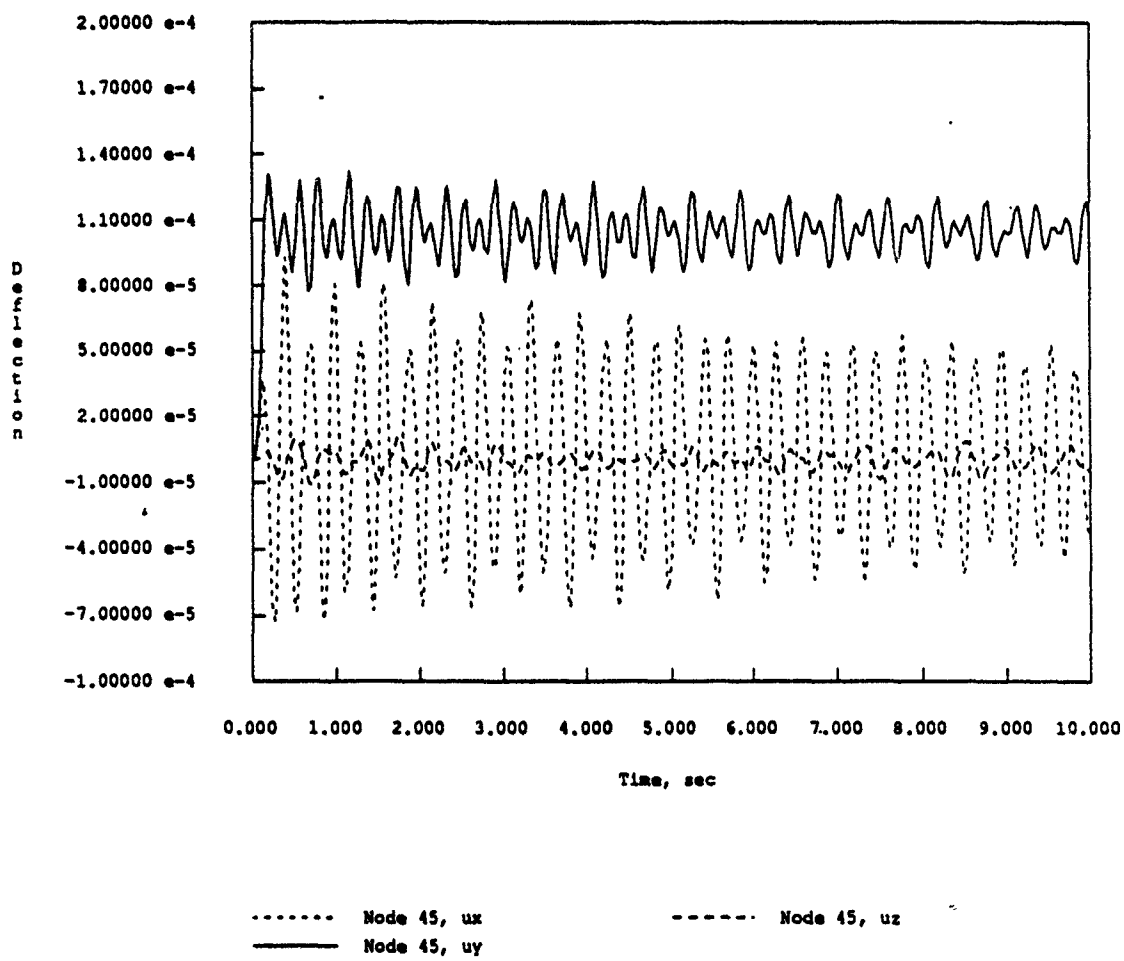
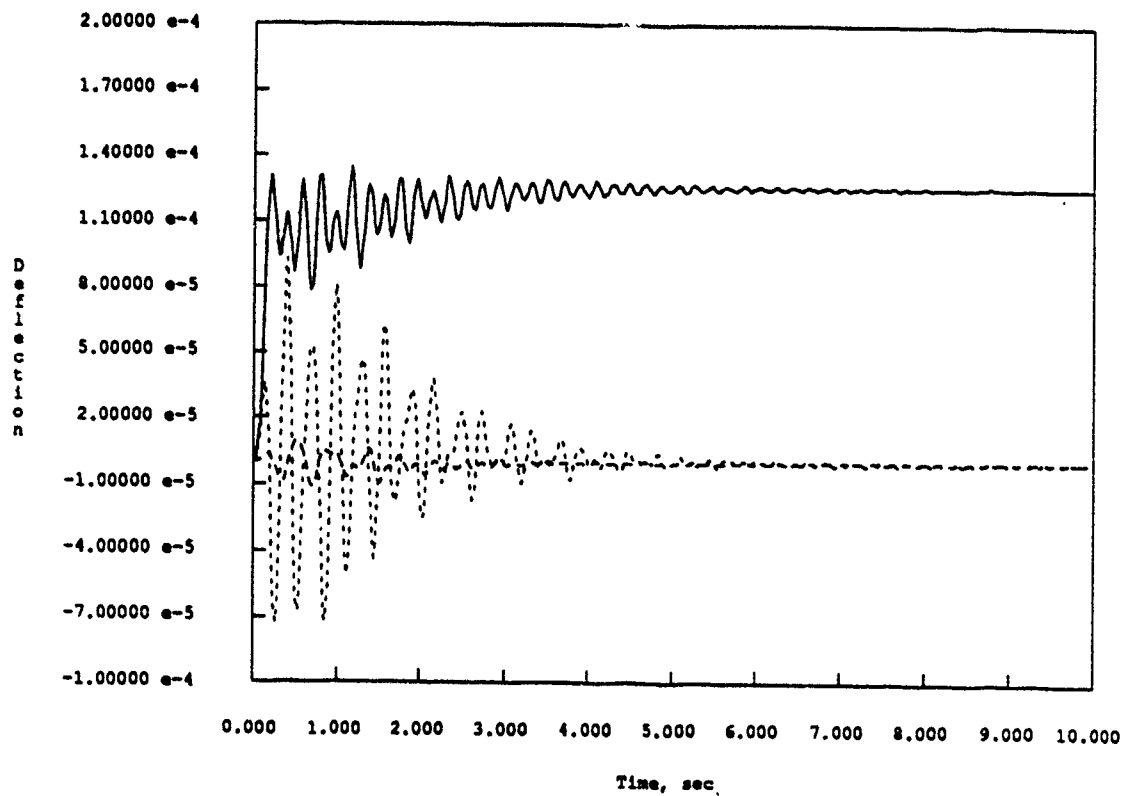


Figure 8: EPS Transient Response

EPS7 Model. Full State Feedback Response



..... Node 45, ux
 ----- Node 45, uz
 _____ Node 45, uy

Figure 9: EPS FSFB Response

EPS7 Model: Controlled Response w/Kalman Filter

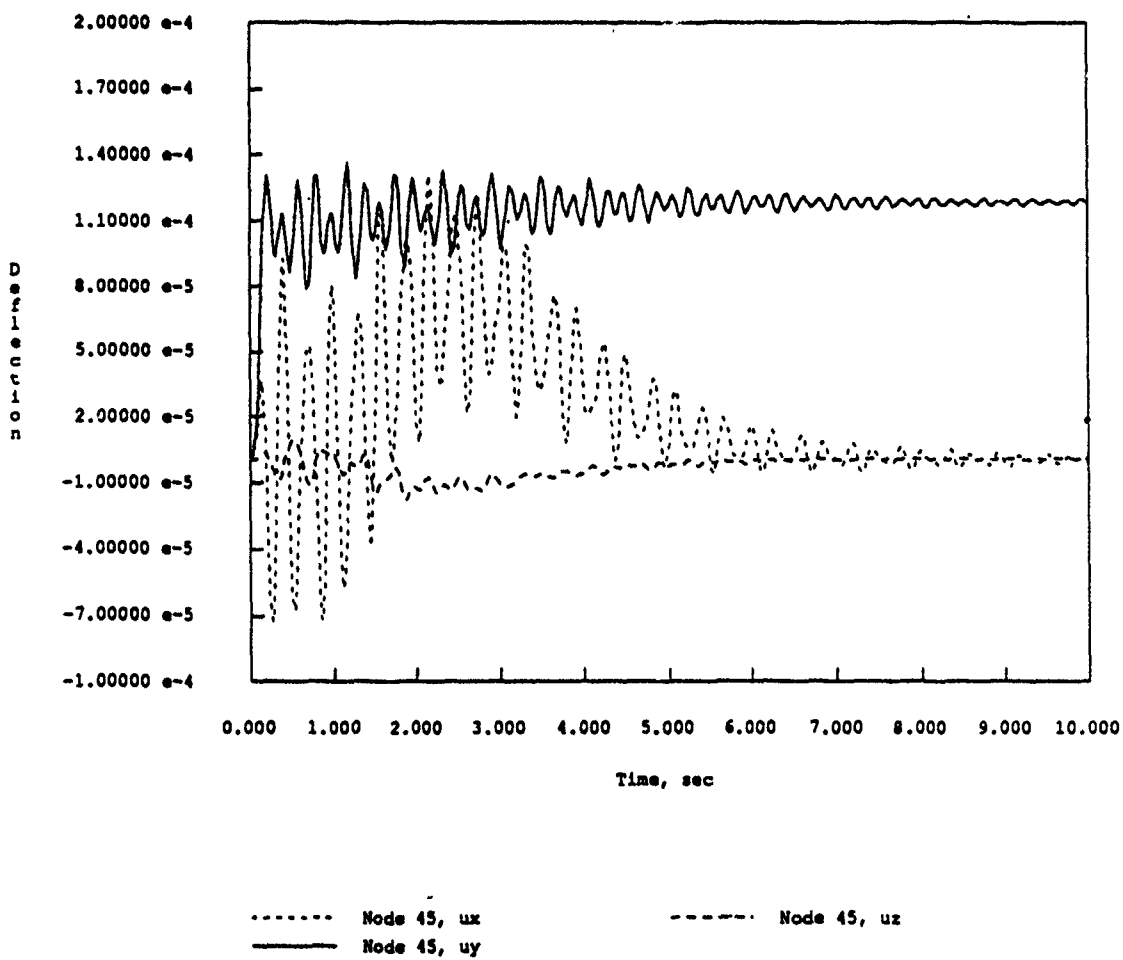


Figure 10: EPS Response w/Filter

REFERENCES

1. Belvin, W. K., "Simulation and Interdisciplinary Design Methodology for Control-Structure Interaction Systems," PhD Thesis, Center for Space Structures and Controls, University of Colorado, Report No. CU-CSSC-89-10, July, 1989.
2. Park, K. C. and Belvin, W. K., "Partitioned Procedures for Control-Structure Interaction Simulations," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 1, Jan.-Feb. 1991, pp. 59-67.
3. Park, K. C. and Belvin, W. K., "Discrete Integration of Continuous Kalman Filtering Equations for Time-Invariant Second-Order Structural Systems," Report No. CU-CSSC-89-08, Center for Space Structures and Controls, University of Colorado, April 1989 (Submitted for publication in *J. Guidance, Control and Dynamics*).
4. Farhat, C. and Crivelli, L., "A General Approach to Nonlinear FE Computations on Shared Memory Multiprocessors," *Comp. Methods in Applied Mech. Eng.*, Vol. 72, No. 2, pp. 126-152 (1989)
5. Farhat, C., "On the Mapping of Massively Parallel Processors onto Finite Element Graphs," Report No. CU-CSSC-88-02, Center for Space Structures and Controls, University of Colorado, April 1988

APPENDIX A

AC SIS User's Manual

**Accelerated
Control
Structure
Interaction
Simulation**

Introduction

ACSIS is an analysis program for full-order simulation of control-structure interaction (CSI) problems. The CSI simulation is carried out using a partitioned analysis procedure which treats the structure (or plant), the observer, and the controller/observer interaction terms as separate entities. This procedure allows ACSIS to maintain relatively small, sparse matrix equations when compared to the process of assembling the computational elements into a single set of equations of motion and solving simultaneously. Although this software can also carry out modal analysis, the transient response and CSI simulation are done in real space using the entire finite element model. (For more information, see Ref. 2 of report, p. 14)

The ACSIS program is run using two previously prepared data input files and interactive input of run options. The two data input files are the finite element input file and the controller definition file.

Interactive Options

The run options are as follows, note that no defaults exist, data must be input each time it is requested for each item requested except during a background run. Files may be given any names, example names are given only to match the truss example at the back of this manual.

Please input analysis type:

This option is for selecting modal analysis, CSI simulation or the transient response of a structure. Not all interactive inputs are required for each analysis type.

Do you wish to save an input file? (y or n)

This option creates a file which saves all the interactive input options. ACSIS can be background run with minor option changes by editing this file and then directing the screen input to this file. This file is very different for each analysis type. (To do this run with the example names after running acsis.exe once interactively, the command would be: acsis.exe <INP_truss> &.

Name of save input file? (filename)

This question asks for the name under which to save the interactive input. example name: INP_truss

Finite Element Model Input File Name (filename):

This file should contain all the finite element nodes, mesh, materials, properties, lumped inertias, fixations, and initial velocity and displacement conditions. It must be prepared in advance in the card format specified later. example name: FEM_truss

Number of modes desired?

This only appears in the modal analysis to request the number of modes to be output. The actual analysis is carried out with double this number of modes for accuracy.

Controller Definition File Name:

This only appears for the CSI simulation. The file should contain the actuator and sensor locations, control gains, and observer gains. It also must be prepared in advance in card format. example name: CON_truss

Please input type of control::

This option is for selecting the form of control law equations used in the CSI simulation. Full state feedback uses the current states of the plant (structure) to determine the control via constant gain matrices. Second-order observer uses only the L2 filter gain matrix of a Kalman filtering type of state estimation where the state variables are position and velocity. The Kalman filter option allows the use of a full set of filter gains, but the gain design must come from a alternate variable casting using position and generalized momenta.

Initial time, final time, control-on time, step size:

Data format is four columns for CSI, but only three columns for transient response since the control-on time part is deleted.

Forcing function ID, scale factor, damping coeff- a,b:

In order for any time dependent forcing function to be easily implemented despite variable time step sizes, the forcing functions must be entered into the subroutine forces.f. The format is to assign each new forcing function an ID number in a elseif statement. An example of forces.f is included at the back of the manual. Then forces.f must be recompiled into an object file and acsis.exe relinked. This is easily done by typing make which detects changes to the fortran files and does only the necessary compiling. A particular forcing function is chosen by entering its ID number in the first column, in addition the force can be scaled by a constant using the scaling factor in the second column. The Rayleigh damping coefficients a and b are the third and fourth columns.

Phase lag fix? (y or n):

Specifies whether to include an extra iteration of control and sensor state prediction at each integration step to improve accuracy. Not generally needed unless the user is investigating the source of instability in a simulation and needs to test the sensitivity of the response to the partitioned algorithm's extrapolation method.

Gain scale factors (4 total):

These scaling factors are, in order: the F1 control gain matrix (displacement), F2 control gain matrix (velocity), L1 and L2 state estimator filter gain matrix. This question appears only for CSI.

ACSIS has two types of output options. The first is the displacement or velocity motion of up to twenty separate degrees of freedom. Interactive questions are, 'Number of displacement results to output (max 10)' followed by as many 'Input node #, dof for displacement output #' as necessary. Then these repeat for velocity results. The name of the file where the output will be stored is requested by 'Output file name? (filename).' example name: OUT_truss. The second output option saves the displacement of all nodes at any time step where output is sent (see next entry). The format is suitable for animation of the entire structure. Question asks 'Animation Output? (y or n)' then for an animation filename if necessary.

Send output every how many steps?

This option affects both output options to reduce the size of the output and animation files. It causes output to only be sent after a integer number of time step iterations. To get output each time step, simply enter 1.

Finite Element Input File

The finite element input file consists of title cards followed by columns of data. The title cards can be in any order, but they must be all capitalized with the appropriate number of columns of data for each card. The program reads rows of data until encountering a new card. Data which represents an integer value may be entered with a decimal point while real data may be entered without a decimal point as necessary. Any line beginning with a * anywhere in the file is ignored and can be used to insert comments. Any blank line will result in a read error.

NODES

Each node must be defined on a separate line. Columns can be separated by any number of spaces or a comma. Data format is four columns: node number, x-coordinate, y-coordinate, z-coordinate.

TOPOLOGY

Each element must be defined on a separate line. Truss elements require two nodes then two columns of zeroes. (Truss elements would also require pin releases in ATTRIBUTES below.) Beam elements require two nodes then a third reference node representing a point in the xz plane of the beam and then a column of zeroes. Element type refers to finite element formulation. (Currently only type 1 = timeshenko beam element is now available.) Data format is six columns: element number, element type, node #1, node #2, node #3, node #4.

ATTRIBUTES

Each element is characterized by an ID number from each of the MATERIAL and PROPERTIES cards below. Each element also has six pin release codes. The first code is for longitudinal stiffness, the second code is for torsional stiffness, the third and fifth codes are bending stiffness at each end in the y direction and must be the same value, and the fourth and sixth codes are for bending stiffness in the z direction and also must be the same. (0 is stiff, 1 is released.) Data format is nine columns: element number, material type, property type, and six pin release codes.

MATERIAL

The material data is formatted in four columns: material type, Young's modulus, shear modulus and density of the material.

PROPERTIES

The properties data is formatted in six columns: property type, cross sectional area of the element, I_y , I_x , I_y , I_x , shear shape factor SSF2, shear shape factor SSF3.

FIXITY

Nodes with any fixations are defined here in the finite element file. The nodes must be entered with the fixity of all six of their DOF's, restrained or not. (1 is restrained, 0 is free) Data format is seven columns: node number, x, y, z, ϕ_x , ϕ_y , ϕ_z .

INERTIA

All lumped inertias must be entered with each separate DOF on an individual line. Therefore a single node could take up to six lines to define. Data format is three columns: node number, DOF number (1-6), and value of inertia.

INITIAL CONDITIONS

All initial displacement and velocity conditions are entered into the finite element file. Data format is four columns: node number, DOF number (1-6), initial displacement, and initial velocity.

END

End of file.

Controller Definition Cards

The controller definition file also consists of title cards followed by data entry. Each card must be followed by the appropriate number of columns of data and in some cases the appropriate number of rows. Integers can be entered in real format and vice versa if necessary. Any blank line will result in a read error.

NACT

Number of actuators in the entire control system.

BMAT

This entry creates the actuator position matrix or B matrix. There should be one row for each actuator, data format is four columns: node number, DOF number (1-6), actuator number, and sensitivity.

NSEN

Number of sensors in the entire controls system.

HDMA

This entry creates the matrix of displacement sensor locations. One row per displacement sensor, data format is four columns: node number, DOF number (1-6), sensor number, and sensitivity.

HVMA

This entry creates the matrix of velocity sensor locations. One row per velocity sensor, data format is four columns: node number, DOF number (1-6), sensor number, and sensitivity.

F1GA

This is a list of the F1 or displacement control gains. The data format is four columns: node number, DOF number (1-6), actuator number, and value of gain.

F2GA

This is a list of the F2 or velocity control gains. The data format is four columns: node number, DOF number (1-6), actuator number, and value of gain.

L1GA

This is a list of the state estimator L1 filter gains. The data format is four columns: node number, DOF number (1-6), actuator number, and value of gain.

L2GA

This is a list of the state estimator L2 filter gains. The data format is four columns: node number, DOF number (1-6), actuator number, and value of gain.

END

End of file.

Examples

This section includes all the files and procedures needed to run all three analysis on a simple elastic bar problem. The naming of the files is a simple and easy to remember system, however no particular format is necessary.

The finite element file was created simply by typing the node locations, connectivity (topology), etc. with a text editor.

File: FEM.spring

```
*
MESH
*
NODES
  1  0.00 0.00 0.00
  2  1.00 0.00 0.00
  3  2.00 0.00 0.00
  4  3.00 0.00 0.00
TOPOLOGY
  1  1  1  2  0  0
  2  1  2  3  0  0
  3  1  3  4  0  0
ATTRIBUTES
  1  1  1  0  1  1  1  1  1
  2  1  1  0  1  1  1  1  1
  3  1  1  0  1  1  1  1  1
MATERIAL
  1 1000.  0.0  0.0
PROPERTIES
```

```

1 1.00 0.00 0.00 0.00 0.0 0.0
FIXITY
1 1 1 1 1 1 1
2 0 1 1 1 1 1
3 0 1 1 1 1 1
4 0 1 1 1 1 1
INERTIA
2 1 0.100
3 1 0.100
4 1 0.100
INITIAL
3 1 0.100 0.000
END

```

The modal analysis only requires the number of modes desired in addition to the finite element file. The file INP_spring0 documents the interactive inputs used in the modal analysis. The results are saved in file EIG_spring.

File: INP_spring0

```

-1
n
    * ACSIS input file,two lines above are
    * analysis type and save input file. Do
    * not change them by editing this file.
    * Finite element input file?(filename)
FEM_spring
    * Number of modes desired?
    3
    * Output file?(filename)
EIG_spring

```

File: EIG_spring

SUBSPACE ITERATION ROUTINE

```

NB OF EIGENVALUES=    3
NB OF VECTOR=        3
NB OF DOF=           3
TOLERANCE= 1.000E-04

NB OF RIGID MODES=    0

```

ITERATION NO 1

1.000E+00 1.000E+00 1.000E+00
 ITERATION NO 2
 1.297E-14 3.194E-14 3.899E-14
 EIGEN ANALYSIS RESULTS:

MODE	EIGENVALUE	RADIAL FREQUENCY	CYCLIC FREQUENCY
----	-----	-----	-----
1	1980.6	44.504	7.0831
2	15550.	124.70	19.846
3	32470.	180.19	28.679

EIGENVECTORS:

1	2.3305	1.8689	-1.0372	0.	0.
2	1.8689	-1.0372	2.3305	0.	0.
3	1.0372	-2.3305	-1.8689	0.	0.

MASS MATRIX DIAGONAL:

2	1	3	1.00000000000000D-01
3	1	2	1.00000000000000D-01
4	1	1	1.00000000000000D-01

To run the transient response of the structure, a forcing function or initial condition would be needed to excite the structure. An initial condition would be added to the finite element file. A forcing function must be added to forces.f with a new ID number, then this number given as interactive input. In this case, an initial displacement acts on the second degree of freedom, which is defined in the finite element model input file. The file INP_spring1 documents the interactive inputs used in the transient analysis.

File: INP_spring1

-3

n

- * ACSIS input file,two lines above are
- * analysis type and save input file. Do
- * not change them by editing this file.
- * Finite element input file?(filename)

FEM_spring

- * Initial, final, step size?
- 0.00000000 1.00000000 0.00100000
- * Forcing function,scale f, damping a,b?
- 0 0.000000 0.00000000 0.00002000
- * Output file name?(filename)

OUT_spring

- * Number of displacement outputs?
- 1
- 3 1
- * Number of velocity outputs?

```

0      * Send output every how many steps?
1      * Send animation output?(y or n)
n

```

In addition to the finite element file, the interactive input, and an excitation, CSI simulation requires a controller definition file. A full state feedback controller for the truss structure is defined in CON_spring.

File: CON_spring

```

NACT
1
BMAT
3 1 1 1.0
F1GAIN
2 1 1 193.314150000000
3 1 1 1574.63150000000
4 1 1 -1669.04600000000
F2GAIN
2 1 1 20.7742560000000
3 1 1 27.7904030000000
4 1 1 10.7802120000000
NSEN
1
HVMAT
3 1 1 1.0
L1GAIN
2 1 1 8.79289090000000D-02
3 1 1 -4.88079010000000D-02
4 1 1 3.01027350000000D-03
L2GAIN
2 1 1 1.03809320000000D-01
3 1 1 6.14101300000000
4 1 1 -3.06087250000000
END

```

Then if acsis.exe is run interactively and the input is saved, the file INP_spring2 can be produced. The file could be edited to change the input files, the output file, the forcing function ID, the length of simulation, control-on time, etc. Then to run it again, type acsis.exe<INP_spring2> scr &. The scr is a scratch file which will store the screen output.

File: INP_spring2

```

n
    * ACSIS input file,two lines above are
    * analysis type and save input file. Do
    * not change them by editing this file.
    * Finite element input file?(filename)
FEM_spring
    * Controller file name?(filename)
CON_spring
    * Please input type of control:
      3
    * Initial,final,control-on,step size?
0.00000000  1.00000000  0.10000000  0.00100000
    * Forcing function,scale f, damping a,b?
0      0.000000  0.00000000  0.00002000
    * Phase lag fix?(y or n)
n
    * Gain scale factors (4 total)?
1.00000000  1.00000000  1.00000000  1.00000000
    * Output file name?(filename)
OUT_spring
    * Number of displacement outputs?
1
      3      1
    * Number of velocity outputs?
0
    * Send output every how many steps?
1
    * Send animation output?(y or n)
n

```

APPENDIX B

Matlab Procedure and Scripts
for Controller and Kalman Filter
Gain Designs

Introduction

In order to retain simplicity in the ACSIS code for parallel implementation purposes, no control system design algorithms were included. Instead, using modal data output from the eigenmode analysis module of ACSIS, a procedure was developed using the Pro-Matlab and its Control System Toolbox, which includes algorithm scripts for optimal control solutions via the solution of an algebraic Ricatti equation. In order to accomodate large order dynamical systems, the design is accomplished in the uncoupled normal modes domain, using the available lowest eigenmodes from ACSIS.

The procedure begins by copying and editing the mode data output from ACSIS (see the listing for EIG_spring in Appendix A) into readable variables for input to Matlab. The typical approach was to create one file as a Matlab script (i.e. a ".m" file), with the eigenvalues, eigenvectors, and mass/dof data from ACSIS at the beginning, followed by actuator and sensor influence matrices (related to the physical degrees of freedom), the objective function weighting matrices, and the function which calls other Matlab scripts to determine the solution. An example of the above input for the spring problem described in Appendix A is in file EIG_spring.m, which is listed below. Compare this the the ACSIS mode data output shown in Appendix A to see how the editing was accomplished, and the additional control design data added

File: EIG_spring.m

```
lam = [ 1980.6
        15550.
        32470. ];
v1=[1 2.3305 1.8689 -1.0372 0. 0.
     2 1.8689 -1.0372 2.3305 0. 0.
     3 1.0372 -2.3305 -1.8689 0. 0. ];
m=[ 2 1 3 1.0000000000000D-01
    3 1 2 1.0000000000000D-01
    4 1 1 1.0000000000000D-01];
t=[v1(:,2:4)];
qb=zeros(3,1);
qb(2,1)=1.0;
hd=zeros(1,3);
hv=zeros(1,3);
hv(1,2)=1.0;
qv=[1;.5;.1];
q=diag([qv;qv]);
r=.0001*eye(1);
[f1,f2]=mlqr(lam,t,m,qb,q,r,3);
qv=[1;1;1];
q=diag([qv;qv]);
r=100*eye(1);
[k1,k2]=mkf(lam,t,m,hd,hv,q,r,3);
save f1out f1 /ascii
save f2out f2 /ascii
```

```
save k1out k1 /ascii
save k2out k2 /ascii
```

The scripts `mlqr.m` and `mkf.m` were written to accept as input the vector of eigenvalues, the eigenvector matrix (orthogonal vectors stored in columns), a matrix of node, component, d.o.f, mass data, and the actuator (or sensor) influence matrix and weighting matrices. The scripts output the gain results in four-column arrays, with one gain per row, and the corresponding node number, displacement component, and actuator (or sensor) identification. The top-level problem script (listed above) then saves the output in external files and the analysis is complete. The design scripts (listed below) also include checks on controllability and observability of the system based on the modal data and influence matrices defined, and produce plots of the closed loop poles resulting from the gain design. This aids the analyst in assessing the expected performance (and stability) of the exact system before moving the data back to ACSIS for simulation. The script `contrank.m` finds the rank of the controllability matrix through iterative rank calculations of submatrices so as to avoid the illconditioning experienced in the full matrix. This is both faster and more accurate for determining whether a particular actuator placement has full control of the included structural modes.

File: `mlqr.m`

```
function[F1out,F2out]=mlqr(lam,t,m,qb,Q,R,nmode)
%
% Controller gain design for second-order
% structural system via given eigenmodes.
% Gains are transformed to be coefficients
% of structural variables (disp,velocity);
% i.e. plant is second-order, of size ndof.
%
% Arguments:
%
% lam: vector of eigenvalues (nmode x 1)
% t: matrix of eigenvectors (ndof x nmode)
% m: mass diagonal and dof mapping info (ndof x 4)
% qb: actuator position influence matrix (nact x ndof)
% Q: optimal design state weighting matrix (2*nmode x 2*nmode)
% R: optimal design feedbk weight. matrix (nact x nact)
%
% F1out: F1 gain matrix for 2nd-order plant
% F2out: F2 gain matrix for 2nd-order plant
%
% Written by K.F. Alvin
%
format short e
nmodmax=length(lam);
[ndof,nact]=size(qb);
%
% Variables:
% mass: mass matrix
```



```

% A: state transition matrix
% B: actuator influence matrix
% G: control gain matrix
%
massd(m(:,3))=m(:,4);
mass=diag(massd);
A=[zeros(nmode),eye(nmode);-1*diag(lam(1:nmode)),zeros(nmode)];
Amax=[zeros(nmodmax),eye(nmodmax);-1*diag(lam),zeros(nmodmax)];
B=[zeros(nmode,nact);t(:,1:nmode)*qb];
Bmax=[zeros(nmodmax,nact);t'*qb];
disp('Number of structural modes and actuators used:')
disp([nmode,nact])
disp('Rank of the controllability matrix:')
disp(constrank(A,B))
disp('Determining controller gains for given system...')
G=lqr(A,B,Q,R);
wmax=1.1*max(sqrt(lam));
axis([-wmax,wmax,-wmax,wmax]);
plot(eig(A-B*G),'*')
grid
title('Roots of controlled system')
hold
pause
%
% partition gain matrix
%
G1=G(:,1:nmode);
G2=G(:,nmode+1:nmode+nmode);
%
% Transform resultant gain matrices for use in
% partitioned csi algorithm using second-order
% structure(plant) equations.
%
disp('Mapping gains back to physical domain...')
f1=G1*t(:,1:nmode)*mass;
f2=G2*t(:,1:nmode)*mass;
%
% find modal damping ratios of controller for calculated gains:
%
Gmax=[f1*t,f2*t];
lambda=eig(Amax-Bmax*Gmax);
plot(lambda,'*')
pause
nfreq=sqrt(imag(lambda).^2 + real(lambda).^2);
mdamp=-real(lambda)./nfreq;
disp('Resultant modal damping ratios for controller:')
disp([' Damping ', ' Damped Freq (rad/s) '])
disp([mdamp,nfreq.*sqrt(1-mdamp.^2),lambda])
bdamp=max(-2*mdamp./nfreq);
disp('Estimated minimum stiffness damping coefficient necessary')
disp(' to stabilize residual modes due to gain roundoff accumulation:')
disp(bdamp)
disp('Writing gains in node correspondence output form...')

```

```

Flout=zeros(nact*ndof,4);
F2out=zeros(nact*ndof,4);
for i=1:nact;
    kmin=(i-1)*ndof+1;
    kmax=i*ndof;
    Flout(kmin:kmax,:)= [m(:,1:2),i*ones(ndof,1),f1(i,m(:,3))'];
    F2out(kmin:kmax,:)= [m(:,1:2),i*ones(ndof,1),f2(i,m(:,3))'];
end;
disp('Finished mlqr')

```

File: mkf.m

```

function [L1out,L2out]=mkf(lam,t,m,hd,hv,Q,R,nmode)
%
% Kalman filter design for second-order
% structural system via given eigenmodes
% and transformed to independent
% displacement/gen. momentum variable
% casting for partitioned csi transient
% analysis. See Belvin/Park paper for
% filter variable definitions.
%
% Arguments:
%
% lam: vector of eigenvalues (nmode x 1)
% t: matrix of eigenvectors (ndof x nmode)
% m: mass diagonal and dof mapping info (ndof x 4)
% hd: sensor position influence matrix (nsen x ndof)
% hv: velocity position influence matrix (nsen x ndof)
% Q: optimal design state weighting matrix (2*nmode x 2*nmode)
% R: optimal design feedbk weight. matrix (nsen x nsen)
%
% L1out: L1 gain matrix for 2nd-order filter
% L2out: L2 gain matrix for 2nd-order filter
%
% Written by K.F. Alvin
%
format short e
nmodmax=length(lam);
[nsen,ndof]=size(hd);
%
% Variables:
% mass: mass matrix
% A: state transition matrix
% G: noise influence matrix
% C: output influence matrix
% K: filter gain matrix
%
massd(m(:,3))=m(:,4);
mass=diag(massd);
A=[zeros(nmode),eye(nmode);-1*diag(lam(1:nmode)),zeros(nmode)];
Amax=[zeros(nmodmax),eye(nmodmax);-1*diag(lam),zeros(nmodmax)];

```

```

G=eye(nmode+nmode);
C=[hd*t(:,1:nmode),hv*t(:,1:nmode)];
Cmax=[hd*t,hv*t];
disp('Number of structural modes and sensors used:')
disp([nmode,nsen])
disp('Rank of the observability matrix:')
disp(contrank(A',C'))
disp('Determining filter gains for given system...')
K=lqe(A,G,C,Q,R);
%
% partition gain matrix
%
K1=K(1:nmode,:);
K2=K(nmode+1:nmode+nmode,:);
Kmax=[K1;zeros(nmodmax-nmode,nsen);K2;zeros(nmodmax-nmode,nsen)];
%
% find modal damping ratios of filter for calculated gains:
%
lambda=eig(Amax-Kmax*Cmax);
plot(lambda,'+')
hold
pause
a1=-real(lambda);
b1=imag(lambda);
disp('Resultant modal damping ratios for filter:')
disp([' Damping ', ' Freq (rad/s) '])
disp([a1./sqrt(a1.^2+b1.^2),b1])
%
% Transform resultant gain matrices for use in
% partitioned csi algorithm using second-order
% Kalman filter approach.
%
disp('Mapping gains back to physical domain...')
l1=t(:,1:nmode)*K1;
l2=mass*t(:,1:nmode)*K2;
disp('Writing gains in node correspondence output form...')
L1out=zeros(nsen*ndof,4);
L2out=zeros(nsen*ndof,4);
for i=1:nsen;
    kmin=(i-1)*ndof+1;
    kmax=i*ndof;
    L1out(kmin:kmax,:)= [m(:,1:2),i*ones(ndof,1),l1(m(:,3),i)];
    L2out(kmin:kmax,:)= [m(:,1:2),i*ones(ndof,1),l2(m(:,3),i)];
end;
disp('Finished mkf')

```

File: contrank.m

```

function maxrank=contrank(a,b)
maxrank=0;
[nstate,nact]=size(b);
i=0;

```

```

mat=b;
newrank=rank(mat);
while newrank > maxrank
    maxrank=newrank;
    i=i+1;
    mat=[mat,(a^i)*b];
    newrank=rank(mat);
    if newrank==nstate
        maxrank=newrank;
    end
end

```

Unfortunately, the external files created from Matlab with the gain results are written completely in terms of real numbers, whereas the first three columns are actually to be read by ACSIS as integers (they are used as indices). A separate utility was written to convert the format of these files; the source code is listed below. On Unix systems, the user simply assigns the standard input to be the current data file created by Matlab, and gives another file name for the standard output. The code is basically just a filter to change the three columns of indices to integers. The output can then be pasted directly into the control definition file used by ACSIS.

File: convcont.f

```

program convcont

parameter (NMAX=100000)
real*8 f(NMAX),v(4)
integer node(NMAX),dof(NMAX),act(NMAX)

n=0

100  read (*,*,err=200,end=200) (v(i),i=1,4)
     n=n+1
     node(n)=int(v(1))
     dof(n) =int(v(2))
     act(n) =int(v(3))
     f(n)   =v(4)
     goto 100

200  do 300 i=1,n
     print *, node(i),dof(i),act(i),f(i)
300  continue

end

```

APPENDIX C

Stability Analysis of a CSI Partitioned Simulation Algorithm with State Estimator

The equation of the open-loop plant without passive damping in modal second-order form is

$$\ddot{q} + \omega^2 q = u \quad (1)$$

The controller uses a second-order observer to estimate the plant state, along with a full-state feedback control gain design.

$$\begin{aligned} u &= -(\eta\omega^2 p + \zeta\omega \dot{p}) \\ \ddot{p} + \omega^2 p &= u + \xi\gamma \\ \gamma &= z - \dot{p} \\ z &= \dot{q} \end{aligned} \quad (2)$$

where q and p are the plant and estimator states, respectively, u is the control force, γ is the state estimation error, z is the sensor output, and η, ζ, ξ are gain coefficients for position and velocity feedback, and the estimator filter.

The partitioned analysis procedure uses a stabilized form of the control law and estimation error determination to reduce inaccuracies associated with the extrapolation of variables in the controller force prediction. A first-order filtering is achieved by taking the time derivative of (2a,c),

$$\begin{aligned} \dot{u} &= -\eta\omega^2 \dot{p} - \zeta\omega \ddot{p} \\ \dot{\gamma} &= \dot{z} - \ddot{p} \end{aligned} \quad (3)$$

and then embedding the equations of motion through substitution for $\ddot{p}$. This leads to the following two coupled, first-order differential equations for the prediction of the control force u and state error γ .

$$\begin{Bmatrix} \dot{u} \\ \dot{\gamma} \end{Bmatrix} + \begin{bmatrix} \zeta\omega & \zeta\xi\omega \\ 1 & \xi \end{bmatrix} \begin{Bmatrix} u \\ \gamma \end{Bmatrix} = \begin{Bmatrix} 0 \\ \dot{z} \end{Bmatrix} + \begin{Bmatrix} \zeta\omega^3 \\ \omega^2 \end{Bmatrix} p - \begin{Bmatrix} \eta\omega^2 \\ 0 \end{Bmatrix} \dot{p} \quad (4)$$

Time discretization of (4) using an implicit midpoint rule leads to the following coupled difference equation:

$$\begin{bmatrix} 1 + \delta\zeta\omega & \delta\zeta\xi\omega \\ \delta & 1 + \delta\xi \end{bmatrix} \begin{Bmatrix} u^{n+\frac{1}{2}} \\ \gamma^{n+\frac{1}{2}} \end{Bmatrix} = \begin{Bmatrix} 0 \\ z_p^{n+\frac{1}{2}} \end{Bmatrix} + \begin{Bmatrix} \delta\zeta\omega^3 - \eta\omega^2 \\ \delta\omega^2 \end{Bmatrix} p_p^{n+\frac{1}{2}} - \begin{Bmatrix} \zeta\omega \\ 1 \end{Bmatrix} \dot{p}^n \quad (5)$$

where $\delta \equiv \text{half-step size} = \frac{h}{2}$. Solving this equation requires knowledge of the plant (to obtain sensor output) and observer states. These values are extrapolated as:

$$\begin{aligned} p_p^{n+\frac{1}{2}} &= p^n + \delta \dot{p}^n \\ z_p^{n+\frac{1}{2}} &= \dot{q}_p^{n+\frac{1}{2}} = \dot{q}^n \end{aligned} \quad (6)$$

Using these equations to obtain $u^{n+\frac{1}{2}}$ and $\gamma^{n+\frac{1}{2}}$ allows the plant and observer equations to be solved independently. Midpoint time integration of (1) and (2b) leads to the following equations:

$$\begin{aligned}
 (1 + \delta^2 \omega^2) q^{n+\frac{1}{2}} &= \delta^2 u^{n+\frac{1}{2}} + q^n + \delta \dot{q}^n \\
 (1 + \delta^2 \omega^2) p^{n+\frac{1}{2}} &= \delta^2 u^{n+\frac{1}{2}} + p^n + \delta \dot{p}^n + \delta^2 \xi \gamma^{n+\frac{1}{2}} \\
 \dot{q}^{n+\frac{1}{2}} &= \frac{1}{\delta} (q^{n+\frac{1}{2}} - q^n) \\
 \dot{p}^{n+\frac{1}{2}} &= \frac{1}{\delta} (p^{n+\frac{1}{2}} - p^n) \\
 q^{n+1} &= 2q^{n+\frac{1}{2}} - q^n \\
 p^{n+1} &= 2p^{n+\frac{1}{2}} - p^n \\
 \dot{q}^{n+1} &= 2\dot{q}^{n+\frac{1}{2}} - \dot{q}^n \\
 \dot{p}^{n+1} &= 2\dot{p}^{n+\frac{1}{2}} - \dot{p}^n
 \end{aligned} \tag{7}$$

Computational stability of the modal form of the CSI partitioned equations of motion using the aforementioned time discretization can be assessed by seeking a nontrivial solution of

$$\begin{Bmatrix} q^{n+1} \\ p^{n+1} \\ \dot{q}^{n+1} \\ \dot{p}^{n+1} \end{Bmatrix} = \lambda \begin{Bmatrix} q^n \\ p^n \\ \dot{q}^n \\ \dot{p}^n \end{Bmatrix} \tag{8}$$

such that

$$|\lambda| \leq 1 \tag{9}$$

for stability. Substituting (8) into (5-7), we obtain

$$\mathbf{J}\mathbf{x} = 0 \tag{10}$$

where

$$\mathbf{x}_1 = [p_p^{n+\frac{1}{2}} \quad u^{n+\frac{1}{2}} \quad p^{n+\frac{1}{2}} \quad \dot{p}^{n+\frac{1}{2}} \quad p^n \quad \dot{p}^n]^T$$

$$\mathbf{x}_2 = [z_p^{n+\frac{1}{2}} \quad \gamma^{n+\frac{1}{2}} \quad q^{n+\frac{1}{2}} \quad \dot{q}^{n+\frac{1}{2}} \quad q^n \quad \dot{q}^n]^T$$

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{J}_{12} \\ \mathbf{J}_{21} & \mathbf{J}_{22} \end{bmatrix} \tag{11}$$

$$J_{11} = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & -\delta \\ (\eta\omega^2 - \delta\zeta\omega^3) & (1 + \delta\zeta\omega) & 0 & 0 & 0 & \zeta\omega \\ 0 & -\delta^2 & (1 + \delta^2\omega^2) & 0 & -1 & -\delta \\ 0 & 0 & -\frac{1}{\delta} & 1 & \frac{1}{\delta} & 0 \\ 0 & 0 & -2 & \lambda + 1 & 0 & 0 \\ 0 & 0 & 0 & -2 & 0 & \lambda + 1 \end{bmatrix} \quad (12)$$

$$J_{12} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \delta\zeta\xi\omega & 0 & 0 & 0 & 0 \\ 0 & -\delta^2\xi & 0 & 0 & 0 & 0 \\ 0 & -\delta\xi & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (13)$$

$$J_{21} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -\delta\omega^2 & \delta & 0 & 0 & 0 & 1 \\ 0 & -\delta^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (14)$$

$$J_{22} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -1 \\ -1 & (1 + \delta\xi) & 0 & 0 & 0 & 0 \\ 0 & 0 & (1 + \delta^2\omega^2) & 0 & -1 & -\delta \\ 0 & 0 & -\frac{1}{\delta} & 1 & \frac{1}{\delta} & 0 \\ 0 & 0 & -2 & 0 & \lambda + 1 & 0 \\ 0 & 0 & 0 & -2 & 0 & \lambda + 1 \end{bmatrix} \quad (15)$$

A nontrivial solution to (10) is found from

$$\det J = 0 \quad (16)$$

which leads to the characteristic equation

$$\begin{aligned} & ((1 - \delta\xi)(1 + \delta^3\zeta\omega^3 - \delta^2\eta\omega^2) + \delta\xi(1 + \delta^2\omega^2))z^4 \\ & + (\delta\zeta\omega(1 - \delta\xi) + \delta\xi(1 + \delta^3\zeta\omega^3 - \delta^2\eta\omega^2))z^3 \\ & ((1 - \delta\xi)(\delta^2\omega^2 + \delta^2\eta\omega^2) + \delta^2\omega^2(1 + \delta^3\zeta\omega^3 - \delta^2\eta\omega^2) \\ & + \delta\xi(\delta\zeta\omega + \delta^2\omega^2 + \delta^4\omega^4))z^2 \\ & (\delta^2\omega^2(\delta\zeta\omega) + \delta\xi(\delta^2\omega^2 + \delta^2\eta\omega^2))z \\ & + \delta^2\omega^2(\delta^2\omega^2 + \delta^2\eta\omega^2) = 0 \end{aligned} \quad (17)$$

where

$$\lambda = \frac{1+z}{1-z}, \quad |\lambda| \leq 1 \iff \operatorname{Re}(z) \leq 0 \quad (18)$$

Thus, a test of the polynomial equation for possible positive real roots by the Routh-Hurwitz criterion indicates that the partitioned approach as applied to the modal equations give a computationally stable solution for no velocity feedback $\zeta = 0$ provided

$$h \leq \frac{2}{\sqrt{\eta\omega}} \quad (19)$$

$$h \leq \frac{2}{\xi} \quad (20)$$

APPENDIX D

ACSIS Source Code

File: Makefile

.SUFFIXES: .f .o

FFLAGS =

.f.o:

fortran -c \$(FFLAGS) *.f

```
OBJS =      acsis.o      acsisout.o      addstf.o      \  
            beam3d.o     forces.o      input.o      \  
            pmvmul.o     prepfem.o     profile.o     \  
            read.o       solver.o      nophlag.o     \  
            zerovect.o   lu.o          prepcon.o     \  
            control.o    secorder.o    measure.o     \  
            eigens.o     singeig.o    animout.o     \  
            stiffr.o     estifvm.o    renum.o      \  
            kfilter.o
```

acsis.exe: \$(OBJS)
fortran -o acsis.exe \$(FFLAGS) \$(OBJS)

File: shared.inc

```
C  
C -----  
C  
C shared.inc (ACSIS database)  
C  
C -----  
C  
C Argument definitions:  
C adamp: Rayleigh damping coefficient alpha  
C b: actuator location matrix (packed storage)  
C brow: row number of corresponding real value in b  
C bcol: column number of corresponding real value in b  
C bval: number of nonzero values in b  
C bdamp: Rayleigh damping coefficient beta  
C confile: controller input file name  
C contype: id for type of control  
C coxyz: array of the x,y, and z components of each node  
C delta: one-half time step  
C delsq: one-half time step squared  
C ec: control prediction integration coefficient matrix  
C emat: array of element material types  
C eo: observer construct matrix S in vector form  
C eprop: array of element property types  
C es: structure construct matrix S (M+delta*D+delsq*K)  
C etype: array of element types  
C elnum: array of element numbers for domain decomposition  
C f: vector of applied forces  
C f1: control gain matrix
```

C f2: control gain matrix
C femfile: finite element input file name
C forceid: identification number of forcing function
C gamma: state correction force
C gc: RHS vector for control prediction module
C gk: RHS vector for Kalman filter momentum eqn
C go: RHS vector for observer module
C gs: RHS vector for structure module
C h: time step size
C hd: displacement sensor location matrix (packed storage)
C hdrow: row number of corresponding real value in hd
C hdccl: column number of corresponding real value in hd
C hdval: number of nonzero values in hd
C hv: velocity sensor location matrix (packed storage)
C hvrow: row number of corresponding real value in hv
C hvcol: column number of corresponding real value in hv
C hvval: number of nonzero values in hv
C id: DOF mapping array: id(comp,node #)=Global DOF #
C ix: array of element connectivity and orientation
C inertia: array of concentrated inertias or lumped masses
C jdiag: array of diagonal element addresses
C l1: State estimator filter gain matrix
C l2: State estimator filter gain matrix
C mask:
C mass: mass matrix M in reduced vector form
C mat: array of different materials
C mlen: length of global matrices in profile vector storage
C nact: actual number of actuators
C ncsi: actual number of actuators and sensors
C ndisout: number of displacement results to output
C ndof: actual number of degrees of freedom
C ndomain: actual number of element domains (for dom. decomp.)
C nel: actual number of elements
C neld: array of number of elements in each domain
C nnp: actual number of nodes
C nsen: actual number of sensors
C nvelout: number of velocity results to output
C nolag: logical flag to signal corrector loop in measurement
C outfile: output file name
C outlabel: array of output data requested
C pin: array of element pin release codes
C pivot: Column pivoting info from FACTA
C prop: array of different properties
C q: generalized displacement vector
C q0: initial displacement condition
C qalpha: gain scale factor for f1
C qalpha0: gain scale factor for l1
C qbeta: gain scale factor for f2
C qbeta0: gain scale factor for l2
C qdot: velocity vector
C qdot0: initial velocity condition
C qe: state estimator displacement vector
C qedot: state estimator velocity vector
C r: Solution vector of control module {u,gamma}
C scalef: Scaling factor for forcing function
C stiff: stiffness matrix K in reduced vector form
C t0: initial time
C tc: control-on time
C tf: final time

```

C      u:      vector of control forces
C
C      Parameter definitions
C      MAXACT:   max. # of actuators
C      MAXCSI:   max. combined # of actuators and sensors
C      MAXDAT:   max. # of materials and properties
C      MAXDOF:   max. # of degrees of freedom
C      MAXDOM:   max. # of decomposition domains
C      MAXELE:   max. # of elements
C      MAXLEN:   max. length of global vectors in reduced form
C      MAXNODE:  max. # of nodes
C
parameter(MAXDOF=3000, MAXACT=50, MAXCSI=100)
parameter(MAXNODE=1000, MAXELE=3000, MAXDAT=100)
parameter(MAXLEN=200000, MAXDOM=50, MAXNZV=200)

real*8 t0,tf,tc,h,delta,delsq,qalpha,qbeta,qalphao,qbetao
real*8 q(MAXDOF),qdot(MAXDOF),qe(MAXDOF),qedot(MAXDOF)
real*8 u(MAXACT),gamma(MAXACT),f(MAXDOF),r(MAXCSI)
real*8 es(MAXLEN),eo(MAXLEN),ec(MAXCSI,MAXCSI)
real*8 gs(MAXDOF),go(MAXDOF),gc(MAXCSI),scalef,pe(MAXDOF)
real*8 mass(MAXLEN),stif(MAXLEN),adamp,bdamp,gk(MAXDOF)
real*8 coxyz(3,MAXNODE),mat(6,MAXDAT),prop(10,MAXDAT)
real*8 q0(6,MAXNODE),qdot0(6,MAXNODE),inertia(6,MAXNODE)
real*8 b(MAXNZV),hd(MAXNZV),hv(MAXNZV),estif(78,500)
real*8 f1(MAXACT,MAXDOF),f2(MAXACT,MAXDOF)
real*8 l1(MAXDOF,MAXACT),l2(MAXDOF,MAXACT)
integer etype(MAXELE),ix(4,MAXELE),emat(MAXELE),forceid
integer eprop(MAXELE),pin(6,MAXELE),id(6,MAXNODE)
integer mask(MAXNODE),contype,brow(MAXNZV),bcol(MAXNZV)
integer hdrow(MAXNZV),hdcol(MAXNZV),hvrow(MAXNZV)
integer hvcol(MAXNZV),bval,hdval,hvval
integer ndof,nact,nsen,ncsi,mlen,jdiag(MAXDOF),nnp,nel,neig
integer neld(MAXDOM),elnum(MAXELE,MAXDOM),eldom(MAXELE),ndomain
integer outlabel(40),ndisout,nvelout,pivot(MAXCSI)
integer iadjcy(MAXLEN),icount(MAXNODE+1),perm(MAXNODE)
integer xls(MAXNODE)
logical animate,nolag
character*32 femfile,confile,outfile,animfile

COMMON /FILES/ femfile,confile,outfile,outlabel,ndisout,
               nvelout,animfile,animate,nolag
COMMON /TIMERS/ t0,tf,tc,h,delta,delsq
COMMON /STATES/ q,qdot,qe,qedot,u,gamma,f,r,pe
COMMON /FENDAT/ mass,stif,adamp,bdamp,coxyz,mat,prop,q0,qdot0,
               inertia,scalef
COMMON /INTEGR/ es,eo,ec,gs,go,gc,gk,pivot
COMMON /DIMENS/ ndof,nact,nsen,ncsi,mlen,jdiag,nnp,nel,neig
COMMON /CONDAT/ b,hd,hv,f1,f2,l1,l2,
               qalpha,qbeta,qalphao,qbetao
COMMON /ELEDOM/ estif,ndomain,neld,elnum,eldom
COMMON /INTGER/ forceid,etype,ix,emat,eprop,pin,id,mask,
               contype,brow,bcol,hdrow,hdcol,hvrow,hvcol,
               bval,hdval,hvval
COMMON /RESEQN/ iadjcy,icount,perm,xls

```

File: acsis.f

C=Program ACSIS
 C=Purpose Accelerated CSI Simulation
 C=Author K. Alvin
 C=Date May 1990
 C=Block Fortran

```

C-----C
C                                     C
C   program ACSIS                     C
C                                     C
C   Purpose:  2nd Order Accelerated CSI Simulation  C
C                                     C
C                                     C
C-----C

```

```

      program ACSIS

C   GET SHARED DATA FILE

      include 'shared.inc'

C   LOCAL VARIABLES

      real*8 t,z(MAXACT)

      integer n,m,runtime,outskip

C   LOGIC

      call INPUT(runtime,outskip)

      if (runtime) 100,200,300

C   EIGENMODE ANALYSIS

100  continue
      call PREPFEM
      call EIGENS
      goto 999

C   CSI SIMULATION

200  call PREPFEM
      call PREPCOM

      n = 0
      m = 0
      print *, 'Finished Preprocessing . . . starting simulation'
      print *, 'Time =', t0
      CALL ACSISOUT(t0)

      do 250 t=t0,t1,h

          call FORCES(t+h/2)

C   Predict CSI coupling variables u and gamma

```

```

        if (t .ge. tc) then
            call MEASURE(z)
            call CONTROL(z)
            if (nolag) then
                call NOPHLAG(z)
                call CONTROL(z)
            endif
        endif

C      Structure and observer set up for parallel execution

CVD$  CNCALL
      do 275 i=1,2

C      Integrate Observer Equations

        if ((i .eq. 1).and.(t .ge. tc)) then
            if (contype .eq. 0) then
                call SECORDER(mass,stif,adamp,bdamp,f,go,co,qe,qedot,
                             delta,delsq,jdiag,ndof,MAXDOF)
            elseif (contype .eq. 1) then
                call KFILTER
            endif

C      Integrate Structure Equations

            elseif (i .eq. 2) then
                call SECORDER(mass,stif,adamp,bdamp,f,gs,es,q,qdot,
                             delta,delsq,jdiag,ndof,MAXDOF)
            endif

275      continue

C      PRINT TIME EACH 100 iterations

        n = n + 1
        m = m + 1
        if (n .ge. 100) then
            print *, 'Time = ',t+h
            n = 0
        endif
        if (m .ge. outskip) then
            call ACSISOUT(t+h)
            write(24,'(40f12.8)') t,(z(i),i=1,nlen)
            m = 0
        endif

250      continue
      goto 999

C      TRANSIENT RESPONSE

300      call PREPFEM

        n = 0
        m = 0
        print *, 'Finished Preprocessing . . . starting simulation'
        print *, 'Time = ',t0

```

```

    call ACSISOUT(t0)

    do 350 t=t0,tf,h

        call FORCES(t+h/2)

        call ZEROVECT(gs,ndof)

        call SEORDER(mass,stif,adamp,bdamp,f,gs,es,q,qdot,
                     delta,delsq,jdiag,ndof,MAXDOF)

C    PRINT TIME EACH 100 iterations

        n = n + 1
        m = m + 1
        if (n .ge. 100) then
            print *, 'Time = ', t+h
            n = 0
        endif
        if (m .ge. outskip) then
            call ACSISOUT(t+h)
            m = 0
        endif

350    continue

999    stop
    end

```

File: acsisout.f

```

C=Module ACSISOUT
C=Purpose Write desired output from ACSIS for plotting, etc.
C=Author K. Alvin
C=Date May 1990
C=Block Fortran
C -----
C
C    Subroutine ACSISOUT
C
C    Purpose:
C        This subroutine outputs formatted displacement and velocity
C        results at a given time for plotting time histories. The
C        desired output variables are defined in outlabel().
C -----
C
C    Arguments
C        t - time
C
C
C    subroutine ACSISOUT(t)
C
C        include 'shared.inc'
C        real*8 t

```


C LOCAL VARIABLES

integer i

C LOGIC

```
write(13,'(40f12.8)') t,(q(id(outlabel(i+10),outlabel(i))),
. i=1,ndisout),(qdot(id(outlabel(i+30),outlabel(i+20))),
. i=1,nvelout)
```

```
write(23,'(40f12.8)') t,(qe(id(outlabel(i+10),outlabel(i))),
. i=1,ndisout),(qedot(id(outlabel(i+30),outlabel(i+20))),
. i=1,nvelout)
```

```
write(25,'(40f12.8)') t,(u(i),i=1,nact),(gamma(i),i=1,nseu)
```

```
if (animate) call ANINOUT(q,id,nnp,t,15)
```

```
return
end
```

File: addstf.f

C=Module ADDSTF

C=Purpose Assemble Global stiffness matrix

C=Author who knows

C=Update January 1989, by E. Pramono

C=Block Fortran

```
subroutine ADDSTF(sk,lm,bk,jdiag,nseq)
```

```
C+-----+C
C PURPOSE: C
C THIS SUBROUTINE ASSEMBLES THE ELEMENT STIFFNESS MATRICES C
C INTO THE COMPACTED GLOBAL STIFFNESS VECTOR. C
C C
C ARGUMENTS: C
C sk - ELEMENT STIFFNESS MATRIX C
C lm - LOCATION VECTOR FOR ELEMENT STIFFNESS MATRIX C
C bk - COMPACTED GLOBAL STIFFNESS VECTOR C
C jdiag - VECTOR OF DIAGONAL ELEMENT ADDRESSES C
C nseq - NUMBER OF DEGREES OF FREEDOM PER ELEMENT C
C-----C
```

C ARGUMENTS

```
real*8 sk(nseq,nseq), bk(1)
```

```
integer lm(18), jdiag(1)
```

```
C integer lm(18), jdiag(1), nseq
```

C LOCAL ARGUMENTS

```
integer i, j, k, l, m
```

C ASSEMBLE GLOBAL STIFFNESS AND LOAD ARRAYS

```
do 20 j = 1, nseq
```

```
  k = lm(j)
```

```
  if (k .eq. 0) goto 20
```

```

      l = jdiag(k) - k
C      l = jdiag(k+1) - k
      do 10 i = 1, nseq
        m = lm(i)
        if(m .gt. k .OR. m .eq. 0) goto 10
        m = l + m
        bk(m) = bk(m) + sk(i,j)
10      continue
20      continue
C
      return
      end
C=End Fortran

```

File: beam3d.f

```

C=Module BEAM3D
C=Purpose Construct 3-d Timoshenko beam element stiffness and lumped mass
C=Author K. Alvin
C=Date May 1990
C=Block Fortran

```

```

      subroutine BEAM3D(n,ni,nj,nk,xyz,emod,gmod,rho,area,ssf2,ssf3,
      &                  jtor,i2,i3,ipin,sk,sm)

```

C ARGUMENTS:

```

C
C      n      Element ID Number
C      ni     Node ID Number at End i
C      nj     Node ID Number at End j
C      xyz    Node Location Array
C      emod   Material Elastic Modulus (Young's Modulus)
C      gmod   Material Modulus of Rigidity (Shear Modulus)
C      rho    Material Mass Density
C      area   Element Cross-sectional area
C      ssf2   Shear shape factor in element x2 direction
C      ssf3   Shear shape factor in element x3 direction
C      jtor   Torsional constant J
C      i2     Area moment of inertia about element x2 axis
C      i3     Area moment of inertia about element x3 axis
C      ipin   Pin release codes: 0=Fixed, 1=Fixed
C            (1) Axial
C            (2) Torsional
C            (3) End A rotation about x2 axis
C            (4) End A rotation about x3 axis
C            (5) End B rotation about x2 axis
C            (6) End B rotation about x3 axis
C      sk     Element Stiffness Matrix
C      sm     Element Mass Matrix

```

```

      integer n,ni,nj,nk,ipin(1)
      real*8 xyz(3,1),emod,gmod,rho,area,ssf2,ssf3,jtor,i2,i3
      real*8 sk(12,1),sm(12,1)

```

C LOCAL VARIABLES:

```

      integer i,j
      real*8 dc(3,3),length,xlength,kc(10),mc(3)

```

C LOGIC

C Find Element Length

```
length = 0.0d0
do 10 i = 1,3
  dc(1,i) = xyz(i,nj) - xyz(i,ni)
  length = length + dc(1,i)**2
10 continue
length = sqrt(length)
if (length .eq. 0.0d0) then
  print *, 'BAR2D: Zero element length; n= ',n
  return
endif
```

C Find direction cosines for x1,x2,x3 element axes

```
do 15 i=1,3
  dc(1,i) = dc(1,i)/length
  if (nk .eq. 0) then
    dc(2,i) = 0.0
  else
    dc(2,i) = xyz(i,nk) - xyz(i,ni)
  endif
15 continue
if (nk .eq. 0) dc(2,3) = 1.0
16 dc(3,1) = dc(1,2)*dc(2,3) - dc(2,2)*dc(1,3)
  dc(3,2) = dc(2,1)*dc(1,3) - dc(1,1)*dc(2,3)
  dc(3,3) = dc(1,1)*dc(2,2) - dc(2,1)*dc(1,2)
  rlength = sqrt(dc(3,1)**2 + dc(3,2)**2 + dc(3,3)**2)
  if (rlength .ne. 0.) goto 17
  dc(2,2) = 1.0
  dc(2,3) = 0.0
  goto 16
17 do 18 i=1,3
  dc(3,i) = dc(3,i)/rlength
18 continue
dc(2,1) = dc(3,2)*dc(1,3) - dc(1,2)*dc(3,3)
dc(2,2) = dc(1,1)*dc(3,3) - dc(3,1)*dc(1,3)
dc(2,3) = dc(3,1)*dc(1,2) - dc(1,1)*dc(3,2)
```

C Compute various stiffness constants, accounting for pin codes

```
if (ipin(1) .eq. 0) then
  kc(1) = area*emod/length
else
  kc(1) = 0.0d0
endif

if (ipin(4) .eq. 0) then
  if (ipin(6) .eq. 0) then
    kc(2) = area*gmod*ssf2/length
    kc(6) = i3*emod/length
    kc(7) = kc(2)*length/2.0d0
    kc(9) = kc(7)*length/2.0d0
  else
    print *, 'BEAM3D: Pin code error, x3 direction, el #',n
```

```

endif
else
  if (ipin(6) .eq. 0) then
    print *, 'BEAM3D: Pin code error, x3 direction, el #', n
  else
    kc(2) = 0.0d0
    kc(6) = 0.0d0
    kc(7) = 0.0d0
    kc(9) = 0.0d0
  endif
endif

if (ipin(3) .eq. 0) then
  if (ipin(5) .eq. 0) then
    kc(3) = area*gmod*ssf3/length
    kc(5) = i2*emod/length
    kc(8) = kc(3)*length/2.0d0
    kc(10) = kc(8)*length/2.0d0
  else
    print *, 'BEAM3D: Pin code error, x2 direction, el #', n
  endif
else
  if (ipin(5) .eq. 0) then
    print *, 'BEAM3D: Pin code error, x2 direction, el #', n
  else
    kc(3) = 0.0d0
    kc(5) = 0.0d0
    kc(8) = 0.0d0
    kc(10) = 0.0d0
  endif
endif

if (ipin(2) .eq. 0) then
  kc(4) = jtor*gmod/length
else
  kc(4) = 0.0d0
endif

mc(1) = area*rho*length/2.0d0
mc(2) = i2*rho*length/2.0d0
mc(3) = i3*rho*length/2.0d0

sk(1,1) = kc(1)*dc(1,1)*dc(1,1) + kc(2)*dc(2,1)*dc(2,1) +
          kc(3)*dc(3,1)*dc(3,1)
sk(1,2) = kc(1)*dc(1,1)*dc(1,2) + kc(2)*dc(2,1)*dc(2,2) +
          kc(3)*dc(3,1)*dc(3,2)
sk(1,3) = kc(1)*dc(1,1)*dc(1,3) + kc(2)*dc(2,1)*dc(2,3) +
          kc(3)*dc(3,1)*dc(3,3)
sk(1,4) = kc(7)*dc(2,1)*dc(3,1) - kc(8)*dc(3,1)*dc(2,1)
sk(1,5) = kc(7)*dc(2,1)*dc(3,2) - kc(8)*dc(3,1)*dc(2,2)
sk(1,6) = kc(7)*dc(2,1)*dc(3,3) - kc(8)*dc(3,1)*dc(2,3)
sk(1,7) = -sk(1,1)
sk(1,8) = -sk(1,2)
sk(1,9) = -sk(1,3)
sk(1,10) = sk(1,4)
sk(1,11) = sk(1,5)
sk(1,12) = sk(1,6)
sk(2,2) = kc(1)*dc(1,2)*dc(1,2) + kc(2)*dc(2,2)*dc(2,2) +
          kc(3)*dc(3,2)*dc(3,2)

```

```

sk(2,3) = kc(1)*dc(1,2)*dc(1,3) + kc(2)*dc(2,2)*dc(2,3) +
          kc(3)*dc(3,2)*dc(3,3)
sk(2,4) = kc(7)*dc(2,2)*dc(3,1) - kc(8)*dc(3,2)*dc(2,1)
sk(2,5) = kc(7)*dc(2,2)*dc(3,2) - kc(8)*dc(3,2)*dc(2,2)
sk(2,6) = kc(7)*dc(2,2)*dc(3,3) - kc(8)*dc(3,2)*dc(2,3)
sk(2,7) = -sk(1,2)
sk(2,8) = -sk(2,2)
sk(2,9) = -sk(2,3)
sk(2,10) = sk(2,4)
sk(2,11) = sk(2,5)
sk(2,12) = sk(2,6)
sk(3,3) = kc(1)*dc(1,3)*dc(1,3) + kc(2)*dc(2,3)*dc(2,3) +
          kc(3)*dc(3,3)*dc(3,3)
sk(3,4) = kc(7)*dc(2,3)*dc(3,1) - kc(8)*dc(3,3)*dc(2,1)
sk(3,5) = kc(7)*dc(2,3)*dc(3,2) - kc(8)*dc(3,3)*dc(2,2)
sk(3,6) = kc(7)*dc(2,3)*dc(3,3) - kc(8)*dc(3,3)*dc(2,3)
sk(3,7) = -sk(1,3)
sk(3,8) = -sk(2,3)
sk(3,9) = -sk(3,3)
sk(3,10) = sk(3,4)
sk(3,11) = sk(3,5)
sk(3,12) = sk(3,6)
sk(4,4) = kc(4)*dc(1,1)*dc(1,1)+(kc(10)+kc(5))*dc(2,1)*dc(2,1)
          + (kc(9)+kc(6))*dc(3,1)*dc(3,1)
sk(4,5) = kc(4)*dc(1,1)*dc(1,2)+(kc(10)+kc(5))*dc(2,1)*dc(2,2)
          + (kc(9)+kc(6))*dc(3,1)*dc(3,2)
sk(4,6) = kc(4)*dc(1,1)*dc(1,3)+(kc(10)+kc(5))*dc(2,1)*dc(2,3)
          + (kc(9)+kc(6))*dc(3,1)*dc(3,3)
sk(4,7) = -sk(1,4)
sk(4,8) = -sk(2,4)
sk(4,9) = -sk(3,4)
sk(4,10) = -kc(4)*dc(1,1)*dc(1,1)+(kc(10)-kc(5))*dc(2,1)*dc(2,1)
           + (kc(9)-kc(6))*dc(3,1)*dc(3,1)
sk(4,11) = -kc(4)*dc(1,1)*dc(1,2)+(kc(10)-kc(5))*dc(2,1)*dc(2,2)
           + (kc(9)-kc(6))*dc(3,1)*dc(3,2)
sk(4,12) = -kc(4)*dc(1,1)*dc(1,3)+(kc(10)-kc(5))*dc(2,1)*dc(2,3)
           + (kc(9)-kc(6))*dc(3,1)*dc(3,3)
sk(5,5) = kc(4)*dc(1,2)*dc(1,2)+(kc(10)+kc(5))*dc(2,2)*dc(2,2)
          + (kc(9)+kc(6))*dc(3,2)*dc(3,2)
sk(5,6) = kc(4)*dc(1,2)*dc(1,3)+(kc(10)+kc(5))*dc(2,2)*dc(2,3)
          + (kc(9)+kc(6))*dc(3,2)*dc(3,3)
sk(5,7) = -sk(1,5)
sk(5,8) = -sk(2,5)
sk(5,9) = -sk(3,5)
sk(5,10) = sk(4,11)
sk(5,11) = -kc(4)*dc(1,2)*dc(1,2)+(kc(10)-kc(5))*dc(2,2)*dc(2,2)
           + (kc(9)-kc(6))*dc(3,2)*dc(3,2)
sk(5,12) = -kc(4)*dc(1,2)*dc(1,3)+(kc(10)-kc(5))*dc(2,2)*dc(2,3)
           + (kc(9)-kc(6))*dc(3,2)*dc(3,3)
sk(6,6) = kc(4)*dc(1,3)*dc(1,3)+(kc(10)+kc(5))*dc(2,3)*dc(2,3)
          + (kc(9)+kc(6))*dc(3,3)*dc(3,3)
sk(6,7) = -sk(1,6)
sk(6,8) = -sk(2,6)
sk(6,9) = -sk(3,6)
sk(6,10) = sk(4,12)
sk(6,11) = sk(5,12)
sk(6,12) = -kc(4)*dc(1,3)*dc(1,3)+(kc(10)-kc(5))*dc(2,3)*dc(2,3)
           + (kc(9)-kc(6))*dc(3,3)*dc(3,3)
sk(7,7) = sk(1,1)

```

```

sk(7,8) = sk(1,2)
sk(7,9) = sk(1,3)
sk(7,10) = -sk(1,4)
sk(7,11) = -sk(1,5)
sk(7,12) = -sk(1,6)
sk(8,8) = sk(2,2)
sk(8,9) = sk(2,3)
sk(8,10) = -sk(2,4)
sk(8,11) = -sk(2,5)
sk(8,12) = -sk(2,6)
sk(9,9) = sk(3,3)
sk(9,10) = -sk(3,4)
sk(9,11) = -sk(3,5)
sk(9,12) = -sk(3,6)
sk(10,10) = sk(4,4)
sk(10,11) = sk(4,5)
sk(10,12) = sk(4,6)
sk(11,11) = sk(5,5)
sk(11,12) = sk(5,6)
sk(12,12) = sk(6,6)

```

```

do 50 i = 1,12
  do 55 j = 1,12
    sm(i,j) = 0.d0
55   continue
50   continue

```

C Row-sum rotated mass matrix to re-diagonalize

```

sm(1,1) = mc(1)
sm(2,2) = mc(1)
sm(3,3) = mc(1)
sm(4,4) = mc(2)*(dc(1,1)*dc(1,1)+dc(2,1)*dc(2,1)) +
.         mc(3)*(dc(1,1)*dc(1,1)+dc(3,1)*dc(3,1)) +
.         mc(2)*(dc(1,1)*dc(1,2)+dc(2,1)*dc(2,2)) +
.         mc(3)*(dc(1,1)*dc(1,2)+dc(3,1)*dc(3,2)) +
.         mc(2)*(dc(1,1)*dc(1,3)+dc(2,1)*dc(2,3)) +
.         mc(3)*(dc(1,1)*dc(1,3)+dc(3,1)*dc(3,3))
C   sm(4,5) = mc(2)*(dc(1,1)*dc(1,2)+dc(2,1)*dc(2,2)) +
C   .         mc(3)*(dc(1,1)*dc(1,2)+dc(3,1)*dc(3,2))
C   sm(4,6) = mc(2)*(dc(1,1)*dc(1,3)+dc(2,1)*dc(2,3)) +
C   .         mc(3)*(dc(1,1)*dc(1,3)+dc(3,1)*dc(3,3))
C   sm(5,4) = mc(2)*(dc(1,1)*dc(1,2)+dc(2,1)*dc(2,2)) +
C   .         mc(3)*(dc(1,1)*dc(1,2)+dc(3,1)*dc(3,2))
sm(5,5) = mc(2)*(dc(1,2)*dc(1,2)+dc(2,2)*dc(2,2)) +
.         mc(3)*(dc(1,2)*dc(1,2)+dc(3,2)*dc(3,2)) +
.         mc(2)*(dc(1,1)*dc(1,2)+dc(2,1)*dc(2,2)) +
.         mc(3)*(dc(1,1)*dc(1,2)+dc(3,1)*dc(3,2)) +
.         mc(2)*(dc(1,2)*dc(1,3)+dc(2,2)*dc(2,3)) +
.         mc(3)*(dc(1,2)*dc(1,3)+dc(3,2)*dc(3,3))
C   sm(5,6) = mc(2)*(dc(1,2)*dc(1,3)+dc(2,2)*dc(2,3)) +
C   .         mc(3)*(dc(1,2)*dc(1,3)+dc(3,2)*dc(3,3))
C   sm(6,4) = mc(2)*(dc(1,1)*dc(1,3)+dc(2,1)*dc(2,3)) +
C   .         mc(3)*(dc(1,1)*dc(1,3)+dc(3,1)*dc(3,3))
C   sm(6,5) = mc(2)*(dc(1,2)*dc(1,3)+dc(2,2)*dc(2,3)) +
C   .         mc(3)*(dc(1,2)*dc(1,3)+dc(3,2)*dc(3,3))
sm(6,6) = mc(2)*(dc(1,3)*dc(1,3)+dc(2,3)*dc(2,3)) +
.         mc(3)*(dc(1,3)*dc(1,3)+dc(3,3)*dc(3,3)) +

```

```

      mc(2)*(dc(1,1)*dc(1,3)+dc(2,1)*dc(2,3)) +
      mc(3)*(dc(1,1)*dc(1,3)+dc(3,1)*dc(3,3)) +
      mc(2)*(dc(1,2)*dc(1,3)+dc(2,2)*dc(2,3)) +
      mc(3)*(dc(1,2)*dc(1,3)+dc(3,2)*dc(3,3))
      sm(7,7) = mc(1)
      sm(8,8) = mc(1)
      sm(9,9) = mc(1)
      sm(10,10) = sm(4,4)
C      sm(10,11) = sm(4,5)
C      sm(10,12) = sm(4,6)
C      sm(11,10) = sm(5,4)
      sm(11,11) = sm(5,5)
C      sm(11,12) = sm(5,6)
C      sm(12,10) = sm(6,4)
C      sm(12,11) = sm(6,5)
      sm(12,12) = sm(6,6)

      do 100 i=1,12
        do 200 j=1,i-1
          sk(i,j) = sk(j,i)
200      continue
100     continue

      return
      end
C=End Fortran

```

File: forces.f

```

C=Module FORCES
C=Purpose Calculate applied force vector at given time
C=Author K. Alvin
C=Date May 1990
C=Block Fortran
C -----
C
C      Subroutine FORCES
C
C      Purpose:
C      Returns force from stored function at any given time.
C      The forcing functions are hardwired by the user. The
C      function is selectable at program execution using the
C      forcing function ID, which by convention is the statement
C      label used in branching.
C -----
C
C
C      subroutine FORCES(time)
C
C      include 'shared.inc'
C      real*8 time,pi
C
C      LOGIC
C
C      pi = 3.1415926

```

```

      call ZEROVECT(f,ndof)
101  if (forceid .eq. 101) then
      if (time .le. .02) then
        f(id(2,15)) = 100.*(1.-cos(2.*pi*time/.02))
      endif
102  elseif (forceid .eq. 102) then
      if (time .lt. .1) then
        f(id(2,95)) = 10.
      elseif (time .eq. .1) then
        f(id(2,95)) = 0.
      elseif ((time .gt. .1).and.(time .lt. .2)) then
        f(id(2,95)) = -10.
      else
        f(id(2,95)) = 0.
      endif
103  elseif (forceid .eq. 103) then
      if (time .le. .01) then
        f(id(1,15)) = 100
      endif
104  elseif (forceid .eq.104) then
      if ((time .gt. 0) .and. (time .lt. .17)) then
        f(id(3,125)) = 10
      elseif ((time .gt. .17) .and. (time .lt. 1.0)) then
        f(id(3,125)) = -10
      endif
105  elseif (forceid .eq. 105) then
      if (time .le. .01) then
        f(id(2,9)) = 100.*(1.-cos(2.*pi*time/.01))
      elseif (time .le. .02) then
        f(id(2,9)) = 100.*(cos(2.*pi*time/.01)-1.)
      endif
    endif
    do 10 i = 1, ndof
      f(i) = scalef * f(i)
10    continue
    return
  end

```

File: input.f

C=Module INPUT
 C=Purpose Input data parameters for ACSIS
 C=Author K. Alvin

C=Date May 1990

C=Block Fortran

```
C -----
C
C
C   Subroutine INPUT
C
C -----
C
C   Argument definitions
C
C   runtime - ID of analysis run type
C   savin   - variable to control creation of input file
C   runfile - variable indicates if run is from input file
C   comment - dummy name for comment input lines
C   outskip - number of steps to skip before sending output
C
C   subroutine INPUT(runtime,outskip)
C
C   include 'shared.inc'
C   integer runtime, outskip
C   character*1 savin,runfile,temp
C   character*48 comment,inpfile
C
C   PRINT AND READ START-UP
C
C   print *, '2nd Order Accelerated CSI Simulation (ACSIS)'
C   print *
C   print *, 'Please input analysis type:'
C   print *
C   print *, '    1. Eigenmode Analysis'
C   print *, '    2. CSI Simulation'
C   print *, '    3. Transient Response'
C   print *
C   read *, runtime
C
C   RUN OPTIONS AND INPUT FILE SETUP
C
C   runfile = 'n'
C   if (runtime .lt. 0) then
C       runtime = -1 * runtime
C       runfile = 'y'
C   endif
C   print *, 'Do you wish to save an input file? (y or n)'
C   read 21, savin
20  format (a32)
21  format (a1)
C   if (savin .eq. 'y') then
C       print *, 'Name of save input file? (filename)'
C       read 20, inpfile
C       open(16,file=inpfile)
C       runtime = -1 * runtime
C       write(16,'(i2)') runtime
C       runtime = -1 * runtime
C       write(16,'(a1)') 'n'
C       write(16,'(a47)') '* ACSIS input file,two lines above are'
C       write(16,'(a48)') '* analysis type and save input file. Do'
C       write(16,'(a48)') '* not change them by editing this file.'
C   endif
```

```

runtype = runtype - 2

if (runfile .eq. 'y') then
  do 30 i = 1,4
    read 20, comment
30    continue
endif
print *, 'Finite Element Model Input File Name (filename)'
read 20, femfile
open(11,file=femfile)
if (savin .eq. 'y') then
  write(16,'(a47)') '* Finite element input file?(filename)'
  write(16,'(a32)') femfile
endif
if (runtype) 100,200,300

C  EIGENMODE INPUTS

100 print *, 'Number of modes desired:'
if (runfile .eq. 'y') read 20, comment
read *, neig
if (runfile .eq. 'y') read 20, comment
print *, 'Output File Name:'
read 20, outfile
open(13,file=outfile)
if (savin .eq. 'y') then
  write(16,'(a35)') '* Number of modes desired?'
  write(16,'(i4)') neig
  write(16,'(a33)') '* Output file?(filename)'
  write(16,'(a32)') outfile
endif

call READFEM

goto 1000

C  CSI INPUTS

200 print *, 'Controller Definition File Name:'
if (runfile .eq. 'y') read 20, comment
read 20, confile
open(12,file=confile)
print *, 'Please input type of control:'
print *
print *, ' 1. Full State Feedback'
print *, ' 2. Luenberger Observer (Li=0)'
print *, ' 3. Kalman Filter'
print *
if (runfile .eq. 'y') read 20, comment
read *, contype
contype = contype - 2
print *, 'Initial time, final time, control-on time, step size:'
if (runfile .eq. 'y') read 20, comment
read *, t0,tf,tc,h
print *, 'Forcing function ID, scale factor, damping coeff- a,b:'
if (runfile .eq. 'y') read 20, comment
read *, forceid,scalef,adamp,bdamp
print *, 'Phase lag fix?(y or n):'
if (runfile .eq. 'y') read 20, comment

```

```

read 21, temp
if (temp .eq. 'y') nolag = .true.
if (temp .eq. 'n') nolag = .false.
print *, 'Gain scale factors (4 total):'
if (runfile .eq. 'y') read 20, comment
read *, qalpha, qbeta, qalphao, qbetao
if (savin .eq. 'y') then
  write(16, '(a42)') '* Controller file name?(filename)'
  write(16, '(a32)') confile
  write(16, '(a42)') '* Please input type of control: '
  write(16, '(i10)') contype + 2
  write(16, '(a46)') '* Initial, final, control-on, step size?'
  write(16, '(4f14.8)') t0, tf, tc, h
  write(16, '(a49)') '* Forcing function, scale f, damping a, b?'
  write(16, '(i4, f15.6, 2f12.8)') forceid, scalef, adamp, bdamp
  write(16, '(a32)') '* Phase lag fix?(y or n)'
  if (nolag) write(16, '(a1)') 'y'
  if (.not. nolag) write(16, '(a1)') 'n'
  write(16, '(a40)') '* Gain scale factors (4 total)?'
  write(16, '(4f14.8)') qalpha, qbeta, qalphao, qbetao
endif

```

call READFEM

goto 999

C TRANSIENT RESPONSE INPUTS

```

300 print *, 'Initial time, final time, step size:'
if (runfile .eq. 'y') read 20, comment
read *, t0, tf, h
print *, 'Forcing function ID, scale factor, damping coeff- a, b:'
if (runfile .eq. 'y') read 20, comment
read *, forceid, scalef, adamp, bdamp
if (savin .eq. 'y') then
  write(16, '(a48)') '* Initial, final, step size?'
  write(16, '(3f14.8)') t0, tf, h
  write(16, '(a49)') '* Forcing function, scale f, damping a, b?'
  write(16, '(i4, f15.6, 2f12.8)') forceid, scalef, adamp, bdamp
endif

```

call READFEM

goto 999

C OUTPUT OPTIONS

```

999 print *, 'Output File Name:'
if (runfile .eq. 'y') read 20, comment
read 20, outfile
open(13, file=outfile)
print *, 'Number of displacement results to output (max 10):'
if (runfile .eq. 'y') read 20, comment
read *, ndisout
do 500 i=1, ndisout
  print *, 'Input node #, dof for displacement output#', i
  read *, outlabel(i), outlabel(i+10)
500 continue
print *, 'Number of velocity results to output (max 10):'

```

```

if (runfile .eq. 'y') read 20, comment
read *,nvelout
do 600 i=1,nvelout
    print *,'Input node $, dof for velocity output$',i
    read *,outlabel(i+20),outlabel(i+30)
600 continue
print *,'Send output every how many steps?'
if (runfile .eq. 'y') read 20, comment
read *, outskip
if (savin .eq. 'y') then
    write(16,'(a38)') '* Output file name?(filename)'
    write(16,'(a32)') outfile
    write(16,'(a42)') '* Number of displacement outputs?'
    write(16,'(i4)') ndisout
    do 650 i=1,ndisout
        write(16,'(2i8)') outlabel(i),outlabel(i+10)
650 continue
    write(16,'(a38)') '* Number of velocity outputs?'
    write(16,'(i4)') nvelout
    do 660 i=1,nvelout
        write(16,'(2i8)') outlabel(i+20),outlabel(i+30)
660 continue
    write(16,'(a44)') '* Send output every how many steps?'
    write(16,'(i3)') outskip
endif

```

C ANIMATION OPTION

```

print *,'Animation Output? (y or n):'
if (runfile .eq. 'y') read 20, comment
read 21, temp
if (temp .eq. 'y') animate = .true.
if (temp .eq. 'n') animate = .false.
if (animate) then
    print*, 'Animation file name (filename)'
    if (runfile .eq. 'y') read 20, comment
    read 20, animfile
    open (15, file=animfile)
endif
if (savin .eq. 'y') then
    write(16,'(a41)') '* Send animation output?(y or n)'
    if (animate) write(16,'(a1)') 'y'
    if (.not. animate) write(16,'(a1)') 'n'
    if (animate) then
        write(16,'(a31)') '* Animation file name?'
        write(16,'(a32)') animfile
    endif
endif

delta = h/2.
delsq = delta**2

1000 return
end

```

File: pmvmul.f

C=Module PMVMUL
 C=Author K. Alvin
 C=Date May 1990
 C=Block Fortran

C -----

C
 C Subroutine PMVMUL

C
 C Purpose:
 C This subroutine multiplies a matrix in vector form
 C and a vector.

C -----

C
 C Arguments
 C a - matrix in vector form
 C b - vector
 C c - result vector
 C neq - order of vector and square matrix
 C jdiag - array of diagonal addresses for a

C
 C subroutine PMVMUL(a,jdiag,b,neq,c)
 C recursive subroutine PMVMUL(a,jdiag,b,neq,c)

real*8 a(1), b(1), c(1)
 integer jdiag(1), neq
 do 100 i=1,neq
 c(i) = a(jdiag(i))*b(i)
 100 continue
 do 200 i=2,neq
 do 300 j=jdiag(i-1)+1,jdiag(i)-1
 k = jdiag(i) - j
 c(i) = c(i) + a(j)*b(i-k)
 300 continue
 200 continue
 do 250 i=2,neq
 do 400 j=jdiag(i-1)+1,jdiag(i)-1
 k = jdiag(i) - j
 c(i-k) = c(i-k) + a(j)*b(i)
 400 continue
 250 continue
 return
 end

C
 C -----

C
 C Subroutine PMVMAD

C
 C Purpose:
 C Multiply a matrix in vector form and a vector and add the
 C resultant vector multiplied by a constant to a second vector

```

C      multiplied by a second vector
C
C      -----
C
C      Arguments
C      a      - matrix in vector form
C      b      - vector to be multiplied with matrix
C      c      - resultant and vector to be added
C      fact1   - constant multiplier of matrix and first vector
C      fact2   - constant multiplier of second vector
C      jdiag   - array of diagonal addresses for matrix
C      neq     - order of vectors and matrix
C
C      subroutine PMVMAD(a,jdiag,b,neq,fact1,c,fact2)
C
C      recursive subroutine PMVMAD(a,jdiag,b,neq,fact1,c,fact2)
C
C      real*8 a(1), b(1),c(1),fact1,fact2
C      integer jdiag(1), neq
C
C      do 100 i=1,neq
C          c(i) = fact2*c(i) + fact1*a(jdiag(i))*b(i)
100      continue
C
C      do 200 i=2,neq
C          do 300 j=jdiag(i-1)+1,jdiag(i)-1
C              k = jdiag(i) - j
C              c(i) = c(i) + fact1*a(j)*b(i-k)
300          continue
200      continue
C
C      do 250 i=2,neq
C          do 400 j=jdiag(i-1)+1,jdiag(i)-1
C              k = jdiag(i) - j
C              c(i-k) = c(i-k) + fact1*a(j)*b(i)
400          continue
250      continue
C
C      return
C      end

```

File: prepfem.f

```

C=Module PREPFEM
C=Purpose Preprocess Structure Finite Element module for ACSIS
C=Author K. Alvin
C=Date May 1990
C=Block Fortran

```

```

C      -----
C
C      Subroutine PREPFEM
C
C      Purpose:
C      This subroutine prepares the finite element mass,
C      stiffness, and S matrices in reduced profile vector form

```

```

C
C -----
C
C Local variables:
C
C   sk      Element Stiffness matrix
C   sm      Element Mass Matrix
C   lm      Local/Global DOF Mapping vector
C   nseq    Number of element degrees of freedom
C   em,ep   Material and Propety id # for element
C
C   subroutine PREPFEM
C
C       include 'shared.inc'
C
C LOCAL VARIABLES
C
C   parameter (MAXSEQ=24)
C   real*8 sk(MAXSEQ,MAXSEQ),sm(MAXSEQ,MAXSEQ),mc,kc
C   integer lm(MAXSEQ),nseq,em,ep
C
C   call RENUM
C
C Set up skyline storage profile for global matrices
C
C   call PROFILE(ix,id,jdiag,nnp,nel,4,6,mle,n dof,mask)
C
C Perform automatic domain decomposition
C
C   call DOMDEC
C
C Check size of skyline profile against storage limitation
C
C   if (mle .gt. MAXMLEN) then
C       print*, 'PREPFEM: error, global matrix exceeded max. size'
C   endif
C
C Zero Global Matrices prior to assembly
C
C   call ZEROVECT(stif,mle)
C   call ZEROVECT(mass,mle)
C
C ASSEMBLE EACH ELEMENT MASS AND STIFFNESS
C
C   do 100 n=1,nel
C
C       do 20 k=1,4
C           j=ix(k,n)
C           if ((etype(n).eq.1).and.(k.gt.2)) j = 0
C           do 30 i=1,6
C               kk=6*(k-1) + i
C               if (j .ne. 0) then
C                   lm(kk) = id(i,j)
C               else
C                   lm(kk) = 0
C               endif
C           enddo
C       continue
C   30      continue
C   20      continue

```

```

        if (etype(n) .eq. 1) then
            nseq = 12
            em = emat(n)
            ep = eprop(n)
            call BEAN3D(n,ix(1,n),ix(2,n),ix(3,n),coxyz,mat(1,em),
                mat(2,em),mat(3,em),prop(1,ep),prop(5,ep),prop(6,ep),
                prop(2,ep),prop(3,ep),prop(4,ep),pin(1,n),sk,sm)
            elseif (ix(1,n) .ne. 0) then
                print*, 'PREPFEH Element type not found,n=',n,'etype=',etype(n)
            endif

C   ADD ELEMENT TO GLOBAL MASS AND STIFFNESS

            call ADDSTF(sk,lm,stif,jdiag,nseq)
            call ADDSTF(sm,lm,mas,jdiag,nseq)

C   SAVE THE ELEMENT STIFFNESS FOR E-BY-E COMPUTATIONS

            call SAVESK(sk,n,nseq)

100    continue

C   ADD LUMPED INERTIAS TO GLOBAL MASS

        do 125 i=1,nnp
            do 130 j=1,6
                if (id(j,i) .eq. 0) goto 130
                k=jdiag(id(j,i))
                mass(k) = mass(k) + inertia(j,i)
130        continue
125    continue

C   ASSEMBLE AND FACT .IZE es (S MATRIX)

        mc = 1. + delta*adamp
        kc = delta*bdamp + delsq
        do 200 i=1,m1en
            es(i) = mc*mass(i) + kc*stif(i)
200    continue

            call SOLVER(es,gs,jdiag,ndof,1)

C   INITIALIZE DISPLACEMENT AND VELOCITY VECTORS

        do 300 i = 1, nnp
            do 350 j = 1,6
                if (id(j,i) .ne. 0) then
                    q(id(j,i)) = q0(j,i)
                    qdot(id(j,i)) = qdot0(j,i)
                endif
350        continue
300    continue

        return
        end

        subroutine SAVESK(sk,n,nseq)

        include 'shared.inc'

```



```

real*8 sk(nseq,1)
integer n,nseq

k=0
do 10 j=1,nseq
  do 20 i=1,j
    k=k+1
    estifm(k,n)=sk(i,j)
20    continue
10    continue

return
end

subroutine DOMDEC

include 'shared.inc'

logical nchk,ndchk(MAXNODE,MAXDOM)
integer ndom

do 10 j=1,MAXDOM
  neld(j)=0
  do 20 i=1,nnp
    ndchk(i,j)=.false.
20    continue
10    continue
ndomain=0

do 100 n=1,nel

  ndom=0
  nchk=0
  do 200 while (nchk.eq.0)
    ndom=ndom+1
    if (ndom.gt.ndomain) ndomain=ndom
    nchk=1
    if (ndchk(ix(1,n),ndom)) nchk=0
    if (ndchk(ix(2,n),ndom)) nchk=0
    if (nchk.eq.1) then
      eldom(n)=ndom
      ndchk(ix(1,n),ndom)=.true.
      ndchk(ix(2,n),ndom)=.true.
    endif
200    continue

    neld(ndom)=neld(ndom)+1
    elnum(neld(ndom),ndom)=n

100    continue

return
end

```

File: profile.f

```

C=Module PROFILE
C=Purpose Compute the number of equations and set profile for K
C=Author Bob Taylor
C=Date who knows
C=Update January 1989 by E. Pramono
C=Block Fortran
      subroutine PROFILE(ix,id,jdiag,nnp,nel,nen,ndof,nad,neq,mask)

```

```

C-----+C
C  PURPOSE:
C    THIS SUBROUTINE COMPUTES THE NUMBER OF EQUATIONS REQUIRED
C    TO SOLVE THE PROBLEM BY ELIMINATING RESTRAINED DEGREES OF
C    FREEDOM FROM THE SYSTEM OF EQUATIONS. KNOWING THE EQUATION
C    NUMBERS COORESPONDING TO THE NODAL DEGREES OF FREEDOM, THE
C    DIAGONAL ELEMENT LOCATIONS CAN BE COMPUTED FOR STORING THE
C    GLOBAL STIFFNES MATRIX IN COMPACTED VECTOR FORM.
C-----C

```

```

C  ARGUMENTS

```

```

C    integer ix(nen,1), id(ndof,1), jdiag(1)
C    integer nnp, nel, nad, neq, mask(1)
C    integer nnp, nel, nen, ndof, nad, neq, mask(1)

```

```

C  LOCAL ARGUMENTS

```

```

C    integer i, j, k, l, m, n, j1, k1, l1, m1

```

```

C  SET UP EQUATION NUMBERS

```

```

C    neq = 0
C    do 30 n = 1, nnp
C      do 20 m = 1, ndof
C        j = id(m,mask(n))
C        if (j .eq. 1) goto 10
C        neq = neq + 1
C        id(m,mask(n)) = neq
C        jdiag(neq) = 0
C        goto 20
10      id(m,mask(n)) = 0
20      continue
30      continue

```

```

C  COMPUTE COLUMN HEIGHTS

```

```

C    do 80 n = 1, nel
C      do 70 m = 1, nen
C        m1 = ix(m,n)
C        if (m1 .le. 0) goto 70
C        do 60 l = 1, ndof
C          l1 = id(l,m1)
C          if (l1 .eq. 0) goto 60
C          do 50 k = m, nen
C            k1 = ix(k,n)
C            if (k1 .le. 0) goto 50
C            do 40 j = 1, ndof
C              j1 = id(j,k1)
C              if (j1 .eq. 0) goto 40

```

```

            i = MAX0(11,j1)
            jdiag(i) = MAX0(jdiag(i), IABS(11-j1))
40          continue
50          continue
60          continue
70          continue
80          continue
C
C
C COMPUTE DIAGONAL POINTERS
C
      nad = 1
      jdiag(1) = 1
      if (neq .eq. 1) return
      do 90 n = 2, neq
        jdiag(n) = jdiag(n) + jdiag(n-1) + 1
90      continue
      nad = jdiag(neq)
C
      return
      end
C=End Fortran

```

File: read.f

```

C=Module READ
C=Author K. Alvin
C=Date May 1990
C=Block Fortran

```

```

C -----
C
C Subroutine READFEM
C
C Purpose:
C   This subroutine reads the data file for the finite
C   element model.
C -----
C
C Arguments
C   ctype - stores code for type of line
C
C
C subroutine READFEM
C
C GLOBALS
C
C   include 'shared.inc'
C
C LOCALS
C
C   integer j,n,ctype,GETTYPE
C   character*132 aline
C   real*8 in
C
C INITIALIZE SIZE OF PROBLEM

```

```

nnp = 0
nel = 0
ndof = 0
ndomain = 0

```

C IDENTIFY CARD TYPE AND ASSIGN INPUT

```

10  read(11,1000,end=9999) aline
100  ctype = GETTYPE(aline)
    if (ctype) 10,10,150
150  if (aline(1:4) .eq. 'NODE') goto 200
    if (aline(1:4) .eq. 'TOPO') goto 300
    if (aline(1:4) .eq. 'ATTR') goto 400
    if (aline(1:4) .eq. 'MATE') goto 500
    if (aline(1:4) .eq. 'PROP') goto 600
    if (aline(1:4) .eq. 'FIXI') goto 700
    if (aline(1:4) .eq. 'INIT') goto 800
    if (aline(1:4) .eq. 'INER') goto 900 .
    if (aline(1:4) .eq. 'END ') goto 10
    if (aline(1:4) .eq. 'MESH') goto 10
    print *, 'READFEM: Unrecognized card type; ',aline(1:4)
    goto 10

```

C READ NODES

```

200  read(11,1000,end=9999) aline
    ctype = GETTYPE(aline)
    if (ctype) 200,250,100
250  read(aline,*) n,(coxyz(j,n),j=1,3)
    if (n .gt. nnp) nnp = n
    goto 200

```

C READ TOPOLOGY

```

300  read(11,1000,end=9999) aline
    ctype = GETTYPE(aline)
    if (ctype) 300,350,100
350  read(aline,*) n,etype(n),(ix(j,n),j=1,4)
    if (n .gt. nel) nel = n
    goto 300

```

C READ ATTRIBUTES

```

400  read(11,1000,end=9999) aline
    ctype = GETTYPE(aline)
    if (ctype) 400,450,100
450  read(aline,*) n,emat(n),eprop(n),(pin(j,n),j=1,6)
    if (eldom(n).gt.ndomain) ndomain=eldom(n)
    goto 400

```

C READ MATERIAL

```

500  read(11,1000,end=9999) aline
    ctype = GETTYPE(aline)
    if (ctype) 500,550,100
550  read(aline,*) n,(mat(j,n),j=1,3)
    goto 500

```

C READ PROPERTIES

```

600  read(11,1000,end=9999) aline
      ctype = GETTYPE(aline)
      if (ctype) 600,650,100
650  read(aline,*) n,(prop(j,n),j=1,6)
      goto 600

C    READ FIXITY

700  read(11,1000,end=9999) aline
      ctype = GETTYPE(aline)
      if (ctype) 700,750,100
750  read(aline,*) n,(id(j,n),j=1,6)
      goto 700

C    READ INITIAL CONDITIONS

800  read(11,1000,end=9999) aline
      ctype = GETTYPE(aline)
      if (ctype) 800,850,100
850  read(aline,*) n,j,q0(j,n),qdot0(j,n)
      goto 800

C    READ INERTIA

900  read(11,1000,end=9999) aline
      ctype = GETTYPE(aline)
      if (ctype) 900,950,100
950  read(aline,*) n,j,in
      inertia(j,n)=inertia(j,n)+in
      goto 900

1000 format(a132)
9999 continue

      return
      end

C
C -----
C
C    Subroutine READCON
C
C    Purpose:
C      This subroutine reads the actuator and sensor
C      locations and the gains for the control system
C
C -----
C
C    Arguments
C      ctype - stores code for type of line
C
C
C    Abbreviations
C      NACT - number of actuators
C      NSEN - number of sensors
C      BNAT - locations of actuators
C      HDMA - array of displacement sensor locations
C      HVMA - array of velocity sensor locations
C      FIGA - control gain matrix

```

```

C      F2GA - control gain matrix
C      L1GA - state estimator filter gain matrix
C      L2GA - state estimator filter gain matrix

```

```

subroutine READCON

```

```

include 'shared.inc'

```

```

C      LOCALS

```

```

real*8 val
integer j,n,ctype,GETTYPE
character*132 aline

```

```

bval = 0
hdval = 0
hvval = 0
hdbval = 0
hvbval = 0

```

```

C      IDENTIFY CARD TYPE AND ASSIGN INPUT

```

```

10  read(12,1000,end=9999) aline
100  ctype = GETTYPE(aline)
    if (ctype) 10,10,150
150  if (aline(1:4) .eq. 'NACT') goto 200
    if (aline(1:4) .eq. 'NSEN') goto 300
    if (aline(1:4) .eq. 'BNAT') goto 400
    if (aline(1:4) .eq. 'HDMA') goto 500
    if (aline(1:4) .eq. 'HVMA') goto 600
    if (aline(1:4) .eq. 'F1GA') goto 700
    if (aline(1:4) .eq. 'F2GA') goto 800
    if (aline(1:4) .eq. 'L2GA') goto 900
    if (aline(1:4) .eq. 'L1GA') goto 1100
    if (aline(1:4) .eq. 'END ') goto 10
    print *, 'READCON: Unrecognized card type; ',aline(1:4)
    goto 10

```

```

C      READ INPUT CARDS

```

```

200  read(12,1000,end=9999) aline
    ctype = GETTYPE(aline)
    if (ctype) 200,250,100
250  read(aline,*) nact
    goto 200

300  read(12,1000,end=9999) aline
    ctype = GETTYPE(aline)
    if (ctype) 300,350,100
350  read(aline,*) nsen
    goto 300

400  read(12,1000,end=9999) aline
    ctype = GETTYPE(aline)
    if (ctype) 400,450,100
450  read(aline,*) i,j,n,val
    bval = bval + 1
    b(bval) = val
    brow(bval) = id(j,i)

```

```

        bcol(bval) = n
        goto 400

500  read(12,1000,end=9999) aline
      ctype = GETTYPE(aline)
      if (ctype) 500,550,100
550  read(aline,*) i,j,n,val
      hdval = hdval + 1
      hd(hdval) = val
      hdrow(hdval) = n
      hdc(hdval) = id(j,i)
      goto 500

600  read(12,1000,end=9999) aline
      ctype = GETTYPE(aline)
      if (ctype) 600,650,100
650  read(aline,*) i,j,n,va
      hvval = hvval + 1
      hv(hvval) = val
      hvrow(hvval) = n
      hvcol(hvval) = id(j,i)
      goto 600

700  read(12,1000,end=9999) aline
      ctype = GETTYPE(aline)
      if (ctype) 700,750,100
750  read(aline,*) i,j,n,val
      f1(n,id(j,i)) = qalpha*val
      goto 700

800  read(12,1000,end=9999) aline
      ctype = GETTYPE(aline)
      if (ctype) 800,850,100
850  read(aline,*) i,j,n,val
      f2(n,id(j,i)) = qbeta*val
      goto 800

900  read(12,1000,end=9999) aline
      ctype = GETTYPE(aline)
      if (ctype) 900,950,100
950  read(aline,*) i,j,n,val
      l2(id(j,i),n) = qbetao*val
      goto 900

1100 read(12,1000,end=9999) aline
      ctype = GETTYPE(aline)
      if (ctype) 1100,1150,100
1150 read(aline,*) i,j,n,val
      l1(id(j,i),n) = qalphao*val
      goto 1100

1000 format(a132)
9999 continue
      return
      end

```

C
C
C

```

C      Function GETTYPE
C
C      Purpose:
C      This function identifies whether a line is a character
C      input, data input or comment.
C
C      -----
C
C      function GETTYPE(string)
C
C      GLOBALS
C
C      character*132 string
C
C      LOCALS
C
C      integer GETTYPE,ctype(10)
C      character*1 head(10)
C
C      data head /' ','@','*','$','%','&','*','C','c',' ' /
C      data ctype /-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,0/
C
C      LOGIC
C
C      GETTYPE=1
C      do 100 i=1,10
C      if (string(1:i) .eq. head(i)) GETTYPE=ctype(i)
100    continue
C
C      return
C      end

```

File: solver.f

```

C=Module SOLVER
C=Purpose Solves the system of linear symmetric equations
C=Author who knows
C=Date
C=Update January 1989 by E. Pramono
C=Block Fortran
C      SUBROUTINE SOLVER(BK,BR,JDIAG,NEQ,IFLAG)
C      recursive SUBROUTINE SOLVER(BK,BR,JDIAG,NEQ,IFLAG)
C+-----+C
C      PURPOSE:
C      THIS SUBROUTINE SOLVES THE SYSTEM OF LINEAR SYMMETRIC
C      EQUATIONS IN VECTOR FORM USING THE CROUT REDUCTION
C      METHOD.
C
C      ARGUMENTS:
C      BK      - GLOBAL STIFFNESS EQUATIONS IN VECTOR FORM
C      BR      - GLOBAL LOAD VECTOR
C      JDIAG   - LOCATION VECTOR FOR DIAGONALS IN [BK]
C      NEQ     - NUMBER OF EQUATIONS
C      IFLAG   - FLAG INDICATING WHICH FUNCTION IS TO BE PERFORMED
C      1 -> FORWARD REDUCTION
C      2 -> BACKWARD SUBSTITUTION

```


C-----C

C ARGUMENTS

REAL*8 BK(1), BR(1)
INTEGER JDIAG(1), NEQ, IFLAG

C LOCAL VARIABLES

REAL*8 ZERO, EZERO, TOL, DAVAL, DOT, D, RDD, DD
INTEGER LDFLAG, JR, J, JD, JH, IS, IE, K, JDT
INTEGER JJ, ID, I, IR, IH

JJ = 6

C

C

C NEW PARAMETERS

C

ZERO = 0.0D0
EZERO = 0.3D-14
TOL = 0.5D-7
LDFLAG = 0

C

C

C FACTOR BK TO UT*D*U OR REDUCE R

C

JR = 0
DO 70 J = 1, NEQ
JD = JDIAG(J)
JH = JD - JR
IS = J - JH + 2
IF (JH - 2) 60, 30, 10

C

10

IF (IFLAG .NE. 1) GOTO 50
IE = J - 1
K = JR + 2
ID = JDIAG(IS-1)

C

C

C IF DIAGONAL IS ZERO COMPUTE A NORM FOR SINGULARITY TEST

C

JDT = JDIAG(IE) + 1
IF (BK(JD) .EQ. ZERO .AND. IFLAG .EQ. 1) THEN
CALL DATEST (BK(JDT), JH-2, DAVAL)
END IF

C

C

C REDUCE ALL EQUATIONS EXCEPT FIRST ROW AND DIAGONAL

C

DO 20 I = IS, IE
IR = ID
ID = JDIAG(I)
IH = MINO(ID-IR-1, I-IS+1)
IF (IH .GT. 0) BK(K) = BK(K) - DOT(BK(K-IH), BK(ID-IH), IH)
K = K + 1

20

CONTINUE

C

C

C REDUCE FIRST ROW AND DIAGONAL

```

C -----
30  IF (IFLAG .NE. 1) GOTO 50
    IR = JR + 1
    IE = JD - 1
    K = J - JD
    DD = BK(JD)
    DO 40 I = IR, IE
        ID = JDIAG(K+I)
        IF (BK(ID) .EQ. 0.0) GOTO 40
        D = BK(I)
        BK(I) = BK(I)/BK(ID)
        BK(JD) = BK(JD) - D*BK(I)
40  CONTINUE
C
C -----
C  CHECK FOR POSSIBLE ERRORS AND PRINT WARNINGS
C -----
    RDD = BK(JD)
    IF (DABS(RDD) .LT. TOL*DABS(DD)) WRITE (JJ,2000) J
    IF (DD .LT. ZERO .AND. RDD .GT. ZERO) WRITE (JJ,2001) J
    IF (DD .GT. ZERO .AND. RDD .LT. ZERO) WRITE (JJ,2001) J
    IF (DABS(RDD) .LT. EZERO) WRITE (JJ,2002) J
C
C -----
C  COMPLETE RANK TEST FOR A ZERO DIAGONAL TEST
C -----
    IF (DD .EQ. ZERO .AND. JH .GT. 0) THEN
        IF (DABS(RDD) .LT. TOL*DAVAL) WRITE (JJ,2003) J
    END IF
C
C -----
C  REDUCE RIGHT HAND SIDE
C -----
50  IF (IFLAG .EQ. 2) BR(J) = BR(J) - DOT(BK(JR+1), BR(IS-1), JH-1)
60  JR = JD
70  CONTINUE
    IF (IFLAG .NE. 2) RETURN
C
C -----
C  DIVIDE BY DIAGONAL TERMS
C -----
    DO 80 I = 1, NEQ
        ID = JDIAG(I)
        IF (BK(ID) .NE. 0.0) BR(I) = BR(I)/BK(ID)
        IF (BR(I) .NE. ZERO) LDFLAG = 1
80  CONTINUE
C
C -----
C  CHECK FOR ZERO LOAD VECTOR
C -----
    IF (LDFLAG .EQ. 0) WRITE(JJ,2004)
C
C -----
C  BACK SUBSTITUTE
C -----
    J = NEQ
    JD = JDIAG(J)
90  D = BR(J)
    J = J - 1

```

```

      IF (J .LE. 0) RETURN
      JR = JDIAG(J)
      IF (JD - JR .LE. 1) GOTO 110
      IS = J - JD + JR + 2
      K = JR - IS + 1
      DO 100 I= IS, J
        BR(I) = BR(I) - BK(I+K)*D.
10C   CONTINUE
110   JD = JR
      GOTO 90

C
C -----
C   WARNING FORMATS
C -----
2000  FORMAT('!!! WARNING !! 1 - IN SOLVER, LOSS OF AT LEAST 7 DIGITS'
+         /18X, 'IN REDUCING DIAGONAL OF EQUATION;',4X,I5)
2001  FORMAT('!!! WARNING !! 2 - IN SOLVER, SIGN OF DIAGONAL CHANGED'
+         /18X, 'WHEN REDUCING EQUATION;',15X,I5)
2002  FORMAT('!!! WARNING !! 3 - IN SOLVER, REDUCED DIAGONAL IS ZERO'
+         /18X, 'FOR EQUATION;',25X,I5)
2003  FORMAT('!!! WARNING !! 4 - IN SOLVER, RANK FAILURE FOR A ZERO'
+         /18X, 'UNREDUCED DIAGONAL IN EQUATION;',7X,I5)
2004  FORMAT('!!! WARNING !! 5 - IN SOLVER, ZERO LOAD VECTOR')
C
      END
C=End Fortran
C=Module DATEST
C=Block Fortran
C   SUBROUTINE DATEST(A,JH,DAVAL)
      recursive SUBROUTINE DATEST(A,JH,DAVAL)
C+-----+C
C
C   TEST FOR RANK
C
C   INPUTS;
C       A(J) - COLUMN OF UNREDUCED ELEMENTS IN ARRAY
C       JH   - NUMBER OF ELEMENTS IN COLUMN
C
C   OUTPUTS;
C       DAVAL - SUM OF ABSOLUTE VALUES
C+-----+C
C
C   ARGUMENTS
C
C       REAL*8 A(1), DAVAL
C       INTEGER JH
C
C   LOCAL ARGUMENTS
C
C       INTEGER J
C
C       DAVAL = 0.0D0
C       DO 10 J = 1, JH
C         DAVAL = DAVAL + DABS(A(J))
10C   CONTINUE
C
C       RETURN
C       END

```

```

C
C
C=End Fortran
C=Module DOT
C=Block Fortran
C    FUNCTION DOT(A,B,N)
C        recursive FUNCTION DOT(A,B,N)
C+-----+C
C    PURPOSE:                                C
C        THIS FUNCTION SUBROUTINE PERFORMS THE DOT PRODUCT OF TWO    C
C        VECTORS.                                                         C
C                                                                    C
C    ARGUMENTS:                                                            C
C        A  - FIRST VECTOR INVOLVED IN DOT PRODUCT                     C
C        B  - SECOND VECTOR INVOLVED IN DOT PRODUCT                   C
C        N  - NUMBER OF ELEMENTS IN EACH OF THE TWO VECTORS          C
C-----C
C
C        REAL*8 DOT, A(1), B(1)
C        INTEGER N
C
C        INTEGER I
C
C        DOT = 0.0
C        DO 10 I = 1, N
C            DOT = DOT + A(I)*B(I)
10    CONTINUE
C
C        RETURN
C        END
C=End Fortran

```

File: nophlag.f

```

C=Module NOPHLAG
C=Author K. Alvin
C=Date July 1990
C=Block Fortran
C -----
C
C    Subroutine NOPHLAG
C
C    Purpose:
C        This subroutine solves for the structural displacement
C        and velocity vectors at the half-step for the phase lag
C        correction loop, and gets new measurement
C
C -----
C
C    Arguments
C        delbeta - delta * bdamp
C
C
C    subroutine NOPHLAG(zp)
C
C        real*8 zp(1)
C        include 'shared.inc'

```

```

C    LOCAL VARIABLES

      integer i
      real*8 v(MAXDOF),delbeta

C    LOGIC
C    ADD APPLIED FORCES TO RHS AND PREPARE MASS MULTIPLIER

      do 10 i=1,ndof
        gs(i) = gs(i) + f(i)
        v(i) = (1. + delta*adamp)*q(i) + delta*qdot(i)
10     continue

C    SOLVE FOR RIGHT HAND SIDE, gs

      do 77 i = 1,ndof
        gs(i) = delsq*gs(i) + v(i)*mass(jdiag(i))
77     continue

      if (bdamp .ne. 0.) then
        delbeta = delta*bdamp
        call PHVMAD(stif,jdiag,q,ndof,delbeta,gs,1.d0)
      endif

C    SOLVE FOR DISPLACEMENT, q, USING RHS AND MATRIX S

      call SOLVER(es,gs,jdiag,ndof,2)

      do 100 i=1,ndof
        v(i) = (gs(i)-q(i))/delta
100    continue

      call ZEROVECT(zp,nzen)

      do 200 jj = 1,hdval
        i = hrow(jj)
        j = hcol(jj)
        zp(i) = zp(i) + hd(jj)*gs(j)
200    continue
      do 250 jj = 1,hvval
        i = hrow(jj)
        j = hvcol(jj)
        zp(i) = zp(i) + hv(jj)*v(j)
250    continue

      return
      end

```

File: zerovect.f

C=Module ZEROVECT
 C=Purpose Initialize vector of given length to zero
 C=Author K. Alvin
 C=Date May 1990
 C -----

```

C
C   Subroutine ZEROVECT
C

```

```

C -----
C
      subroutine ZEROVECT(v,n)

      real*8 v(1)
      integer n

      do 100 i=1,n
         v(i) = 0.d0
100    continue

      return
      end

```

File: lu.f

```

      SUBROUTINE LUFAC(A,N,PIVOT,DET,IER,NMAX)
C*****
C
C   SUBROUTINE FACTOR USES GAUSSIAN ELIMINATION WITH
C   PARTIAL PIVOTING AND IMPLICIT SCALING TO DETERMINE
C   THE L*U DECOMPOSITION OF A SQUARE MATRIX "A" OF
C   ORDER N. THE ALGORITHM ALSO FINDS THE DETERMINENT
C   OF "A". UPON COMPLETION, THE ELEMENTS OF THE UPPER
C   TRIANGULAR MATRIX "U" ARE CONTAINED IN THEIR RESPECTIVE
C   LOCATIONS IN MATRIX "A". THE ELEMENTS OF MATRIX "L"
C   ARE CONTAINED IN THE LOWER TRIANGULAR PORTION OF "A",
C   BUT ARE SCRAMBLED WITH RESPECT TO "U" BECAUSE OF ROW
C   INTERCHANGE OPERATIONS NOT PERFORMED ON THE ELEMENTS
C   OF "L". THE VECTOR PIVOT (SEE BELOW) MUST BE USED TO
C   UNSCRAMBLE "L" IF IT IS TO BE USED FOR OTHER OPERATIONS.
C
C   VARIABLES:  A=FULL SQUARE MATRIX (DOUBLE PRECISION)
C               N=ORDER OF MATRIX A (INTEGER)
C               PIVOT=VECTOR CONTAINING A RECORD OF
C                   ROW INTERCHANGES. THE INTEGER
C                   VALUE PIVOT(K) IS THE ROW WHICH
C                   WAS INTERCHANGED WITH ROW K AT
C                   FORWARD ELIMINATION STEP K. (INTEGER)
C               DET=DETERMINENT OF MATRIX A (DOUBLE PRECISION)
C               IER=ERROR FLAG. IF IER=1, THE MATRIX A WAS FOUND
C                   TO BE SINGULAR, AND THE ROUTINE WAS EXITED. IF
C                   IER=0, THE DECOMPOSITION WAS SUCCESSFUL.
C
C*****
      INTEGER PIVOT(1),IER,N,I,J,K,IO,NMAX
      REAL*8 A(NMAX,1),S(1000),C(1000),DET,TEMP
      DET=1.0D0
C*****
C
C   FIND THE ROW NORMALIZING COEFFICIENTS S(I) FOR IMPLICIT SCALING.
C   EXIT ROUTINE IF ANY S(I)=0.0

```

```

C
C*****
DO 100 I=1,N
  S(I)=0.000
  DO 110 J=1,N
    IF (ABS(A(I,J)).GT.S(I)) S(I)=ABS(A(I,J))
110  CONTINUE
    IF (S(I).EQ.0) THEN
      IER=1
      DET=0.000
      RETURN
    END IF
100  CONTINUE
C*****
C
C    START FORWARD ELIMINATION STEP K
C
C*****
DO 120 K=1,N-1
C*****
C
C    DETERMINE PIVOT ELEMENT A(IO,K) BY FINDING THE ROW IO
C    BETWEEN K AND N CONTAINING THE MAXIMUM NORMALIZED
C    VALUE IN COLUMN K. SET PIVOT(K)=IO
C
C*****
C(K)=0.000
DO 130 I=K,N
  TEMP=ABS(A(I,K))/S(I)
  IF (TEMP.GT.C(K)) THEN
    C(K)=TEMP
    IO=I
  END IF
130  CONTINUE
  PIVOT(K)=IO
C*****
C
C    EXIT ROUTINE IF ALL VALUES IN COLUMN K AT OR BELOW
C    THE MAIN DIAGONAL ARE EQUAL TO 0.0
C
C*****
IF (C(K).EQ.0.0) THEN
  IER=1
  DET=0.000
  RETURN
END IF
C*****
C
C    INTERCHANGE ROWS IO AND K FOR COLUMNS K TO N. SKIP IF IO=K.
C    SET DET=-DET IF ROWS ARE INTERCHANGED.
C
C*****
IF (IO.EQ.K) GOTO 150
DET=-1.000*DET
DO 140 J=K,N
  TEMP=A(K,J)
  A(K,J)=A(IO,J)
  A(IO,J)=TEMP
140  CONTINUE

```

```

C*****
C
C   ELIMINATE COLUMN K BELOW MAIN DIAGONAL BY MULTIPLYING
C   ROW K FROM COLUMN K TO N BY A(I,K)/A(K,K) AND SUBTRACTING
C   FROM ROW I. STORE THE MULTIPLIER FOR ROW I IN THE ELIMINATED
C   COLUMN K. MULTIPLY THE RUNNING PRODUCT DET BY DIAGONAL ELEMENT A(K,K).
C
C*****
150  DO 160 I=K+1,N
      A(I,K)=A(I,K)/A(K,K)
      DO 170 J=K+1,N
        A(I,J)=A(I,J)-A(I,K)*A(K,J)
170    CONTINUE
160    CONTINUE
      DET=DET*A(K,K)
120  CONTINUE
C*****
C
C   CHECK LAST ROW/COLUMN FOR SINGULARITY. IF THERE IS NO ERROR,
C   COMPLETE CALCULATION OF THE DETERMINANT, SET THE ERROR FLAG
C   TO INDICATE NORMAL COMPLETION, AND EXIT.
C
C*****
      IF (A(N,N).EQ.0.0) THEN
        IER=1
        DET=0.000
        RETURN
      END IF
      DET=DET*A(N,N)
      IER=0
      RETURN
      END

      SUBROUTINE UNSOLVE(A,N,B,PIVOT,NMAX)
C*****
C
C   SUBROUTINE SOLVE
C
C*****
      INTEGER PIVOT(1),N,I,J,K,NMAX
      REAL*8 A(NMAX,1),Z(1),TEMP
      DO 100 K=1,N-1
        IF (PIVOT(K).EQ.K) GOTO 110
        TEMP=B(K)
        B(K)=B(J)
        B(J)=TEMP
110    DO 120 I=K+1,N
          B(I)=B(I)-A(I,K)*B(K)
120    CONTINUE
100    CONTINUE
      B(N)=B(N)/A(N,N)
      DO 130 I=N-1,1,-1
        DO 140 J=I+1,N
          B(I)=B(I)-A(I,J)*B(J)
140    CONTINUE
        B(I)=B(I)/A(I,I)
130    CONTINUE
      RETURN
      END

```



```

C=END FORTRAN
C=DECK FACTA
C=PURPOSE - Factors the A matrix as L U = A, with partial pivoting
C=AUTHOR W K BELVIN, Sept. 24, 1987
C
C -----
C      Input
C      amat----[n X n] matrix to be factored, destroyed on output
C      np-----problem size
C
C      Output
C      amat----contains the LU decomposition
C
C -----
C      subroutine FACTA(amat,np,nrow,lp)
C
C      real*8 amat(*),eta
C      integer lp(*)
C
C      do 50 i=1,np
50    lp(i)=i
C
C      Find largest pivot.
C
C      do 100 k=1,np-1
C      amaxk=0.
C      mmax=0
C      do 200 m=k,np
C      if (abs(amat(lp(m)+(k-1)*nrow)) .gt. amaxk) then
C      amaxk=abs(amat(lp(m)+(k-1)*nrow))
C      mmax=m
C      endif
200    continue
C
C      l=lp(k)
C      lp(k)=lp(mmax)
C      lp(mmax)=l
C
C      do 400 i=k+1,np
C      eta=amat(lp(i)+(k-1)*nrow)/amat(lp(k)+(k-1)*nrow)
C      amat(lp(i)+(k-1)*nrow)=eta
C      do 500 j=k+1,np
C      amat(lp(i)+(j-1)*nrow)=amat(lp(i)+(j-1)*nrow)-
C      eta*amat(lp(k)+(j-1)*nrow)
500    continue
400    continue
C
C      100 continue
C
C      return
C      end
C
C END FORTRAN
C=DECK LUSOLV
C=PURPOSE - Solve L U x = b,
C=AUTHOR W K BELVIN, Sept. 24, 1987
C
C -----

```

```

c   First solves  $L y = b$ , then  $U x = y$ 
c
c   Input
c       amat----[n X n] matrix factored by FACTA
c       np-----problem size
c       lp      --pointer vector based on pivoting
c       rhs----RHS of equation
c
c   Output
c       amat----contains the LU decomposition
c       x-----the solution vector
c
c -----
c       subroutine LUSOLV(amat,np,nrow,lp,rhs,x)
c
c       real*8 amat(*),x(*),rhs(*)
c       integer lp(*)
c
c       do 50 i=1,np
c           x(i)=rhs(i)
c       50 continue
c
c   Solve Lower System-----
c**** Outer loop
c
c       do 100 k=1,np
c           rhs(k)=x(lp(k))
c           if (rhs(k) .eq. 0.) go to 100
c
c**** Inner loop
c
c           d~ 200 i=k+1,np
c               x(lp(i))=x(lp(i))-amat(lp(i)+(k-1)*nrow)*rhs(k)
c       200 continue
c       100 continue
c
c   Solve Upper System-----
c
c       do 300 k=np,1,-1
c           x(k)=rhs(k)/amat(lp(k)+(k-1)*nrow)
c           do 400 i=1,k-1
c               rhs(i)=rhs(i)-x(k)*amat(lp(i)+(k-1)*nrow)
c       400 continue
c       300 continue
c       return
c       end

```

File: prepcon.f

```

C=Module PREPCON
C=Purpose Preprocess control analysis module for ACSIS
C=Author K. Alvin
C=Date June 1990
C=Block Fortran
C -----
C
C   Subroutine PREPCON

```

```

C
C      Purpose:
C      This subroutine prepares ec, the control prediction integration
C      matrix and eo, the observer construct matrix S ( $M + \delta t * D + \delta t^2 * K$ )
C
C      -----
C
C

```

```

C      subroutine PREPCOM

C      include 'shared.inc'

C      LOCAL VARIABLES

C      real*8 mc,kc
C      integer ier

C      LOGIC

C      call READCOM

C      Form Control Prediction Integration Coefficient Matrix
C
C      contype = -1 : Full State Feedback
C      contype = 0 : Luenberger Observer
C      contype = +1 : Kalman Filter

C      do 10 i = 1,nact + nsen
C        do 20 j = 1,nact + nsen
C          ec(i,j) = 0.d0
20      continue
10      continue

C      if (contype) 100,200,400

100     ncsi = nact

C      do 110 i = 1,nact
C        do 120 jj = 1,bval
C          j = bcol(jj)
C          k = brow(jj)
C          ec(i,j) = ec(i,j) + delta*f2(i,k)*b(jj)/mass(jdiag(k))
120      continue
110      continue

C      goto 600

200     ncsi = nact + nsen

C      do 210 i = 1,nact
C        do 220 jj = 1,bval
C          j = bcol(jj)
C          k = brow(jj)
C          ec(i,j) = ec(i,j) + delta*f2(i,k)*b(jj)/mass(jdiag(k))
220      continue
C      do 240 j = nact+1,ncsi
C        do 250 k = 1,ndof
C          ec(i,j) = ec(i,j) + delta*f2(i,k)*l2(k,j-nact)

```

```

250      continue
240      continue
210      continue
      do 260 ii = 1,hvval
        i = nact + hvrow(ii)
        do 270 jj = 1,bval
          j = bcol(jj)
          k = brow(jj)
          if (hvcoll(ii) .ne. k) goto 270
          ec(i,j) = ec(i,j) + delta*hv(ii)*b(jj)/mass(jdiag(k))
270      continue
        do 290 j = nact+1,ncsi
          k = hvcoll(ii)
          ec(i,j) = ec(i,j) + delta*hv(ii)*l2(k,j-nact)
290      continue
260      continue

      goto 600

400      ncsi = nact + nsen

      do 470 i = 1,nact
        do 480 jj = 1,bval
          j = bcol(jj)
          k = brow(jj)
          ec(i,j) = ec(i,j) + delta*f2(i,k)*b(jj)/mass(jdiag(k))
480      continue
        do 500 j = nact+1,ncsi
          do 510 k = 1,ndof
            ec(i,j) = ec(i,j) + delta*f2(i,k)*l2(k,j-nact)/
              mass(jdiag(k))
510      continue
500      continue
470      continue
      do 520 ii = 1,hvval
        i = nact + hvrow(ii)
        do 530 jj = 1,bval
          j = bcol(jj)
          k = brow(jj)
          if (hvcoll(ii) .ne. k) goto 530
          ec(i,j) = ec(i,j) + delta*hv(ii)*b(jj)/mass(jdiag(k))
530      continue
        do 550 j = nact+1,ncsi
          k = hvcoll(ii)
          ec(i,j) = ec(i,j) + delta*hv(ii)*l2(k,j-nact)/
            mass(jdiag(k))
550      continue
520      continue

600      continue

      do 1100 i = 1,ncsi
        ec(i,i) = ec(i,i) + 1.d0
1100      continue

C      FACTORIZE ec

      call FACTA(ec,ncsi,MAXCSI,pivot)

```

```

      if (ier .eq. 1) then
        print *, 'PREPCOM: Singular Matrix for Control Integration'
      endif

C     Form Observer Integration Coefficient Matrix, eo

      mc = 1. + delta*adamp
      kc = delta*bdamp + delsq
      do 1200 i=1,mlen
        eo(i) = mc*mass(i) + kc*stif(i)
1200    continue

C     FACTORIZE eo

      call SOLVER(eo,go,jdiag,ndof,1)

C     Initialize Observer States

      call ZEROVECT(qe,ndof)
      call ZEROVECT(qedot,ndof)
      call ZEROVECT(pe,ndof)

      return
      end

```

File: control.f

```

C=Module CONTROL
C=Author K. Alvin
C=Date May 1990
C=Block Fortran

```

```

C -----
C
C     Subroutine CONTROL
C
C     Purpose:
C       This subroutine carries out the numeric integration of one time
C       step of the control system.
C -----
C
C     Arguments
C       qep      - estimated displacement vector at half time step
C       qedotp   - estimated velocity vector at half time step
C       pp       - generalized momentum (f-D*qedotp-K*qep)
C       z        - measured sensor output
C
C       subroutine CONTROL(z)
C
C       include 'shared.inc'
C       real*8 z(1)
C
C     LOCAL VARIABLES
C
C       real*8 qep(MAXDOF),qedotp(MAXDOF),pp(MAXDOF),v(MAXDOF)

```

```

C    LOGIC

C    Form RHS of Control Prediction Equation Set
C
C    contype = -1 : Full State Feedback
C    contype = 0 : Luenberger Observer with L1 = 0
C    contype = +1 : Kalman Filter w/generalized momentum variable

    call ZEROVECT(gc,ncsi)

    if (contype) 100,200,300

100  continue

    do 110 i = 1,ndof
        qe(i) = q(i)
        qedot(i) = qdot(i)
110  continue

200  continue

    do 210 i = 1,ndof
        qep(i) = qe(i) + delta*qedot(i)
        qedotp(i) = qedot(i)
        pp(i) = f(i) - mass(jdiag(i))*adamp*qedotp(i)
        v(i) = qep(i) + bdamp*qedotp(i)
210  continue

    call PMVMAD(stif,jdiag,v,ndof,-1.d0,pp,1.d0)

    do 220 i=1,nact
        do 230 j = 1,ndof
            gc(i) = gc(i) - f1(i,j)*qep(j) - f2(i,j)*(qedot(j) +
                delta*pp(j)/mass(jdiag(j)))
230  continue
220  continue

    if (ncsi .eq. nact) goto 600

    do 240 i=nact+1,ncsi
        k = i - nact
        gc(i) = z(k)
240  continue

    do 245 ii = 1,hdval
        i = hdrow(ii) + nact
        j = hdcoll(ii)
        gc(i) = gc(i) - hd(ii)*qep(j)
245  continue

    do 250 ii = 1,hvval
        i = hvrow(ii) + nact
        j = hvcol(ii)
        gc(i) = gc(i) - hv(ii)*(qedot(j)+delta*pp(j)/mass(jdiag(j)))
250  continue

    goto 600

300  continue

    do 310 i = 1,ndof

```

```

      qep(i) = qe(i)
      pp(i) = f(i) - mass(jdiag(i))*adamp*qep(i)/delta
      v(i) = (1 + bdamp/delta)*qep(i)
310      continue

      call PNMAD(stif,jdiag,v,ndof,-1.d0,pp,1.d0)

      do 320 i=1,nact
        do 330 j = 1,ndof
          gc(i) = gc(i) - f1(i,j)*qep(j) - f2(i,j)*(pe(j) +
            delta*pp(j))/mass(jdiag(j))
330          continue
320        continue
        do 340 i=nact+1,ncsi
          k = i - nact
          gc(i) = z(k)
340          continue
          do 345 ii = 1,hdval
            i = hdrow(ii) + nact
            j = hddcol(ii)
            gc(i) = gc(i) - hd(ii)*qep(j)
345          continue
          do 350 ii = 1,hvval
            i = hvrow(ii) + nact
            j = hvcol(ii)
            gc(i) = gc(i) - hv(ii)*(pe(j) + delta*pp(j))/mass(jdiag(j))
350          continue

C      FIND r, CONTROL AND STATE CORRECTION FORCES
600      call LUSOLV(ec,ncsi,MAXCSI,pivot,gc,r)

      do 610 j=1,nact
        u(j) = r(j)
610      continue
      do 620 j=nact+1,ncsi
        gamma(j-nact) = r(j)
620      continue

C      FIND CONTROL CONTRIBUTION TO RHS VECTOR FOR
C      OBSERVER AND STRUCTAE

      do 710 i=1,ndof
        gs(i) = 0.d0
        go(i) = 0.d0
        gk(i) = 0.d0
710      continue
      do 720 jj=1,bval
        i = brow(jj)
        j = bcol(jj)
        gs(i) = gs(i) + b(jj)*u(j)
        go(i) = go(i) + b(jj)*u(j)
        gk(i) = gk(i) + b(jj)*u(j)
720      continue
      if (contype .eq. 0) then
        do 725 i = 1,ndof
          do 730 j=1,nsen
            go(i) = go(i) + mass(jdiag(i))*l2(i,j)*gamma(j)
730          continue

```

```

725      continue
      elseif (contype .eq. 1) then
        do 735 i = 1,ndof
          do 740 j=1,nlen
            go(i) = go(i) + (l2(i,j)+mass(jdiag(i))*l1(i,j)/delta)
                      *gamma(j)
            gk(i) = gk(i) + l2(i,j)*gamma(j)
740      continue
735      continue
      endif

      return
      end

```

File: recorder.f

```

C=Module SEORDER
C=Author K. Alvin
C=Date May 1990
C=Block Fortran
C -----
C
C      Subroutine SEORDER
C
C      Purpose:
C
C          Solves the second-order dynamical equation:
C
C               $Mx'' + Dx' + Kx = f + g$ 
C
C          at time (n+1) given f(n+1/2), g(n+1/2) and x,x' at n by
C          the midpoint implicit integration rule.
C          Step size is 2*delta.
C
C          D is of the form (alpha*M + beta*K), f is an applied force,
C          and g is assumed to be other applied force from a feedback
C          control loop. The matrix E is the factored form of the
C          integration coefficient matrix: E=(M + delta*D + delta^2*K).
C -----
C
C      Arguments.
C
C          m          - matrix M
C          k          - matrix K
C          alpha      - scalar alpha
C          beta       - scalar beta
C          f          - Force vector f(n+1/2)
C          g          - Feedback force vector g(n+1/2)
C          e          - matrix E
C          x          - Variable vector x(n)
C          xd         - Variable vector x'(n)
C          delta      - Half of integration time step
C          delsq      - delta^2
C          jdiag      - Diagonal location pointer for M,K,E matrices
C          ndof       - Number of equations and length of q

```



```

C      v      - mass multiplier for RHS preparation
C      delbeta - delta * beta

C      subroutine SECORDER(m,k,alpha,beta,f,g,e,x,xd,
C      .      delta,delsq,jdiag,ndof,MAXDOF)

      recursive subroutine SECORDER(m,k,alpha,beta,f,g,e,x,xd,
      .      delta,delsq,jdiag,ndof,MAXDOF)

C      ARGUMENTS

      real*8 m(1),k(1),alpha,beta,f(1),g(1),e(1)
      real*8 x(1),xd(1),delta,delsq
      integer jdiag(1),ndof,MAXDOF

C      LOCAL VARIABLES

      integer i
      real*8 v(3000),delbeta

C      LOGIC
C      ADD APPLIED FORCES TO RHS AND PREPARE MASS MULTIPLIER

      do 10 i=1,ndof
        g(i) = g(i) + f(i)
        v(i) = (1. + delta*alpha)*x(i) + delta*xd(i)
10      continue

C      SOLVE FOR RIGHT HAND SIDE, g

      do 77 i=1,ndof
        g(i) = delsq*g(i) + v(i)*m(jdiag(i))
77      continue

      if (beta .ne. 0.) then
        delbeta = delta*beta

C      Activate EBE computations for internal force by using STIFFRC
C      subroutine. Otherwise use PMVMAD (profile matrix/vector mult-add)

C      call PMVMAD(k,jdiag,x,ndof,delbeta,g,1.d0)
C      call STIFFRC(x,delbeta,g)
C      endif

C      SOLVE FOR DISPLACEMENT, q, USING RHS AND MATRIX E

      call SOLVER(e,g,jdiag,ndof,2)

      do 100 i=1,ndof
        xd(i) = 2.*(g(i) - x(i))/delta - xd(i)
        x(i) = 2.*g(i) - x(i)
100      continue

      return
      end

```

File: measure.f

C=Module MEASURE

C=Author K. Alvin

C=Date May 1990

C=Block Fortran

```
C -----
C
C   Subroutine MEASURE
C
C   Purpose:
C       This subroutine stores new measured sensor data by using the
C       previous displacement and velocity vectors at the sensor
C       locations
C
C -----
C
C   Arguments
C       zp - measured sensor data array
C
```

```
      subroutine MEASURE(zp)

      include 'shared.inc'
      real*8 zp(i)

      call ZEROVECT(zp,nSen)

      do 100 jj = 1,hdval
         i = hdrow(jj)
         j = hdcoll(jj)
         zp(i) = zp(i) + hd(jj)*q(j)
100    continue
      do 200 jj = 1,hvval
         i = hvrow(jj)
         j = hvcol(jj)
         zp(i) = zp(i) + hv(jj)*qdot(j)
200    continue

      return
      end
```

File: eigens.f

C=Module EIGENS

C=Purpose Find Eigenmodes given Mass, Stiffness Matrices

C=Author K. Alvin

C=Date March 1990

C=Block Fortran

```
C -----
C
C   subroutine EIGENS
C
C -----
C
C   COMMON AND GLOBALS
```

```
include 'shared.inc'
```

C LOCAL VARIABLES

```
parameter(NVM=100,MNVM=MAXDOF*NVM,MXCNV=NVM*(NVM+1)/2)
integer isl(MAXDOF),nvec,i,j,k,kk,out
real*8 vl(MNVM),vr(MNVM),akk(MXCNV),amm(MXCNV)
real*8 xx(NVM,NVM),eigv(NVM),eigold(NVM)
real*8 toleig,toljac,omega,fhz
```

```
toleig=.0001
out = 13
```

```
toljac = toleig
nvec = min0(2*neig,100)
nvec = min0(nvec,ndof)
```

C SET UP ISL VECTOR

```
isl(1) = 1
do 50 j=2,ndof
    isl(j) = j - jdiag(j) + jdiag(j-1) + 1
50    continue
```

C CALL EIGENSOLVER

```
call SSPACE(stif,mass,vl,vr,akk,amm,xx,eigv,eigold,isl,
    jdiag,neig,nvec,ndof,toleig,toljac,out)
```

C WRITE OUTPUT

```
write(out,*) 'EIGEN ANALYSIS RESULTS:'
write(out,*) '
write(out,*) ' MODE      EIGENVALUE      RADIAL      CYCLIC'
write(out,*) ' ----      -
write(out,*) '
write(out,*)
do 100 i=1,neig
    omega = dsqrt(eigv(i))
    fhz = omega/(2*3.141592654)
    write(out,'(i5,3(3x,g12.5))') i,eigv(i),omega,fhz
100    continue

write(out,*) 'EIGENVECTORS:'
do 200 j=1,neig,5
    write(out,*)
    do 300 i=1,ndof
        k = ndof*(j-1) + i
        write(out,'(i5,5(1x,g12.5))') i,(vr(kk),kk=k,k+4*ndof,ndof)
300        continue
200    continue

write(out,*) 'MASS MATRIX DIAGONAL:'
do 400 i=1,nnp
    do 450 j=1,6
        if (id(j,i).ne.0) then
            write(out,*) i,j,id(j,i),mass(jdiag(id(j,i)))
        endif
450    continue
```

400 continue

return

end

C=End Fortran

File: singeig.f

```

      SUBROUTINE SSPACE (AK,AM,VL,VR,AKK,AMM,XX,EIGV,EIGOLD,ISL,
1  IDIAG,NEIG,NVEC,NDOF,TOLEIG,TOLJAC,NW)
C-----
C
C   input :
C
C       AK      : stiffness matrix      (profile values) (NDOF)
C       AM      : consistent mass matrix (profile values) (NDOF)
C       ISL     : stores in position "i" the row # of tip of
C                column "i" (NDOF)
C       IDIAG   : position of diagonal terms in profile      (NDOF)
C       NEIG    : # of required eigenvalues
C       NVEC    : # of subspace vectors
C       NDOF    : # of degrees of freedom
C       TOLEIG  : tolerance for eigenvalues convergence
C       TOLJAC  : tolerance for Jacobi convergence
C       NW      : logical unit number for output
C
C   output:
C
C       VL(NDOF,NVEC)      : working array
C       - VL(.,1..NRMOD)   : AM times rigid modes
C       - VL(.,NRMOD..NVEC) : subspace at the previous step
C
C       VR(NDOF,NVEC)      : eigen-vectors
C       - VR(.,1..NRMOD)   : rigid modes
C       - VR(.,NRMOD..NVEC) : subspace at this step (eigenvectors)
C
C       AKK(NVEC*(NVEC + 1)/2) : stiffness matrix in the subspace
C       AMM(NVEC*(NVEC + 1)/2) : consistent mass matrix in the subspace
C       XX(NVEC*NVEC)         : subspace eigenvectors
C
C       EIGV(NVEC)           : current eigenvalues
C       - EIGV(1..NRMOD)     : 0 eigen-values
C       - EIGV(NRMOD..NVEC)  : following eigenvalues > 0
C
C       EIGOLD(NVEC)         : same as EIGV
C-----
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AK(1),AM(1),VL(NDOF,NVEC),VR(NDOF,NVEC),AKK(1),
1  AMM(1),XX(NVEC*NVEC),EIGV(1),EIGOLD(1),ISL(1),IDIAG(1)
C
      WRITE (NW,1003) NEIG,NVEC,NDOF,TOLEIG
C
      CALL INVECT (AK,AM,VL,VR,IDIAG,NDOF,NVEC)
      CALL FACT (AK,IDIAG,ISL,NDOF,NW)
      CALL NULL (AK,AM,VL,VR,IDIAG,ISL,NDOF,NRMOD)

```

```

C
C      NRMOD : # of rigid modes
C
WRITE (NW,1004) NRMOD
NSUB=NVEC-NRMOD
C
CALL ORTHO (VR,VL,NDOF,NRMOD,NVEC)
C
NIT=0
NSMAX=15
NITMAX=16
NVEC1=NVEC-1
DO 5 I=1,NVEC
5  EIGOLD(I)=0.0
C
500 NIT=NIT+1
WRITE (NW,1000) NIT
C
CALL SOLVES (AK,VL(1,NRMOD+1),VR(1,NRMOD+1),IDIAG,ISL,NDOF,NSUB)
CALL ORTHO (VL,VR,NDOF,NRMOD,NVEC)
C
IJ=0
DO 10 J=NRMOD+1,NVEC
C
C      CALCULATE THE UPPER PART OF AKK (SYMMETRIC)
C
DO 10 I=NRMOD+1,J
TR=0.0
DO 11 K=1,NDOF
11  TR=TR+VL(K,I)*VR(K,J)
IJ=IJ+1
AKK(IJ)=TR
10 CONTINUE
C
CALL MULT (AM,VR(1,NRMOD+1),VL(1,NRMOD+1),ISL,IDIAG,NSUB,NDOF)
C
IJ=0
DO 20 J=NRMOD+1,NVEC
C
C      CALCULATE THE UPPER PART OF AMM (SYMMETRIC)
C
DO 20 I=NRMOD+1,J
TR=0.0
DO 21 K=1,NDOF
21  TR=TR+VL(K,I)*VR(K,J)
IJ=IJ+1
AMM(IJ)=TR
20 CONTINUE
C
CALL JACOBI (AKK,AMM,XX,EIGV(NRMOD+1),NSMAX,TOLJAC,NSUB,NW)
C
C      ORDER EIGENVALUES & EIGENVECTORS
C
30 IS=0
DO 40 I=NRMOD+1,NVEC1
IF (EIGV(I+1).GE.EIGV(I)) GO TO 40
IS=1
TR=EIGV(I+1)
EIGV(I+1)=EIGV(I)

```

```

      EIGV(I)=TR
      DO 41 J=1,NSUB
        TR=XX(J+(I-NRMOD)*NSUB)
        XX(J+(I-NRMOD)*NSUB)=XX(J+(I-NRMOD-1)*NSUB)
41      XX(J+(I-NRMOD-1)*NSUB)=TR
40      CONTINUE
      IF (IS.EQ.1) GO TO 30
C
C      SUBSPACE CONVERGENCE TEST
C
      ICONV=0
      DO 50 I=NRMOD+1,NVEC
        TR=DABS((EIGOLD(I)-EIGV(I))/EIGV(I))
        EIGOLD(I)=EIGV(I)
        EIGV(I)=TR
        IF (TR.GT.TOLEIG.AND.I.LE.NEIG) ICONV=1
50      CONTINUE
      WRITE (NW,1001) (EIGV(I),I=1,NVEC)
      IF (ICONV.EQ.0) GO TO 100
C
      IF (MIT.LE.NITMAX) GO TO 70
      WRITE (NW,1002)
      GO TO 100
C
C      UPDATE EIGEN VECTORS
C
70      DO 80 I=1,NDOF
        DO 80 J=1,NSUB
          TR=0.0
          DO 81 K=1,NSUB
81            TR=TR+VR(I,K+NRMOD)*XX(K+(J-1)*NSUB)
            VL(I,J+NRMOD)=TR
80          CONTINUE
        DO 90 I=1,NDOF
          DO 90 J=NRMOD+1,NVEC
90            VR(I,J)=VL(I,J)
          GO TO 500
C
C      CALCULATE FINAL EIGENVECTORS
C
100      DO 110 I=1,NDOF
        DO 110 J=1,NSUB
          TR=0.0
          DO 111 K=1,NSUB
111            TR=TR+VL(I,K+NRMOD)*XX(K+(J-1)*NSUB)
            VR(I,J+NRMOD)=TR
110          CONTINUE
        DO 112 I=1,NVEC
112          EIGV(I)=EIGOLD(I)
C
      RETURN
C
1000  FORMAT (5X,12HITERATION NO,I5)
1001  FORMAT (6(2X,1PE10.3))
1002  FORMAT (5X,24HWE ACCEPT CURRENT VALUES)
1003  FORMAT (//20X,'SUBSPACE ITERATION ROUTINE'// ' NB OF EIGENVALUES=',
1004  1 I5/' NB OF VECTOR=',I5/' NB OF DOF=',I5/' TOLERANCE=',1PE10.3/)
1004  FORMAT (' NB OF RIGID MODES=',I5//)
C

```

```

      END
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE INVECT (AK,AM,VL,VR,IDIAG,NDOF,NVEC)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AK(1),AM(1),VL(1),VR(NDOF,NVEC),IDIAG(1)
      C
      ND=NDOF/NVEC
      C
      DO 10 I=1,NDOF
        II=IDIAG(I)
        VR(I,1)=AM(II)
        VL(I)=AM(II)/AK(II)
        DO 10 J=2,NVEC
          VR(I,J)=0.0
      10 CONTINUE
      C
      LL=NDOF-ND
      C
      DO 20 J=2,NVEC
        TR=0.0
      C
        DO 30 I=1,LL
          IF (VL(I).LT.TR) GO TO 30
          TR=VL(I)
          IJ=I
      30 CONTINUE
      C
        DO 40 I=LL,NDOF
          IF (VL(I).LE.TR) GO TO 40
          TR=VL(I)
          IJ=I
      40 CONTINUE
      C
        VL(IJ)=0.0
        LL=LL-ND
        VR(IJ,J)=1.0
      C
      20 CONTINUE
      C
      RETURN
      END
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE MULT (AM,VR,VL,ISL,IDIAG,NVEC,NDOF)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AM(1),VR(NDOF,NVEC),VL(NDOF,NVEC),ISL(1),IDIAG(1)
      C
      DO 500 IV=1,NVEC
        DO 100 I=1,NDOF
          TR=0.0
          IJ=IDIAG(I)
          IK=ABS(ISL(I))
          KK=I
          DO 110 K=IK,I
            TR=TR+AM(IJ)*VL(KK,IV)
            IJ=IJ-1
            KK=KK-1
      110 CONTINUE
      C
        IF (I.EQ.NDOF) GO TO 99

```



```

D1=AB+SQCH
D2=AD-SQCH
DEN=D1
IF (DABS(D2).GT.DABS(D1)) DEN=D2
IF (DEN.NE.ZERO) GO TO 45
CA=ZERO
CG=-AK(JK)/AK(KK)
GO TO 50
45 CA=AKK/DEN
CG=-AJJ/DEN
C
C
C
PERFORM GENERALIZED ROTATION
50 JP1=J+1
JN1=J-1
KP1=K+1
KN1=K-1
IF (JN1.LT.1) GO TO 70
DO 60 I=1,JN1
IJ=J+JN1/2+I
IK=K+KN1/2+I
AKJ=AK(IJ)
AKK=AK(IK)
AMJ=AM(IJ)
AMK=AM(IK)
AK(IJ)=AKJ+CG+AKK
AM(IJ)=AMJ+CG+AMK
AK(IK)=AKK+CA+AKJ
AM(IK)=AMK+CA+AMJ
60 CONTINUE
70 IF (KP1.GT.N) GO TO 90
DO 80 I=KP1,N
JI=I+(I-1)/2+J
KI=I+(I-1)/2+K
AKJ=AK(JI)
AMJ=AM(JI)
AKK=AK(KI)
AMK=AM(KI)
AK(JI)=AKJ+CG+AKK
AM(JI)=AMJ+CG+AMK
AK(KI)=AKK+CA+AKJ
AM(KI)=AMK+CA+AMJ
80 CONTINUE
90 IF (JP1.GT.KN1) GO TO 110
DO 100 I=JP1,KN1
JI=I+(I-1)/2+J
IK=K+(K-1)/2+I
AKJ=AK(JI)
AMJ=AM(JI)
AKK=AK(IK)
AMK=AM(IK)
AK(JI)=AKJ+CG+AKK
AM(JI)=AMJ+CG+AMK
AK(IK)=AKK+CA+AKJ
AM(IK)=AMK+CA+AMJ
100 CONTINUE
110 AKK=AK(KK)
AMK=AM(KK)
AKJ=AK(JJ)

```

```

      AMJ=AM(JJ)
      AK(KK)=AKK+TWO*CA*AK(JK)+CA*CA*AKJ
      AM(KK)=AMK+TWO*CA*AM(JK)+CA*CA*AMJ
      AK(JJ)=AKJ+TWO*CG*AK(JK)+CG*CG*AKK
      AM(JJ)=AMJ+TWO*CG*AM(JK)+CG*CG*AMK
      AK(JK)=ZERO
      AM(JK)=ZERO

C
C      UPDATE EIGENVECTOR FOR THIS ROTATION
C
      DO 120 I=1,N
        XXJ=XX(I,J)
        XXX=XX(I,K)
        XX(I,J)=XXJ+CG*XXX
        XX(I,K)=XXX+CA*XXJ
120    CONTINUE
150  CONTINUE

C
C      UPDATE EIGENVALUES & CHECK CONVERGENCE
C
      NT=N*(N+1)/2
      ICONV=0
      DO 160 I=1,N
        II=I*(I-1)/2+I
        IF (AK(II).LE.ZERO.OR.AM(II).LE.ZERO) GO TO 900
        TR=AK(II)/AM(II)
        DEN=(TR-EIGV(I))/TR
        EIGV(I)=TR
        IF (DABS(DEN).GT.TOL) ICONV=1
160  CONTINUE
      IF (ICONV.EQ.1) GO TO 499

C
C      CHECK OFF DIAGONAL TERMS
C
      EPS=TOL*TOL
      DO 170 J=1,NR
        IIK=J+1
        DO 170 K=IIK,N
          JJ=J*(J-1)/2+J
          KK=K*(K-1)/2+K
          JK=K*(K-1)/2+J
          EPSAK=(AK(JK)*AK(JK))/(AK(JJ)*AK(KK))
          EPSAM=(AM(JK)*AM(JK))/(AM(JJ)*AM(KK))
          IF (EPSAK.LT.EPS.AND.EPSAM.LT.EPS) GO TO 170
        GO TO 499
170  CONTINUE

C
C      SCALE EIGENVECTORS
C
179  DO 180 I=1,N
        II=I*(I-1)/2+I
        AKK=DSQRT(AM(II))
        DO 180 J=1,N
          XX(J,I)=XX(J,I)/AKK
180  CONTINUE

C
      RETURN

C
499  IF (NSWEEP.LE.NSMAX) GO TO 500

```

```

      WRITE (NW,1000)
      GO TO 179
C
900  WRITE (NW,1001)
      STOP
C
1000 FORMAT (5X,34HNO CONVERGENCE AT NSMAX ITERATIONS)
1001 FORMAT (5X,46HERROR IN JACOBI : MATRIX NOT POSITIVE DEFINITE)
C
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SOLVES (AK,VL,VR,IDIAG,ISL,NDOF,NVEC)
C
C      PURPOSE : SOLVES THE SINGULAR PROBLEM :  $AK \times VL = VR$ 
C                  THE SINGULAR COLUMNS INTO AK ARE INDEXED BY
C                  THE NEGATIVE VALUES OF ISL, THE CORRESPONDING
C                  TERMS OF THE SOLUTION ARE PUT TO 0
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AK(1),VL(NDOF,NVEC),VR(NDOF,NVEC),IDIAG(1),ISL(1)
C
      DO 500 IV=1,NVEC
        DO 50 I=1,NDOF
          VL(I,IV)=VR(I,IV)
50
C
C      BACKSUBSTITUTE
C
      DO 100 IC=2,NDOF
        TR=0.0
        IC1=IC-1
        IM1=IDIAG(IC)-IC
        K=ISL(IC)
        IF (IK.LE.0) THEN
          VL(IC,IV)=0.
          GOTO 100
        ENDIF
        IF (IK.GT.IC1) GO TO 100
        DO 120 K=IK,IC1
          TR=TR+AK(IM1+K)*VL(K,IV)
120      CONTINUE
          VL(IC,IV)=VL(IC,IV)-TR
100      CONTINUE
C
C      SOLVE DU=U
C
      DO 150 IC=1,NDOF
        VL(IC,IV)=VL(IC,IV)/AK(IDIAG(IC))
150      CONTINUE
C
C      BACKSUBSTITUTE
C
      IIC=NDOF
      DO 200 IC=2,NDOF
        TR=VL(IIC,IV)
        IC1=IIC-1
        IK=ISL(IIC)
        IM1=IDIAG(IIC)-IIC

```



```

        IN=IN2+IL
        AK(IN)=AK(IN)-TR
120    CONTINUE
C
C    CALCULATE L&D
C
150    TR=0.0
        IF (NIC.GT.IC1) GO TO 201
        DO 200 IL=NIC,IC1
            IF (IDIAG(IL).LT.0) GOTO 200
            AG=AK(IN2+IL)
            AL=AG/AK(IDIAG(IL))
            AK(IN2+IL)=AL
            TR=TR+AL*AG
200    CONTINUE
201    IN=IDIAG(IC)
        AK(IN)=AK(IN)-TR
        IF (AK(IN).LT.ZERO) IDIAG(IC)=-IDIAG(IC)
100    CONTINUE
        RETURN

901    WRITE (NW,1001)
        STOP

1001    FORMAT (5X,29H***STOP ERROR IN IDIAG VECTOR)
1010    FORMAT (5X,'CONDITIONING OF THE STIFFNESS MATRIX'/2X,
1      'MIN DIAG TERM=',1PE10.3,' MAX DIAG TERM=',E10.3)

END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
SUBROUTINE NULL (AK,AM,VL,VR,IDIAG,ISL,NDOF,NRMOD)
C
C    PURPOSE : CALCULATE THE NULL SPACE OF AK AND PUT AN
C              ORTHONORMALISED BASE OF THIS SPACE INTO THE
C              NRMOD FIRST VECTORS OF VR.
C              VL = AM x VR AFTER EXECUTION
C
C
C    IMPLICIT REAL*8 (A-H,O-Z)
C    DIMENSION AK(1),AM(1),IDIAG(1),ISL(1),VL(NDOF,*),VR(NDOF,*)
C
C    STORE THE SINGULAR COLUMNS INTO THE BEGINNING OF VR
C
C    THE SINGULAR EQUATION ARE NOW INDEXED BY NEGATIVE VALUES
C    INTO ISL INSTEAD OF IDIAG
C
C    NRMOD=0
C    DO 1 IC=1,NDOF
        IF (IDIAG(IC).GT.0) GOTO 1
        IDIAG(IC)=-IDIAG(IC)
        NRMOD=NRMOD+1
        NIC=ISL(IC)
        IN=IDIAG(IC)-IC
        DO 2 K=1,NIC-1
            VR(K,NRMOD)=0.
        DO 3 K=NIC,IC-1
            VR(K,NRMOD)=AK(IN+K)
            AK(IN+K)=0.
2      CONTINUE
3      CONTINUE

```

```

        ISL(IC)=-ISL(IC)
        VR(IC,NRMOD)=-1.
        AK(IDIAG(IC))=1.
        DO 4 K=IC+1,NDOF
4          VR(K,NRMOD)=0.
1    CONTINUE
C
C    BAKSUBSTITUTE
C
        DO 200 N=1,NRMOD
            IIC=NDOF
            DO 200 IC=2,NDOF
                TR=VR(IIC,N)
                IC1=IIC-1
                IK=ISL(IIC)
                IN1=IDIAG(IIC)-IIC
                IF ((IK.GT.IC1).OR.(IK.LE.0)) GO TO 221
                DO 220 K=IK,IC1
                    VR(K,N)=VR(K,N)-AK(IN1+K)*TR
220                CONTINUE
221                IIC=IIC-1
200    CONTINUE
C
C    ORTHOGONALISATION
C
        DO 10 N=1,NRMOD
            DO 20 K=1,N-1
                TR=0.
                DO 30 I=1,NDOF
30                  TR=TR+VL(I,K)*VR(I,N)
                DO 40 I=1,NDOF
40                  VR(I,N)=VR(I,N)-TR*VR(I,K)
20    CONTINUE
                CALL MULT(AM,VL(1,N),VR(1,N),ISL,IDIAG,1,NDOF)
                TR=0.
                DO 50 I=1,NDOF
50                  TR=TR+VR(I,N)*VL(I,N)
                TR=1/SQRT(TR)
                DO 60 I=1,NDOF
                    VR(I,N)=VR(I,N)*TR
                    VL(I,N)=VL(I,N)*TR
60    CONTINUE
10    CONTINUE

        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE ORTHO(VL,VR,NDOF,NRMOD,NVEC)
C
C    PURPOSE : ORTHOGONALISE THE NSUB LAST COLUMNS OF VL (LAST
C               EVALUATED SOLUTION) WITH RESPECT TO THE NULL SPACE
C               OF A, FOR THE AM SCALAR PRODUCT
C
C
        IMPLICIT REAL*8 (A-H,O-Z)
        INTEGER NDOF,NRMOD,NVEC
        DIMENSION VL(NDOF,NVEC),VR(NDOF,NVEC)

        NSUB=NVEC-NRMOD

```

```

      DO 1 J=1,MSUB
        DO 2 I=1,NRMOD
          S=0.
          DO 3 K=1,NDOF
            S=S+VL(K,I)*VL(K,NRMOD+J)
          DO 4 L=1,NDOF
            VL(L,NRMOD+J)=VL(L,NRMOD+J)-S*VR(L,I)
          2      CONTINUE
        1      CONTINUE

```

```

      RETURN
      END

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

File: animout.f

```

C=Module ANIMOUT
C=Author K. Alvin
C=Date June 1990
C=Block Fortran

```

```

C -----
C
C      subroutine ANIMOUT
C
C      Purpose:
C      This subroutine produces an output file to be used to
C      visualize the simulation using MESH.
C -----
C
C

```

```

      subroutine ANIMOUT(q,id,nnp,time,out)
      real*8 q(1),time,v(3)
      integer out,id(6,1),nnp,i,k
      write(out,'(g15.5)') time
      do 100 i=1,nnp
        do 200 k=1,3
          if (id(k,i) .ne. 0) then
            v(k) = q(id(k,i))
          else
            v(k) = 0.
          endif
        200      continue
        write(out,1000) (v(k),k=1,3)
      100      continue
      1000 format(3g15.5)
      return
      end

```

File: stiffrc.f

```

C      subroutine STIFFRC(v,fact,kq)

```

```

recursive subroutine STIFFRC(v,fact,kq)

real*8 v(1),fact,kq(1)

include 'shared.inc'

C  ASSEMBLE EACH ELEMENT MASS AND STIFFNESS

do 100 ii=1,ndomain

CVD$  CMCALL
      do 110 jj=1,neld(ii)
        n = elnum(jj,ii)
        call ELEFRC(v,fact,kq,n)
110    continue

100    continue

      return
      end

C  subroutine ELEFRC(v,fact,kq,n)

recursive subroutine ELEFRC(v,fact,kq,n)

real*8 v(1),fact,kq(1)
integer n

include 'shared.inc'

C  LOCAL VARIABLES

parameter(MAXSEQ=24)
real*8 sk(MAXSEQ,MAXSEQ)
integer lm(MAXSEQ),nseq

do 20 k=1,4
  j=ix(k,n)
  if ((etype(n).eq.1).and.(k.gt.2)) j = 0
  do 30 i=1,6
    kk=6*(k-1) + i
    if (j .ne. 0) then
      lm(kk) = id(i,j)
    else
      lm(kk) = 0
    endif
30    continue
20    continue

  nseq=12

  call LOADSK(sk,n,nseq)

  call ESTIFVM(sk,lm,nseq,v,kq,fact)

  return
  end

```



```

C      subroutine LOADSK(sk,n,nseq)

      recursive subroutine LOADSK(sk,n,nseq)

      include 'shared.inc'

      real*8 sk(nseq,1)
      integer n,nseq

      k=0
      do 10 j=1,nseq
        do 20 i=1,j
          k=k+1
          sk(i,j)=estifm(k,n)
          sk(j,i)=sk(i,j)
20        continue
10      continue

      return
      end

```

File: estifvm.f

```

C      subroutine ESTIFVM(sk,lm,nseq,v,kq,fact)

      recursive subroutine ESTIFVM(sk,lm,nseq,v,kq,fact)

C      ARGUMENTS

      real*8 sk(nseq,1),v(1),kq(1),fact
      integer lm(1),nseq

      do 20 j = 1, nseq
        k = lm(j)
        if (k .eq. 0) goto 20
        do 10 i = 1, nseq
          m = lm(i)
          if (m .eq. 0) goto 10
          kq(m) = kq(m) + sk(i,j)*v(k)*fact
10        continue
20      continue

      return
      end
C=End Fortran

```

File: renum.f

```

C=DECK RENUM
C=PURPOSE- RENUMBERS THE GRID POINTS TO MINIMIZE PROFILE STORAGE
C=AUTHOR W K BELVIN and DUC NGUYEN 7-5-90
C
      subroutine RENUM
C
      include 'shared.inc'

```

```

C
C   Initialize vector
C
      maxtry=2*nnp/3 +1
C.....
      nterms=nnp*maxtry
      do 22 j=1,nterms
12      iadjcy(j)=0
      do 10 i=1,nnp
10      icount(i)=0
C*****
      do 1 i=1,nel
          nodea=ix(1,i)
          nodeb=ix(2,i)
          if ((nodea .eq. 0).or.(nodeb .eq. 0)) go to 1
          icount(nodea)=icount(nodea)+1
          icount(nodeb)=icount(nodeb)+1
          ia=icount(nodea)
          ib=icount(nodeb)
          if(ia.gt.maxtry .or. ib.gt.maxtry) go to 345
          locate=(nodea-1)*maxtry+ia
          iadjcy(locate)=nodeb
          locat=(nodeb-1)*maxtry+ib
          iadjcy(locate)=nodea
1      continue
C*****
      ii=0
      do 37 i=1,nterms
          if ( iadjcy(i) .eq. 0 ) go to 37
          ii=ii+1
          iadjcy(ii)=iadjcy(i)
37      continue
C*****
      last=0
      do 2 i=1,nnp
          last=last+icount(i)
2      continue
      jj=icount(1)
      icount(1)=1
      do 3 i=2,nnp+1
          kk=icount(i)
          icount(i)=icount(i-1)+jj
          jj=kk
3      continue
      go to 556
345  write(6,555)
555  format(2x,'error in dimension for MAXTRY !! ')
556  continue
C*****
      call GENRCN(nnp,icount,iadjcy,perm,mask,xls)
      return
      end
C*****
      subroutine genrcm(neqns,xadj,adjncy,perm,mask,xls)
C.....reference: computer solution of large sparse positive definite
C.....          systems, alan george & joseph w-h liu
C.....          (prentice-hall,inc.,englewood cliffs,NJ 07632)
      integer adjncy(1),mask(1),perm(1),xls(1)

```

```

integer xadj(1),ccsize,i,neqns,nlvl,num,root
do 100 i=1,neqns
  mask(i)=1
100  continue
  num=1
  do 200 i=1,neqns
    if( mask(i).eq.0 ) go to 200
    root=i
    call froot(root,xadj,adjncy,mask,nlvl,xls,perm(num) )
    call rcm(root,xadj,adjncy,mask,perm(num),ccsize,xls)
    num=num+ccsize
    if(num.gt.neqns) go to 987
  200  continue
c.....perm(new node)=old node
c.....now, mask(old node)= new node
987  continue
  do 11 new=1,neqns
    iold=perm(new)
    mask(iold)=new
c    write(6,*) 'iold,mask(iold) = ',iold,mask(iold)
  11  continue
  return
end
cXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
subroutine froot(root,xadj,adjncy,mask,nlvl,xls,ls)
integer adjncy(1),ls(1),mask(1),xls(1)
integer xadj(1),ccsize,j,jstrt,k,kstop,kstrt,
$   mindeg,nabor,ndeg,nlvl,node,nunlvl,root
call rootls(root,xadj,adjncy,mask,nlvl,xls,ls)
ccsize=xls(nlvl+1)-1
if(nlvl.eq.1 .or. nlvl.eq.ccsz) return
100  jstrt=xls(nlvl)
  mindeg=ccsize
  root=ls(jstrt)
  if(ccsize.eq.jstrt) go to 400
  do 300 j=jstrt,ccsize
    node=ls(j)
    ndeg=0
    kstrt=xadj(node)
    kstop=xadj(node+1)-1
    do 200 k=kstrt,kstop
      nabor=adjncy(k)
      if( mask(nabor) .gt. 0) ndeg=ndeg+1
    200  continue
    if(ndeg.ge.mindeg) go to 300
    root=node
    mindeg=ndeg
  300  continue
400  call rootls(root,xadj,adjncy,mask,nunlvl,xls,ls)
  if(nunlvl.le.nlvl) return
  nlvl=nunlvl
  if(nlvl.lt.ccsz) go to 100
  return
end
cXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
subroutine rcm(root,xadj,adjncy,mask,perm,ccsize,deg)
integer adjncy(1),deg(1),mask(1),perm(1)
integer xadj(1),ccsize,fnbr,i,j,jstop,jstrt,k,l,lbegin,
$   lnbr,lperm,lvlend,nbr,node,root

```

```

call degree(root,xadj,adjncy,mask,deg,ccsize,perm)
mask(root)=0
if(ccsize.le.1) return
lvlend=0
lnbr=1
100 lbegin=lvlend+1
    lvlend=lnbr
    do 600 i=lbegin,lvlend
        node=perm(i)
        jstrt=xadj(node)
        jstop=xadj(node+1)-1
        fnbr=lnbr+1
        do 200 j=jstrt,jstop
            nbr=adjncy(j)
            if(mask(nbr).eq.0) go to 200
            lnbr=lnbr+1
            mask(nbr)=0
            perm(lnbr)=nbr
200        continue
        if(fnbr.ge.lnbr) go to 600
        k=fnbr
300        l=k
            k=k+1
            nbr=perm(k)
400        if(l.lt.fnbr) go to 500
            lperm=perm(l)
            if( deg(lperm).le.deg(nbr) ) go to 500
            perm(l+1)=lperm
            l=l-1
            go to 400
500        perm(l+1)=nbr
            if(k.lt.lnbr) go to 300
600        continue
        if(lnbr.gt.lvlend) go to 100
        k=ccsize/2
        l=ccsize
        do 700 i=1,k
            lperm=perm(l)
            perm(l)=perm(i)
            perm(i)=lperm
            l=l-1
700        continue
    return
end
cXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
subroutine rootls(root,xadj,adjncy,mask,nlvl,xls,ls)
integer adjncy(1),ls(1),mask(1),xls(1)
integer xadj(1),i,j,jstop,jstrt,lbegin,ccsize,lvlend,
$      lvsize,nbr,nlvl,node,root
mask(root)=0
ls(1)=root
nlvl=0
lvlend=0
ccsize=1
200 lbegin=lvlend+1
    lvlend=ccsize
    nlvl=nlvl+1
    xls(nlvl)=lbegin
    do 400 i=lbegin,lvlend

```

```

        node=ls(i)
        jstrt=xadj(node)
        jstop=xadj(node+1)-1
        if(jstop.lt.jstrt) go to 400
        do 300 j=jstrt,jstop
            nbr=adjncy(j)
            if( mask(nbr).eq.0) go to 300
            ccsize=ccsize+1
            ls(ccsize)=nbr
            mask(nbr)=0
300      continue
400      continue
        lvsize=ccsize-lvlend
        if(lvsize.gt.0) go to 200
        xls(nlvl+1)=lvlend+1
        do 500 i=1,ccsize
            node=ls(i)
            mask(node)=1
500      continue
        return
    end
cXXXXXXXXXXXXXXXXXXXXX
    subroutine degree(root,xadj,adjncy,mask,deg,ccsize,ls)
    integer adjncy(1),deg(1),ls(1),mask(1)
    integer xadj(1),ccsize,i,ideg,j,jstop,jstrt,
    $      lbegin,lvlend,lvsize,nbr,node,root
    ls(1)=root
    xadj(root)=-xadj(root)
    lvlend=0
    ccsize=1
100    lbegin=lvlend+1
        lvlend=ccsize
        do 400 i=lbegin,lvlend
            node=ls(i)
            jstrt=-xadj(node)
            jstop=iabs( xadj(node+1) ) -1
            ideg=0
            if(jstop.lt.jstrt) go to 300
            do 200 j=jstrt,jstop
                nbr=adjncy(j)
                if( mask(nbr).eq.0 ) go to 200
                ideg=ideg+1
                if(xadj(nbr).lt.0) go to 200
                xadj(nbr)=-xadj(nbr)
                ccsize=ccsize+1
                ls(ccsize)=nbr
200          continue
300          deg(node)=ideg
400          continue
        lvsize=ccsize-lvlend
        if(lvsize.gt.0) go to 100
        do 500 i=1,ccsize
            node=ls(i)
            xadj(node)=-xadj(node)
500      continue
        return
    end

```

File: kfilter.f

```

C      subroutine KFILTER

      recursive subroutine KFILTER

      include 'shared.inc'

      do 10 i = 1,ndof
         go(i) = detsq*(f(i)+go(i))+delta*pe(i)+mass(jdiag(i))*qe(i)
         gk(i) = delta*(f(i)+gk(i))+pe(i)
10      continue

      call SOLVER(eo,go,jdiag,ndof,2)

C      Activate EBE computations for internal force by using STIFFRC
C      subroutine. Otherwise use PHVMAD (profile matrix/vector mult-add

C      call PHVMAD(stif,jdiag,go,ndof,-delta,gk,1.d0)
      call STIFFRC(go,-delta,gk)

      do 100 i = 1,ndof
         qe(i) = 2.*go(i) - qe(i)
         pe(i) = 2.*gk(i) - pe(i)
100      continue

      return
      end

```

**A Modular Multibody Analysis Capability
for
High-Precision, Active Control and Real-Time Applications**

K. C. Park, J.D. Downer, J.C. Chiou and C. Farhat

Department of Aerospace Engineering Sciences and
Center for Space Structures and Controls
University of Colorado, Campus Box 429
Boulder, Colorado 80309

January 1991

Abstract

A computationally oriented formulation and its solution procedures for the analysis of rigid-flexible multibody dynamics systems are presented. The present formulation adopts a set of dual coordinate systems, inertially fixed one for the measure of translational motions and body-based one for rotational motions. The origins of these coordinate systems are located at the centers of rigid bodies and at the nodal points of flexible bodies for implementation ease. The flexibility is modeled by an intrinsic spatial beam theory which is approximated by transverse-shear deformable linear beam elements. The solution of the resulting equations of motion including system constraints is obtained by a partitioned procedure, which solves first for the constraint force vector, then the generalized coordinates for the rigid and flexible components, and finally interaction quantities such as active control forces, maneuvering space ranges, and corrections due to state measurements, with each solution stage being processed by the corresponding separate module. A central feature of the present capability is its high-accuracy of the analysis results by demanding the energy conservation throughout the various stages of response analyses. Several examples are included to demonstrate the capabilities of the present analysis software both on sequential and parallel machines.

1. Introduction

The multibody systems analysis capability that we present herein has been motivated by three emerging needs: real-time simulation, interfaciality with other disciplines such as active control and work-space path planning, and high accuracy in analysis results. The need for real-time or on-line analysis capability stems from space structural operations, deep-sea systems and remotely controlled moving of hazardous materials. The decision to interface multibody dynamics analysis systems with active control strategies rather than to integrate these two computational elements into one has been motivated to preserve software modularity and provide a relatively pain-minimizing environment for future expansions and modifications of the present capabilities. Finally, the requirement for highly accurate analysis results is for the present analysis package to be used in space construction as well as high-precision assembly simulation of mechanical and electronic machinery. Of course, the present package is also applicable to automobile and other transportation systems dynamics simulations as well wherein the accuracy requirement is not as stringent as in aerospace and electronic assembly simulations.

The present multibody simulation package consists of five modules: an input/output processor, the rigid-body elements that include joints and constraints, the spatial beam modeler, the partitioned solver, and the interface provisions. Of the five elements, in this paper we will restrict ourselves to describe the three kernels, viz., the rigid-body elements, the beam modeler, and the solver. To this end the paper is organized as follows. Section 2 presents the kinematic description of rigid-body elements including joint constraints. For physical as well as implementational considerations, the origin of the coordinate systems for each body is located at the center of mass. This is in contrast to many existing kinematic descriptions of kinematics that choose the joint hinges as the origins of the coordinate systems. Specifically, the translational motions are referred to an inertial frame whereas the rotational orientations are based on a body-fixed frame for each body. This dual-coordinate choice yields a constant inertia matrix, which makes the explicit time-marching solution efficient.

The modeling of flexible beams is detailed in Section 3. The present beam model employs a fully nonlinear theory which accounts for both finite rotations and moderately large deformations. Within the context of classical rod theory, the present formulation employs a convected coordinate representation of physical Cauchy stresses and corresponding set of physical strains. As a consequence, the beam model can easily be interfaced with on-line strain measurements and feedback control systems as sensors measure only physical quantities. Another advantage of the formulation is that the degrees of freedom of the beam component embody both the rigid and flexible deformation motions. A distinct feature of the present work is the computational preservation of total energy for undamped systems; this is obtained via an objective strain increment/stress update procedure combined with an energy-conserving time integration algorithm which contains an accurate update of angular orientations.

The solution module for the present multibody analysis capability consists of three sub-constructs: the solver for the translational degrees of freedom, the integrator for the angular orientations, and the stabilized solution of Lagrange multipliers for the system constraints. An active control system can be interfaced with the present capability by treating the active control forces as externally applied disturbances and by providing the necessary state vectors that the active control systems synthesizer must have. A unique feature of the present simulation procedure is that the Lagrange multipliers are obtained from a set of stabilized time-differential equations.

2. Equations of Motion for Constrained Multibody Systems

D'Alembert's equations of motion cast in the form of the principle of virtual work for a constrained multibody system can be stated as [1]

$$\int_V \delta u_i \rho \ddot{u}_i dV + \int_V \sigma_{ij}^e \frac{\partial \delta u_i}{\partial x_j} dV - \int_V \delta u_i f_i dV - \int_S \delta u_i t_i dS - \sum_{k=1}^m \delta \Phi_k \lambda_k = 0$$

$$\Phi_k = 0, \quad k = 1 \dots m. \quad (2.1)$$

In the above equation, the cartesian coordinates x_i represent the particle position after some deformation has taken place, δu_i a kinematically admissible virtual displacement, $\ddot{u}_i$ the absolute acceleration, f_i the external force per unit mass, and t_i the stress vector acting on a surface with outward normal components n_i . Likewise, σ_{ij}^e represents the cartesian components of the Cauchy stress tensor, and ρ is the mass density. Finally, λ_k is the Lagrange multiplier that enforces the k -th system constraint equation, $\Phi_k = 0$, and m is the total number of the constraint conditions. We will now present term by term formulation or discretization procedures to obtain the discrete equations of motion.

2.1 Inertia Operators for Rigid Bodies

The inertia force operator is the first term in (2.1)

$$\delta F^I = \int_V \rho \delta u_i \ddot{u}_i dV = \int_V \rho \delta \mathbf{u} \cdot \ddot{\mathbf{u}} dV \quad (2.2)$$

For the case of rigid bodies, the position vector $\mathbf{r}$ can be expressed as (see Fig. 1 for the description of the present kinematics adopted):

$$\mathbf{u}_p = \mathbf{u}^T \mathbf{e} + \mathbf{s}^T \mathbf{b}, \quad \mathbf{b} = \mathbf{R} \mathbf{e} \quad (2.3)$$

where "boldface" symbols represent three subscripted vectors and the normal type symbols represent three components of a given vector; $\mathbf{e} = \{ \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \}^T$ represents the three orthogonal vectors defining the inertial reference frame; $\mathbf{b} = \{ \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 \}^T$ represents the body-fixed reference frame which is attached to and rotates with the rigid body and

which is related to $\mathbf{e}$ by the (3×3) matrix $\mathbf{R}$; $\mathbf{u} = \{u_1, u_2, u_3\}^T$ represents the inertial components of the original neutral axis position; and $\mathbf{s} = \{s_1, s_2, s_3\}^T$ represents the distance of a particle ρdV from the center of mass measured in the body-fixed coordinate system.

The velocity and acceleration and virtual displacement vectors of point p can thus be derived from (2.3) as

$$\dot{\mathbf{u}}_p = \dot{\mathbf{u}}^T \mathbf{e} + \mathbf{s}^T \dot{\tilde{\omega}}^T \mathbf{b} \quad (2.4)$$

$$\ddot{\mathbf{u}}_p = \ddot{\mathbf{u}}^T \mathbf{e} + \mathbf{s}^T \ddot{\tilde{\omega}}^T \mathbf{b} + \mathbf{s}^T \dot{\tilde{\omega}}^T \dot{\tilde{\omega}}^T \mathbf{b} \quad (2.5)$$

$$\delta \mathbf{u}_p = \delta \mathbf{u}^T \mathbf{e} + \mathbf{s}^T \delta \tilde{\alpha}^T \mathbf{b} \quad (2.6)$$

where the skew-symmetric angular velocity tensor $\tilde{\omega}$ and the virtual rotational tensor $\delta \tilde{\alpha}$ are defined by

$$\tilde{\omega} = \mathbf{R} \dot{\mathbf{R}}^T = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \quad (2.7)$$

$$\delta \tilde{\alpha} = -\delta \mathbf{R} \mathbf{R}^T, \quad (2.8)$$

respectively. By substituting (2.5) and (2.6) into the inertia force operator (2.2), one obtains

$$\delta F^I = \delta \mathbf{u}^T \mathbf{M} (\ddot{\mathbf{u}} + \mathbf{R}^T \tilde{\mathbf{s}}_c^T \ddot{\tilde{\omega}} + \mathbf{R}^T \dot{\tilde{\omega}} \tilde{\mathbf{s}}_c^T \dot{\tilde{\omega}}) + \delta \alpha^T \mathbf{M} \tilde{\mathbf{s}}_c^T \mathbf{R} \ddot{\mathbf{u}} + \mathbf{J} \dot{\tilde{\omega}} + \tilde{\omega} \mathbf{J} \dot{\tilde{\omega}} \quad (2.9)$$

where

$$\mathbf{M} = \int_V \rho dV, \quad \mathbf{J} = \int_V \rho \tilde{\mathbf{s}} \tilde{\mathbf{s}}^T dV, \quad \mathbf{M} \tilde{\mathbf{s}}_c = \int_V \rho \tilde{\mathbf{s}} dV \quad (2.10)$$

A considerable simplification can be made in the inertia force operator if the origin of the body-fixed reference frame coincides with the center of mass of the body and the axes coincide with the principal axis of inertia of the body. With this choice, all the coupling terms between the translational and rotational degrees of freedom vanish as $\tilde{\mathbf{s}}_c = \tilde{\mathbf{0}}$. In addition, the mass and inertia matrices of the body, $\mathbf{M}$ and $\mathbf{J}$, are given as

$$\mathbf{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix}, \quad \mathbf{J} = \begin{bmatrix} J_{11} & 0 & 0 \\ 0 & J_{22} & 0 \\ 0 & 0 & J_{33} \end{bmatrix} \quad (2.11)$$

It is noted that both the translational and rotational degrees of freedom for rigid bodies are attached to the mass center. This choice leads to a different kinematical description of joint constraints as compared to the case of attaching the degrees of freedom at the joints.

2.2 Inertia Force Operator for Spatial Beam

As we intend to model the flexible bodies by the finite element method, a natural choice of attaching the degrees of freedom is at the elemental nodes rather than at the element mass center. Specifically, when we restrict ourselves to spatial beams, the position vector u_p at an arbitrary point on the beam can be expressed as (see Fig. 2)

$$u_p = (X + u)^T e + \ell^T b \quad (2.12)$$

where $X = \{X_1, X_2, X_3\}^T$ represents the inertial components of the original neutral axis position; $u = \{u_1, u_2, u_3\}^T$ represents the inertial components of the subsequent total translational displacement of the neutral axis, and $\ell^T = \{0, \ell_2, \ell_3\}$ are the body-fixed components of the distance from the beam neutral-axis to the material point located on the deformed beam cross-section. It is noted that the beam cross-section is allowed to rotate such that it is not necessarily perpendicular to the neutral axis in order to model transverse shear deformations. Warping deformation of the cross-section is not considered in the present derivation.

The velocity, acceleration, and virtual displacement vectors are obtained from (2.12) as

$$\begin{aligned} \dot{u}_p &= \dot{u}^T e + \ell^T \dot{\omega}^T b \\ \ddot{u}_p &= \ddot{u}^T e + \ell^T (\ddot{\omega}^T + \dot{\omega}^T \dot{\omega}^T) b \\ \delta r_p &= \delta u^T e + \ell^T \delta \alpha^T b \end{aligned} \quad (2.13)$$

To approximate the inertia force operator of the continuum, we interpolate the displacement u by linear shape functions in the form [2]

$$u = \sum_{I=1}^{npe} N_I u_I \quad (2.14)$$

where N_I denotes the spatial linear shape functions, u_I represents the degrees of freedom at the element nodes, and npe denotes the number of nodes per element. The virtual rotation $\delta \alpha$ can be similarly approximated. The discrete inertia force operator for the beam elements can thus be given by

$$\begin{aligned} \delta F^I &= \sum_{I=1}^{npe} \sum_{K=1}^{npe} \left\{ \delta u_I^T \rho A M_{IK}^E \frac{d^2 u_K}{dt^2} + \delta \alpha_I^T \rho J_I M_{IK}^E \frac{d^2 \omega_K}{dt^2} \right\} \\ &\quad + \sum_{I=1}^{npe} \delta \alpha_I^T D_I^E \end{aligned} \quad (2.15)$$

where

$$M_{IK}^E = \int_{\xi} N_I N_K d\xi, \quad D^E(\omega)_I d\xi$$

represent the element mass matrix and dynamic moment due to the centripetal acceleration vector, respectively. The former will be evaluated as a standard lumped mass matrix for

the computational efficiency of explicit integration techniques to be described in Section 5, and the latter will be evaluated from an average of the element nodal angular velocities.

2.3 Constraint Operator due to Joints

As a key attribute in the development of the present multibody dynamics analysis capability is the concept of software modularity, the kernel processing the joint constraints must remain independent from the other computational kernels. A joint is introduced to describe the interaction of two or more contiguous bodies (for more detailed discussions, see, e.g., Nikravesh[3]). Joint examples range from rigid connectors, which allows no relative motion between two bodies, to devices that allows the relative separation of the bodies (see Fig. 3).

Let us consider, as an example, a spherical joint that is characterized by imposing the equality of the positions of two connected bodies at a specific common location. The center of the spherical joint, called p , can be represented in terms of the two body-fixed frames (b_1^i, b_2^i, b_3^i) , and (b_1^j, b_2^j, b_3^j) , where the superscripts (i, j) designate the two contiguous bodies under consideration, respectively. The constraint condition that the two bodies at the point p must have the same translational motions in the e -basis can be expressed as

$$\Phi_s = u_i + s_p^i - u_j - s_p^j = u_i^T e + \ell_{pi}^T b_i - u_j^T e - \ell_{pj}^T b_j = 0 \quad (2.16)$$

To augment the constraint condition to the equations of motion via the method of Lagrange multipliers, we must obtain $\delta\Phi_s$ as

$$\delta\Phi_s = B_s \delta d = 0, \quad (2.17)$$

and the constraint Jacobian matrix B_s for the above example is given by

$$B_s = [I, (\tilde{l}_{pi} R_i)^T, -I, -(\tilde{l}_{pj} R_j)^T] \quad (2.18)$$

where I denotes the (3×3) identity matrix and δd is the virtual displacement vector

$$\delta d = [\delta u_i, \delta \alpha_i, \delta u_j, \delta \alpha_j]^T \quad (2.19)$$

The constraint force operator δF^C can then be derived as

$$\delta F^C = \sum_{I=1}^m \langle \delta u_I^T \quad \delta \alpha_I^T \rangle \left\{ \begin{matrix} B_r^{IT} \\ B_\omega^{IT} \end{matrix} \right\} \lambda^I \quad (2.20)$$

The case of nonholonomic constraints, although not elaborated herein, can be similarly derived.

3. Discretization of Internal Force Operator for Spatial Beams

The internal force operator is the second term in (2.1)

$$\delta F^S = \int_V \frac{\partial \delta r_i}{\partial x_j} \sigma_{ij}^e dV \quad (3.1)$$

where the virtual displacement gradient and the Cauchy stress tensor are expressed in the inertial frame, viz., e -basis. There are two major issues in the finite element discretization of the internal force operator for spatial beams for applications in multibody dynamics analysis. The first is to preserve rigid-body motions for beams that undergo large motions. The second is the choice of a convenient coordinate system so that stresses, strains, and other physical quantities computed based on that coordinate system maintain one-to-one correspondence to those measured in on-line sensors and actuators. This is an important consideration as the present multibody dynamics capability is to be used not only in off-line simulations but also in real-time loop-in-the-system computations. In other words, the strains gage readings and the computed strains should kinematically coincide with each other, and similarly for the actuator force which is based on the computed stresses to be applied to the system. To this end, we choose a convected frame such that one of its axes lies tangent to the deformed beam neutral axis. It will be shown that this choice allows us to develop a simple additive stress update procedure.

Figure 4 illustrates the relations among the three coordinate systems employed for the present purposes; the convected frame is characterized by the unit triad vector $a = \{a_1, a_2, a_3\}^T$ and is related to the inertial reference frame e by $a = T e$. The triad a is chosen to remain constant along each beam element. The difference between a and b can be used to model torsion and bending on the beam cross-section, and the rotation matrix S is introduced for this purpose as $b = S a$ such that $R = S T$.

The internal force operator can now be tensorially transformed into the convected frame a in terms of the convected frame components of the stress tensor (σ_{ij}^a) and a corresponding convected virtual displacement gradient as

$$\delta F^S \equiv \int_V \frac{\partial \delta r_i}{\partial x_j} \sigma_{ij}^e dV = \int_V T_{mi} \frac{\partial \delta r_i}{\partial \xi_k} \sigma_{mk}^a dV \quad (3.2)$$

where T_{mi} are the T -matrix components. By grouping the transformed deformation gradient components in terms of their symmetrical parts as

$$\delta \epsilon_{mk}^a \equiv \frac{1}{2} \left(T_{mi} \frac{\partial \delta r_i}{\partial \xi_k} + T_{ki} \frac{\partial \delta r_i}{\partial \xi_m} \right), \quad (3.3)$$

we arrive at the following expression

$$\delta F^S = \int_V \delta \epsilon_{mk}^a \sigma_{mk}^a dV = \int_V \{ \sigma_{\xi\xi} \quad \hat{\sigma}_{\xi\eta} \quad \hat{\sigma}_{\xi\xi} \} \begin{Bmatrix} \delta \epsilon_{\xi\xi} \\ \delta \hat{\epsilon}_{\xi\eta} \\ \delta \hat{\epsilon}_{\xi\xi} \end{Bmatrix} dV \quad (3.4)$$

where the notation

$$\xi_i = \{ \xi_1, \xi_2, \xi_3 \} = \{ \xi, \eta, \zeta \}, \quad (\cdot)_{ij} = (\cdot)_{ij} + (\cdot)_{ji}$$

denotes the coordinates of the convected reference frame and the engineering shear strain definitions, respectively. It can be shown that the remaining convected-frame strain components, $(\delta\epsilon_{\eta\eta}, \delta\epsilon_{\zeta\zeta}, \delta\epsilon_{\eta\zeta})$ are identically equal to zero due to the beam kinematic assumptions.

The following virtual strain-displacement relations can be deduced from (3.2) and (2.13):

$$\begin{Bmatrix} \delta\epsilon_{\xi\xi} \\ \delta\epsilon_{\xi\eta} \\ \delta\epsilon_{\xi\zeta} \end{Bmatrix} = \delta\gamma + \bar{\ell}^T \delta\kappa = [B] \begin{Bmatrix} \delta u \\ \delta\alpha \end{Bmatrix} \quad (3.5)$$

where

$$\delta\gamma = T \frac{\partial \delta u}{\partial \xi} + \begin{Bmatrix} 0 \\ -\delta\beta_3 \\ \delta\beta_2 \end{Bmatrix} \delta\kappa = \frac{\partial \delta\beta}{\partial \xi}, \quad \delta\beta = S^T \delta\alpha \quad (3.6)$$

In the above expressions, $\delta\gamma$ represents the membrane and two transverse shear strains, $\delta\kappa$ the torsion and two bending strains, $\delta\beta$ the virtual rotations of the cross-section referred to the convected frame, and $[B]$ the strain-displacement relation matrix. This strain operator $[B]$ is derived as

$$[B] = \begin{bmatrix} T \frac{\partial^a}{\partial \xi} & \bar{i}_1 S^T \\ 0 & S^T (\bar{\kappa}_S + I \frac{\partial^b}{\partial \xi}) \end{bmatrix}, \quad \bar{i}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad (3.7)$$

by invoking the following relations to effect the change of the body reference frame of the cross-section orientation in space with respect to the constant convected reference frame

$$\frac{\partial^a \delta\beta}{\partial \xi} = S^T \frac{\partial^a \delta\alpha}{\partial \xi} = S^T \left(\frac{\partial^b \delta\alpha}{\partial \xi} + \bar{\kappa}_S \delta\alpha \right) \quad (3.8)$$

$$\bar{\kappa}_S^T = \frac{\partial^a S}{\partial \xi} S^T \quad (3.9)$$

The internal force operator is put into the following form from (3.4) and (3.5) as

$$\delta F^S = \int_{\xi} \{ \delta\gamma^T N_{\gamma} + \delta\kappa^T M_{\kappa} \} d\xi = \int_{\xi} \{ \delta u^T \quad \delta\alpha^T \} [B]^T \begin{Bmatrix} N_{\gamma} \\ M_{\kappa} \end{Bmatrix} d\xi \quad (3.10)$$

where N_{γ} represent the axial and transverse shear forces per unit length, and M_{κ} represent the torsional and bending moments per unit length given as

$$N_{\gamma} = \int_A \sigma dA, \quad M_{\kappa} = \int_A \bar{\ell}^T \sigma dA \quad (3.11)$$

The computations of N_γ and M_κ are effected by using the following rate-type law:

$$\dot{\sigma}_{kl}^a \simeq c_{klmp} \dot{\epsilon}_{mp}^a \quad (3.12)$$

where c_{klmp} represents the material response tensor, and $\dot{\sigma}_{kl}^a$ and $\dot{\epsilon}_{mp}^a$ represent the convected frame stress and strain rates, respectively. This approximate constitutive law can be derived by transforming the Truesdell rate equation [1], which is an objective equation based on inertial components of Cauchy stresses and the velocity gradient tensor, to the convected basis. This equation is then integrated in time to obtain the following stress update formula

$$\sigma_{kl}^{a, n+1} = \sigma_{kl}^{a, n} + \int_{t^n}^{t^{n+1}} c_{klmp} \dot{\epsilon}_{mp}^a dt \quad (3.13)$$

$$c_{klmp} \Delta \epsilon_{mp}^a$$

Thus, the evaluation of the internal force operator consists of computing the strain-displacement matrix $[B]$ and the strain increments $\Delta\gamma$ and $\Delta\kappa$ to update N_γ and M_κ via (3.11) and (3.13). These will be discussed in the next two subsections.

3.1 Discrete Form of Internal Force Vector

The algorithmic treatment of the nonlinear stiffness operator is addressed in this section. The explicit generalized coordinate integrator to be discussed in Section 5 requires an evaluation of the internal force at a current time step t^{n+1} from the coordinates of the beam configuration at that time. The internal force is first evaluated on the element level for all the finite elements comprising the flexible component from (3.10) as

$$\begin{Bmatrix} S_I^e \\ S_I^b \end{Bmatrix}^E = [B_I^E]^T \begin{Bmatrix} N_\gamma \\ M_\kappa \end{Bmatrix}^E \quad (3.14)$$

after which these individual element computations are assembled to form the internal force of the discrete beam. The necessary computations to be described are the evaluations of the discrete strain operator $[B_I^E]$ defined in (3.7) and the resultant stresses N_γ and M_κ respectively.

A two-noded beam discretization is employed in the present work. For this case, the discrete strain operator $[B_I^E]$ modeling a consistent assumed constant strain field (3.5) which eliminates element locking for transverse-shear deformable beams is obtained as

$$[B_I^E] = \begin{bmatrix} -\frac{1}{l}T & \frac{1}{l}T & \frac{1}{2}i_1 S_1^T & \frac{1}{2}i_1 S_2^T \\ 0 & 0 & S_1^T (\frac{1}{2}\kappa_S - \frac{1}{l}I) & S_2^T (\frac{1}{2}\kappa_S + \frac{1}{l}I) \end{bmatrix} \quad (3.15)$$

where the subscripts (1,2) refer to the element node number. The convected frame T matrix, body frame curvature tensor $\bar{\kappa}_S$, and element neutral-axis length ℓ are constant quantities over the element domain, while the relative cross-section deformation S matrices are nodal quantities.

As discussed in (3.11) and (3.13), the resultant forces can be obtained by the following simple additive procedure:

$$\begin{Bmatrix} N_\gamma \\ M_\kappa \end{Bmatrix}^{n+1} = \begin{Bmatrix} N_\gamma \\ M_\kappa \end{Bmatrix}^n + \Delta \begin{Bmatrix} N_\gamma \\ M_\kappa \end{Bmatrix} \quad (3.16)$$

The resultant stress increments ΔN_γ and ΔM_κ are obtained via the linearized constitutive relations given by

$$\Delta N_\gamma = \begin{bmatrix} EA & 0 & 0 \\ 0 & GA & 0 \\ 0 & 0 & GA \end{bmatrix} \Delta \gamma, \quad \Delta M_\kappa = \begin{bmatrix} GJ & 0 & 0 \\ 0 & EI_2 & 0 \\ 0 & 0 & EI_3 \end{bmatrix} \Delta \kappa \quad (3.17)$$

where the strain increments $\Delta \gamma$ and $\Delta \kappa$ are defined as an incremental analogy to the virtual strains $\delta \gamma$ and $\delta \kappa$. A specific computational procedure designed for use with this incremental interpretation of the continuum-based formulation such that the computed finite strain increments are invariant to arbitrary rigid body motions is discussed in the next two sections.

3.2 Computations of Rotational Matrices and Total Curvature

To evaluate the discrete strain operator $[B_I^E]$ defined in (3.15), the computations of T and S rotational matrices and the total curvature $\bar{\kappa}_S$ are necessary. This is discussed in this subsection.

In computing T , observe that the neutral axis of the beam element is defined as the straight line connecting the two element nodes so that it can be directly calculated from the translational displacements output from the generalized coordinate integrator. First, denoting this unit tangent vector along the neutral axis as a_1 and utilizing the computed component a_1 , obtain the normal vector n^1 at node 1 to the surface formed by b_1^1 and a_1 where the subscript denotes the node number. Next, compute the angle tended by the two vectors $\cos \phi = a_1 \cdot b_1^1$. The rotational operator that rotates a to b can thus be constructed by

$$S^1(n^1, \phi) = \cos \phi I + (1 - \cos \phi) n^1 n^{1T} - \bar{n}^1 \sin \phi \quad (3.18)$$

Once S^1 is determined, T and S^2 are obtained by

$$T = S^{1T} R^1, \quad S^2 = R^2 T^T \quad (3.19)$$

When the angle ϕ small, it is difficult to obtain the required normal vector n^1 . For this case (usually at the beginning of the computations), we invoke the following alternative. The a_2 vector is defined as the cross product of a_1 with the b_3 axis of R^1 , and the remaining axis a_3 defined to complete the righthanded coordinate system. Unlike the first case, it is noted that T is first computed and S is computed according to

$$S^i = R^i T^T, \quad i = 1, 2. \quad (3.20)$$

These two methods of computing T and S matrices are shown in Fig. 5. The rotation matrices of the discrete strain operator (3.17) have thus been obtained.

The body frame components of the curvature tensor $\bar{\kappa}_S^T$ defined in (3.9) as

$$\bar{\kappa}_S^T = \frac{\partial^a S}{\partial \xi} S^T = \begin{bmatrix} 0 & \kappa_3 & -\kappa_2 \\ -\kappa_3 & 0 & \kappa_1 \\ \kappa_2 & -\kappa_1 & 0 \end{bmatrix}, \quad \kappa = \begin{Bmatrix} \kappa_1 \\ \kappa_2 \\ \kappa_3 \end{Bmatrix}$$

are equivalent to

$$\bar{\kappa}_S^T = \frac{\partial^e R}{\partial \xi} R^T \quad (3.21)$$

as the convected frame T matrix is defined to be constant along the element domain where the spatial differentiation is performed. As the Euler parameters of R are directly available from the generalized coordinate integrator to be discussed in Section 5, the Euler parameter representation of finite rotations can be exploited to achieve a simple and robust computation of the element curvature as follows. The Euler parameter - curvature representation is

$$\begin{Bmatrix} 0 \\ \kappa \end{Bmatrix} = 2 \begin{bmatrix} q_0 & q^T \\ -q & q_0 I - \bar{q} \end{bmatrix} \begin{Bmatrix} \frac{\partial q_0}{\partial \xi} \\ \frac{\partial q}{\partial \xi} \end{Bmatrix} = E(q_a) \frac{\partial q}{\partial \xi} \quad (3.22)$$

subject to the constraints

$$q_0^2 + q^T q = 1, \quad \frac{\partial q_0}{\partial \xi} q_0 + \frac{\partial q}{\partial \xi}^T q = 0. \quad (3.23)$$

The approximation adopted herein to evaluate (3.22) is given as

$$\frac{\partial q}{\partial \xi} = \frac{1}{\ell} (q_2 - q_1), \quad q_a = \frac{\frac{1}{2} (q_1 + q_2)}{\|\frac{1}{2} (q_1 + q_2)\|} \quad (3.24)$$

where the subscripts denote the nodal Euler parameters. The above approximation satisfies the constraint conditions in the discrete sense. It can be shown that this discrete computation is invariant to rigid rotations contained in the total nodal Euler parameters [4,5].

3.3 Computation of the Strain Increments

The stress update procedure given by (3.18) and (3.19) requires an effective computation of the strain increments. The accuracy in the computations of the internal force vector is not only dictated by the accuracy in computing $[B_I]$ matrix but also the strain increments as well. The strain increments are recalled from (3.8) by changing the variational operator δ to the incremental operator Δ :

$$\begin{aligned}\hat{\Delta}\gamma &= \mathbf{T} \frac{\partial \hat{\Delta}u}{\partial \xi} + \begin{Bmatrix} 0 \\ -\hat{\Delta}\beta_3 \\ \hat{\Delta}\beta_2 \end{Bmatrix} \\ \hat{\Delta}\kappa &= \frac{\partial \hat{\Delta}\beta}{\partial \xi}\end{aligned}$$

where $\hat{\Delta}u$ and $\hat{\Delta}\beta$ are displacement and rotation increments, respectively.

As the preservation of zero strain states for large rigid motions is a key factor, we distinguish the rotations due to rigid-body motions from those due to deformations within the convected frame components of the total rotational increments $\hat{\Delta}\beta$. To this end, we observe that the convected frame virtual rotations can be decomposed according to

$$\delta\beta = \delta\tau_a + \delta\varphi \quad (3.25)$$

where $\delta\varphi$ and $\delta\tau_a$ are the convected frame components of the rigid body and deformation incremental rotations, respectively, given by

$$\delta\tilde{\varphi}^T = \delta\mathbf{T} \mathbf{T}^T, \quad \delta\tilde{\tau}_a^T = \mathbf{S}^T \delta\mathbf{S} \quad (3.26)$$

With the above decomposition of virtual rotational motions interpreted incrementally, the incremental membrane and transverse shear strains can be computed by

$$\hat{\Delta}\gamma = \mathbf{T} \frac{\partial \hat{\Delta}u}{\partial \xi} + \begin{Bmatrix} 0 \\ -\hat{\Delta}\varphi_3 \\ \hat{\Delta}\varphi_2 \end{Bmatrix} + \begin{Bmatrix} 0 \\ -\hat{\Delta}\tau_{a3} \\ \hat{\Delta}\tau_{a2} \end{Bmatrix} \quad (3.27)$$

Likewise, the incremental curvature representing the torsion and bending strains is given by

$$\hat{\Delta}\kappa = \frac{\partial \hat{\Delta}\tau_a}{\partial \xi} \quad (3.28)$$

as the incremental rotations $\Delta\varphi$ defined from the $\mathbf{T}$ matrix are constant over the element length. The incremental translations are defined as the difference between a current and past displacement configuration as

$$\hat{\Delta}u \equiv u^{n+1} - u^n \quad (3.29)$$

The incremental rotations $\Delta\varphi$ and $\Delta\tau_a$ are interpreted from the rotation matrices ΔT and ΔS , respectively, which are defined in a manner consistent with (3.26) as

$$T^{n+1} = \Delta T T^n, \quad S^{n+1} = S^n \Delta S \quad (3.30)$$

With these preliminary concepts, a more detailed description of the computation of the present incremental strains are offered below.

3.3.1 Membrane Strain Increments

To achieve an objective computation, the first two terms of (3.27), given as

$$\hat{\Delta}\gamma_1 = T \frac{\partial \hat{\Delta}u}{\partial \xi} + \begin{Bmatrix} 0 \\ -\hat{\Delta}\varphi_3 \\ \hat{\Delta}\varphi_2 \end{Bmatrix}, \quad (3.31)$$

must be computed such that rotation increment $\hat{\Delta}\varphi$ compensates for the rigid motion contained within the displacement increment $\hat{\Delta}u$. This is accomplished by defining the skew-symmetric matrix containing the convected frame rotation increments as

$$\hat{\Delta}\tilde{\varphi}^T = (T^{n+1} - T^n) T^{n+\frac{1}{2}T} \quad (3.32)$$

This choice is a varied form of the approximation

$$T^{n+1} = e^{\Delta\tilde{\varphi}^T} T^n \simeq (I + \Delta\tilde{\varphi}^T) T^n$$

The reference frame $T^{n+\frac{1}{2}}$ is simply geometrically midway between the current and past convected reference frames, viz.,

$$\begin{aligned} T^{n+\frac{1}{2}} &= \Delta T^{n+\frac{1}{2}} T^n \\ \Delta T^{n+\frac{1}{2}} &\equiv \exp\left(\frac{1}{2}\Delta\tilde{\varphi}\right) \end{aligned} \quad (3.4.10)$$

This mid-configuration matrix is necessary to assure a skew symmetric result for $\hat{\Delta}\tilde{\varphi}^T$ from (3.32) as

$$\begin{aligned} \hat{\Delta}\tilde{\varphi}^T &= T^{n+1} T^{n+\frac{1}{2}T} - T^n T^{n+\frac{1}{2}T} \\ &= \Delta T^{n+\frac{1}{2}} - \Delta T^{n+\frac{1}{2}T} \end{aligned}$$

To incorporate the computation (3.32) within (3.31), it is necessary to use $T^{n+\frac{1}{2}}$ to rotate the translational derivatives within (3.31). The preceding incremental strain computation maintains correct rigid-body states for pure spatial rigid rotations, and pure spatial rigid rotations accompanied with a stretch of the neutral axis [4,5].

3.3.2 Transverse Shear and Curvature Strain Increments

Let us now concentrate on the incremental transverse shear and curvature strain increments involving deformational rotation increments. To achieve an incremental objectivity, the transverse shear incremental strains

$$\hat{\Delta}\gamma_2 = \begin{Bmatrix} 0 \\ -\hat{\Delta}\tau_{a_2} \\ \hat{\Delta}\tau_{a_1} \end{Bmatrix} \quad (3.33)$$

and incremental curvature strains

$$\hat{\Delta}\kappa = \frac{\partial \hat{\Delta}\tau_a}{\partial \xi} \quad (3.34)$$

are computed independently from the membrane strain $\hat{\Delta}\gamma_1$ as follows.

The rotation parameters characterizing the deformational rotation increments $\hat{\Delta}\tau_a$ correspond to the rotational vector parameterization of ΔS , the magnitudes of which will be limited at most to 45° in absolute angle. To compute $\hat{\Delta}\tau_a$, we will adopt the following incremental interpretation of the Rodrigues parameter representation of the angular variation [6] as

$$\Delta\tau_a = \frac{2}{1 + |\Theta_t|^2} [I + \tilde{\Theta}^T] \Delta\Theta_t \quad (3.35)$$

In the above, the Rodrigues parameters Θ_t and their increments $\Delta\Theta_t$ are determined from the S and ΔS matrices, respectively, via the relation [7]

$$\tilde{\Theta}_t = \frac{(S - S^T)}{1 + \text{tr}S} \quad (3.36)$$

In practice, in order to maintain high computational accuracy within the constant strain approximation context, we introduce a mid-element shear matrix S_a defined as

$$S_a = R_a T^T$$

where the matrix R_a corresponds to the geometric average of the total cross-section orientation matrices at the element nodes and T is the element convected frame. The mid-element total orientation matrix R_a can be immediately found using the normalized average of the nodal Euler parameters as defined in (3.24). The Rodrigues parameters Θ_t and $\Delta\Theta_t$ are then obtained from the matrices

$$S_a^{n+1} = R_a^{n+1} T^{n+1T}, \quad \Delta S_a = S_a^{nT} S_a^{n+1} \quad (3.37)$$

using (3.36).

The elemental curvature (3.34) is computed by differentiating (3.35) as

$$\begin{aligned} \frac{\partial \hat{\Delta}\tau_a}{\partial \xi} &= \frac{2}{1 + |\Theta_t|^2} [I + \tilde{\Theta}^T] \frac{\partial \Delta\Theta_t}{\partial \xi} + \frac{2}{1 + |\Theta_t|^2} \frac{\partial \tilde{\Theta}_t}{\partial \xi} \Delta\Theta_t \\ &\quad - \frac{4 \Theta_t \cdot \frac{\partial \Theta_t}{\partial \xi}}{(1 + |\Theta_t|^2)^2} [I + \tilde{\Theta}^T] \Delta\Theta_t \end{aligned} \quad (3.38)$$

The derivative terms are approximated over an element of length ℓ as

$$\begin{aligned}\frac{\partial \Theta_i}{\partial \xi} &= \frac{1}{\ell} (\Theta_{i2} - \Theta_{i1}) \\ \frac{\partial \Delta \Theta_i}{\partial \xi} &= \frac{1}{\ell} (\Delta \Theta_{i2} - \Delta \Theta_{i1})\end{aligned}$$

using the nodal Rodrigues parameters Θ_{i1} and $\Delta \Theta_{i1}$, $i = 1, 2$. It is noted that this approximation is valid as the Rodrigues parameters are defined relative to a common elemental reference frame T . The non-derivative terms are evaluated using the mid-element shear matrices defined in (3.37).

4. Discrete Equations of Motion for Multibody Systems

In the preceding two sections, we have formulated to compute the inertia force operator δF^I , the internal force operator δF^S , and the constraint force operator δF^C . To complete the derivation of the equations of motion, we recall the external force operator δF^E from (2.2) and discretize it to take the form

$$\delta F^E = \int_V \delta r_i f_i dV = \int_{\xi} \{ \delta u^T \delta \alpha^T \} \left\{ \begin{matrix} f^e \\ f^b \end{matrix} \right\} d\xi \quad (4.1)$$

where f^e represents the inertial components of a force per unit length acting on the beam neutral axis and f^b represents the body-fixed components of a moment per unit length acting on the beam cross-section. Finally, the traction operator δF^T that acts on the exterior surfaces of the deformable body as natural boundary conditions can be discretized as

$$\delta F^T = \int_S \delta r_i t_i dS = \int_S \{ \delta u^T \delta \alpha^T \} \left\{ \begin{matrix} t_i^e \\ t_i^b \end{matrix} \right\} dS \quad (4.2)$$

Assembling all the operators in the virtual work expression (2.2) and observing the variational displacement $\{ \delta u^T \delta \alpha^T \}$ are arbitrary, we arrive at the following discrete equations of motion for constrained multibody systems:

$$\begin{bmatrix} M & 0 \\ 0 & J \end{bmatrix} \begin{Bmatrix} \ddot{u} \\ \dot{\omega} \end{Bmatrix} + B^T \lambda = \begin{Bmatrix} Q_u \\ Q_\omega \end{Bmatrix} \quad (4.3)$$

$$\Phi(u, \dot{u}, R, \omega, t) = 0$$

where

$$\begin{Bmatrix} Q_u \\ Q_\omega \end{Bmatrix} = \begin{Bmatrix} f^e - S^e(u, q) \\ f^b - D(\omega) - S^b(u, q) \end{Bmatrix} \quad (4.4)$$

in which $D(\omega)$ represents the nonlinear acceleration, S the internal force vector, f the external force vector, and $B^T \lambda$ the constraint force vector. As an additional number of unknown Lagrange multipliers λ for each constraint condition have been introduced along

with the generalized coordinates for each degree of freedom, the above system of equations must be augmented with the constraint equations themselves to achieve a determined system of equations.

It should be emphasized that both the translational displacement u and the angular velocity ω embody the large rigid-body motions as well as flexible deformations. As such, the inertia force expression is the same for the rigid bodies and the flexible bodies. Hence, no additional degrees of freedom is necessary for incorporating any flexible members. This simplicity plays an important role in the solution process of the present capability.

5. Solution Procedures for Constrained Multibody Systems

The numerical solution procedure for MBD systems which we advocate in this chapter is termed a *staggered MBD solution procedure* that solves the generalized coordinates in a separate module from that for the constraint force [9]. A major advantage of such a partitioned solution procedure is that additional analysis capabilities such as active controller and design optimization modules can be easily interfaced without embedding them into a monolithic program. The solution of the equations of motion for constrained multibody systems as derived in (4.3), unlike typical structural dynamics problems, must satisfy at each time-marching step the system constraints, be they holonomic or nonholonomic, or time-specified maneuvers. Because of this distinctive requirement, the reliability and cost of a multibody analysis package can be strongly affected by how efficiently and accurately the constraints are preserved during the numerical solution stage.

In the present analysis capability we have chosen a hierarchical approach. In this approach we eliminate the constraint degrees of freedom if they are associated only with open rigid links and no external torque or active control devices are attached to those joints. Otherwise, the Lagrange multipliers that correspond to the rest of the system constraints are computed via the so-called stabilized procedure, which will be described shortly. This procedure requires a reformulation of the algebraic constraint conditions to a set of parabolic differential equations such that the constraint forces can also be integrated in time.

To solve for the generalized coordinates of the multibody system, the equations of motion are partitioned according to the translational and the rotational coordinates. This sets the stage for an efficient treatment of the rotational motions via the singularity-free Euler parameters. The resulting partitioned equations of motion are then integrated via a two-stage explicit stabilized algorithm for updating both the translational coordinates and angular velocities. Once the angular velocities are obtained, the angular orientations are updated via the mid-point implicit formula employing the Euler parameters.

When the two algorithms, namely, the two-stage explicit algorithm for the generalized coordinates and the implicit staggered procedure for the constraint Lagrange multipliers,

are brought together in a staggered manner, they constitute a staggered explicit-implicit procedure as detailed below.

5.1 Constraint Handling Techniques

In principle, it is better to eliminate the constraint conditions if possible if the corresponding forces are not needed for design or interface with other analysis modules. For example, if the system consists of open-tree configurations and no active controller is applied, then it is best to eliminate the joint constraint attributes. On the other hand, when the system includes multiple closed-loop configurations or active controllers are present on several joints, then it becomes important to compute the Lagrange multipliers as accurately as possible.

The present capability employs two complementary procedures for handling the system constraints. In one procedure, we present an implementation procedure for eliminating the open-tree constraints via a coordinate partitioning strategy in conjunction with a so-called *arrow head* solution technique. Emphasis here has been to implement the procedure in parallel machines. In the second procedure, we present incorporate a stabilization procedure for solving the Lagrange multipliers [8,9]. A distinct feature of this stabilization procedure is that it can be implemented in a stand-alone module, thus can be interfaced not only with the equation solver for rigid-body systems but with that for flexible-body systems as well.

5.1.1 Parallel Implementation of Coordinate Partitioning Technique

In the first technique, a projection matrix that spans the null space of the constraint Jacobian matrix $[B]$ is first constructed. A parallel methodology based on a arrow head algorithm then can be applied to the resulting complementary set of equations of motion. We will present the procedure for open-tree systems and for a system that contains closed-loops, a cut-joint technique can be used so that the present scheme can be equally applied.

Let us introduce a projection matrix A such that, when its transposed matrix acts on the constraint force $B^T \lambda$, it gives

$$A^T B^T \lambda = 0 \quad (5.1)$$

This projection matrix can be obtained by expressing the physical velocity $\dot{d}$ in terms of the independent velocities $\dot{d}^f$ and their time derivatives as

$$\dot{d} = A \dot{d}^f, \quad \ddot{d} = A \ddot{d}^f + \dot{A} \dot{d}^f \quad (5.2)$$

Due to the property of (5.1), premultiplying the equations of motion (4.3) by A^T yield

$$A^T M \ddot{d} = A^T Q \quad (5.3)$$

In conventional procedure, $\ddot{\mathbf{d}}$ in the above equation is replaced by (5.2b) and $\ddot{\mathbf{d}}^f$ is then solved from the reduced equations of motion. In the solution to be described below, instead of solving the reduced equations of motion, we augment (5.2b) to (5.3) to form an arrow-head matrix equation [10,11].

$$\begin{bmatrix} \mathbf{M} & -\mathbf{M}\mathbf{A} \\ -\mathbf{A}^T\mathbf{M} & 0 \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{d}} \\ \ddot{\mathbf{d}}^f \end{Bmatrix} = \begin{Bmatrix} \mathbf{M}\dot{\mathbf{A}}\dot{\mathbf{d}}^f \\ -\mathbf{A}^T\mathbf{Q} \end{Bmatrix} \quad (5.4)$$

which can be partitioned as [10,11]

$$\begin{bmatrix} M_{(1,1)} & 0 & 0 & 0 & \dots & D_{(1,n+1)} \\ 0 & M_{(2,2)} & 0 & 0 & \dots & D_{(2,n+1)} \\ 0 & 0 & M_{(3,3)} & 0 & \dots & D_{(3,n+1)} \\ 0 & 0 & 0 & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & M_{(n,n)} & \dots \\ D_{(n+1,1)} & D_{(n+1,2)} & D_{(n+1,3)} & \dots & \dots & 0 \end{bmatrix} \begin{Bmatrix} \ddot{d}_1 \\ \ddot{d}_2 \\ \ddot{d}_3 \\ \dots \\ \ddot{d}_n \\ \ddot{d}^f \end{Bmatrix} = \begin{Bmatrix} g_1 \\ g_2 \\ g_3 \\ \dots \\ g_n \\ g^f \end{Bmatrix} \quad (5.5)$$

where n is the total number of bodies in the system. Decomposed in a manner convenient for parallel computations, one obtains

$$\begin{aligned} \mathbf{M}_j \ddot{\mathbf{u}}_j + \mathbf{D}_{(j,n+1)} \ddot{\mathbf{d}}^f &= \mathbf{g}_j, \quad j = 1, \dots, n \\ \sum_{j=1}^n \mathbf{D}_{(n+1,j)} \ddot{\mathbf{d}}_j &= \mathbf{g}^f \end{aligned} \quad (5.6)$$

where

$$\sum_{j=1}^n \mathbf{D}_{(n+1,j)} = -\sum_{j=1}^n \mathbf{A}_j^T \mathbf{M}_j, \quad \mathbf{D}_{(j,n+1)} = -\mathbf{M}_j \mathbf{A}_j, \quad j = 1, \dots, n$$

$$\mathbf{g}_j = (\mathbf{M}\dot{\mathbf{A}}\dot{\mathbf{d}}^f)_j, \quad j = 1, \dots, n \quad \mathbf{g}^f = -\sum_{j=1}^n \mathbf{A}_j^T \mathbf{Q}_j$$

Each diagonal submatrix $\mathbf{M}_j$ represents the local mass matrix which is decoupled and can be factorized concurrently. An off-diagonal submatrix $\mathbf{D}_j$ denotes the coupling between two connecting bodies in the system. Since $\mathbf{M}$ is a constant matrix, (5.14a) becomes

$$\ddot{\mathbf{d}}_j = \mathbf{M}_j^{-1} (\mathbf{D}_{(j,n+1)} \ddot{\mathbf{d}}^f - \mathbf{g}_j) \quad (5.7)$$

Substituting (5.15) into (5.14b) gives the well-known *Schur complement*

$$\sum_{j=1}^n \mathbf{D}_{(n+1,j)} \mathbf{M}_j^{-1} \mathbf{D}_{(j,n+1)} \ddot{\mathbf{d}}^f = \sum_{j=1}^n \mathbf{D}_{(n+1,j)} \mathbf{M}_j^{-1} \mathbf{Q}_j - \sum_{j=1}^n \mathbf{D}_{(n+1,j)} \dot{\mathbf{A}}_j \dot{\mathbf{d}}^f \quad (5.8)$$

The preceding treatment of the reduced equations of motion provides several parallel computational features. First, the parallelism in multibody systems can be exploited by

mapping each processor onto a group of bodies so that independent computations such as the left hand side of (5.16) can be performed concurrently. Second, since M_i is a constant mass matrix, it needs to be factored only once. Third, to solve for $\ddot{d}^f$, a parallel sparse solver may be utilized. Finally, once $\ddot{d}^f$ is obtained, the evaluation of $\ddot{d}$ from (5.15) is also easily parallelizable.

5.1.2 Stabilized Constraint Force Solution Procedure

When the Lagrange multipliers cannot be eliminated or are to be retained for other purposes, one must solve for them. Of several techniques available [12-15], we adopt a form of stabilized differential equations for the Lagrange multipliers [8,9]. As we shall see, this choice allows the use of a partitioned solution procedure to solve the generalized coordinates separately from the Lagrange multipliers. To effect a partitioned solution of the constraints, a stabilized companion differential equation for the constraint forces is formed by adopting the penalty procedure [16]. The penalty procedure uses the equations

$$\lambda = \frac{1}{\epsilon} \Phi \quad (5.9)$$

as the basic constraint equations instead of (4.3b) for both the holonomic and nonholonomic constraint conditions. In the stabilization introduced in [8], one adopts the following time-differentiated expression for the holonomic case:

$$\dot{\lambda} = \frac{1}{\epsilon} B \dot{d}, \quad \dot{d} = \begin{Bmatrix} \dot{u} \\ \omega \end{Bmatrix} \quad (5.10)$$

The numerical solution to the above companion differential equation is effected as follows.

The constrained equations of motion (3.32) are integrated once from (3.20) using the implicit integration rule

$$d^{n+\frac{1}{2}} = d^n + \delta \ddot{d}^{n+\frac{1}{2}}, \quad \delta = \frac{h}{2}$$

as

$$d^{n+\frac{1}{2}} = \delta M^{-1} (Q^{n+\frac{1}{2}} - B^T \lambda^{n+\frac{1}{2}}) + d^n \quad (5.11)$$

This expression is substituted into (5.18) to obtain the stabilized differential equation for the Lagrange multipliers

$$\epsilon \dot{\lambda}^{n+\frac{1}{2}} + \delta B M^{-1} B^T \lambda^{n+\frac{1}{2}}$$

The same integration rule is applied to this equation to result in the discrete update

$$(\epsilon I + \delta^2 B M^{-1} B^T) \lambda^{n+\frac{1}{2}} = \epsilon \lambda^n + r_\lambda^{n+\frac{1}{2}} \quad (5.12)$$

$$r_{\lambda}^{n+1/2} = \delta^2 B M^{-1} Q^{n+1/2} + \delta B \dot{d}^n \quad (5.13)$$

The solution procedure for λ presented above can now be invoked in a staggered manner in conjunction with the generalized coordinate solver to be described below.

5.2 Time Integration of the Equations of Motion

Once λ is computed by the procedure described in Section 5.1 or $\bar{d}$ when using the partitioning algorithm, one still needs to compute $\dot{d}$, and u and the angular orientation matrices R (and their parameters) at each time step. This task is carried out by employing an explicit-implicit transient analysis algorithm that exploits the special kinematical relationships of the generalized rotational coordinates vs. the angular velocity, namely, the Euler parameters. First, the integration of the translational coordinates and the angular velocity is accomplished by the central difference formula. It should be mentioned that the use of the central difference formula does impose a stepsize restriction due to its stability limit ($\omega_{max} h \leq 2$) where ω_{max} is the highest angular velocity of the system components for rigid-body systems or the highest frequency of the entire flexible members for flexible-body systems. The simplicity of its programming effort and robustness of its solution results can often become compelling enough to adopt an explicit formula, which is the view taken here.

5.2.1 Explicit Generalized Coordinate Integrator

In conventional structural dynamics analysis, explicit time integration of the equations of motion by the central difference formula involves the following two updates per step:

$$\begin{cases} \dot{d}^{n+1/2} = \dot{d}^{n-1/2} + h \ddot{d}^n \\ d^{n+1} = d^n + h \dot{d}^{n+1/2} \end{cases} \quad (5.14)$$

where the rotational degrees of freedom are assumed to be a infinitesimal vector. Unfortunately, this simplistic procedure is not directly applicable to the rotational part of the equations of motion involve large motions such that ω is not directly integrable to yield the total rotational quantities except for some special kinematic configurations. This motivates us to partition $\dot{d}$ into the translational velocity vector, $\dot{u}$, which is directly integrable and the angular velocity vector, ω , which is not, and treat them differently, viz.:

$$\bar{d} = \begin{Bmatrix} \ddot{u} \\ \dot{\omega} \end{Bmatrix}, \quad \dot{d} = \begin{Bmatrix} \dot{u} \\ \omega \end{Bmatrix} \quad (5.15)$$

The equations of motion (5a) can be partitioned according to the above partitioning:

$$\begin{bmatrix} M_d & 0 \\ 0 & M_\omega \end{bmatrix} \begin{Bmatrix} \ddot{u} \\ \dot{\omega} \end{Bmatrix} = \begin{Bmatrix} Q_d \\ Q_\omega \end{Bmatrix} \quad (5.16)$$

where

$$\begin{Bmatrix} Q_d \\ Q_\omega \end{Bmatrix} = \begin{Bmatrix} f_d - D_d(\dot{d}) - S_d(d, e) - B_d^T \lambda \\ f_\omega - D_\omega(\omega) - S_\omega(d, e) - B_\omega^T \lambda \end{Bmatrix} \quad (5.17)$$

in which the subscripts $(\dot{u}, \omega)$ refer to the translational and the rotational motions, respectively, f is the external force vector, D is the generalized damping force including the centrifugal force, S is the internal force vector including member flexibility, q is the angular orientation parameters, B_u and B_ω are the partition of the combined gradient matrices of the constraint conditions (3).

The application of the central difference formula to obtain $(\dot{u}, \omega)^{n+1}$ for the above equations of motion is given as follows. First, assume that $u^{n+1/2}$ and $q^{n+1/2}$ are already computed so that we can compute $\ddot{u}^{n+1/2}$ and $\ddot{\omega}^{n+1/2}$:

$$\begin{Bmatrix} \ddot{u}^{n+1/2} \\ \ddot{\omega}^{n+1/2} \end{Bmatrix} = -M^{-1} \begin{Bmatrix} D_d^{n+1/2} + S_d^{n+1/2} - B_d^T \lambda^{n+1/2} \\ D_\omega^{n+1/2} + S_\omega^{n+1/2} - B_\omega^T \lambda^{n+1/2} \end{Bmatrix} \quad (5.18)$$

Second, we update the translational velocity and the angular velocity vectors at the step $(n+1)$ by

$$\begin{cases} \dot{u}^{n+1} = \dot{u}^n + h\ddot{u}^{n+1/2} \\ \omega^{n+1} = \omega^n + h\ddot{\omega}^{n+1/2} \end{cases} \quad (5.19)$$

Third, we update the translational displacement, u , by

$$u^{n+3/2} = u^{n+1/2} + h\dot{u}^{n+1} \quad (5.20)$$

A requirement in the explicit integration of the multibody dynamics equations is that the damping terms and the moment due to the centripetal acceleration D_d and D_ω have to be approximated as accurately as possible. This is in part because active control strategies are strongly affected by the level of damping, and in part due to numerical instability when the term $D_\omega = \tilde{\omega} J \omega$ is poorly computed. To see this, let us integrate just a set of rotational equations of motion

$$J \dot{\omega} + \tilde{\omega} J \omega = f_\omega$$

by the central difference formula to obtain

$$\omega^{n+\frac{1}{2}} = \omega^{n-\frac{1}{2}} + hJ^{-1}(f^n - \tilde{\omega}^n J \omega^n) \quad (5.21)$$

Observe that by virtue of the central difference formula in a full step integration case, that is, marching from $t = nh$ to $t = (n+1)h$, the angular velocity vectors are available only at the half-step intervals, viz., $\omega^{n-3/2}$ and $\omega^{n-1/2}$. Hence, ω^n that is needed in computing $\tilde{\omega}^n J \omega^n$ is not available. One algorithmic way to overcome this deficiency is to advance

only one half step at each integration step. In other words, the next step marches from $t = (n + \frac{1}{2})h$ to $t = (n + \frac{3}{2})h$ instead of $t = (n + 1)h$ to $t = (n + 2)h$. In this way, at any integration step, the $\tilde{\omega} J \omega$ term can be computed accurately. The same procedure is also used to integrate the translational displacements and velocities. To summarize, we offer the following computational sequence which is called a *two-stage explicit integration scheme* [9].

(a)	$u^{n+\frac{1}{2}} = u^{n-\frac{1}{2}} + h \dot{u}^n$
(b)	$\dot{u}^{n+\frac{1}{2}} = \dot{u}^{n-\frac{1}{2}} u^n$
(c)	$\omega^{n+\frac{1}{2}} = \omega^{n-\frac{1}{2}} \omega^n$
(d)	$q^{n+\frac{1}{2}} = q(\omega^{n+\frac{1}{2}})$
(e)	$S^{n+\frac{1}{2}} = S(u^{n+\frac{1}{2}}, q^{n+\frac{1}{2}})$
(f)	$D^{n+\frac{1}{2}} = D(\omega^{n+\frac{1}{2}}), \quad f^{n+\frac{1}{2}} = f(t^{n+\frac{1}{2}})$
(g)	$Q^{n+\frac{1}{2}} = Q(f^{n+\frac{1}{2}}, S^{n+\frac{1}{2}}, D^{n+\frac{1}{2}})$
(h)	$\lambda^{n+\frac{1}{2}} = \lambda(\lambda^n, Q^{n+\frac{1}{2}})$
(i)	$\ddot{u}^{n+\frac{1}{2}} = M^{-1} (Q_u^{n+\frac{1}{2}} - B_u^T \lambda^{n+\frac{1}{2}})$
(j)	$\dot{\omega}^{n+\frac{1}{2}} = J^{-1} (Q_\omega^{n+\frac{1}{2}} - B_\omega^T \lambda^{n+\frac{1}{2}})$

The computation of the Euler parameters $q^{n+\frac{1}{2}}$ in Step (d) will be discussed in the next subsection. The computation of the internal force $S^{n+\frac{1}{2}}$ in Step (e) was detailed in Section 3. Finally, the computation of the Lagrange multipliers $\lambda^{n+\frac{1}{2}}$ was presented in Section 5.1.

5.2.2 Rotational Parameter Integration

The preceding subsection discussed the two-stage explicit integrator as applied to the translational displacement and velocity coordinates, and the angular velocity vector. To obtain the rotational orientation parameters, we employ the Euler parameters: $(q_0, \mathbf{q})$ defined as

$$\begin{Bmatrix} q_0 \\ \mathbf{q} \end{Bmatrix} = \begin{Bmatrix} \cos \frac{\theta}{2} \\ \mathbf{n} \sin \frac{\theta}{2} \end{Bmatrix}, \quad \text{with constraint } q_0^2 + \mathbf{q}^T \mathbf{q} = 1 \quad (5.22)$$

The rotation matrix is represented as a function of the Euler parameters as

$$\mathbf{R} = \begin{bmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1 q_2 + q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 - q_0 q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_1 q_3 + q_0 q_2) & 2(q_2 q_3 - q_0 q_1) & 2(q_0^2 + q_3^2) - 1 \end{bmatrix} \quad (5.23)$$

The choice of the Euler parameters allows one to utilize the following kinematical equations

$$\dot{q} = A(\omega) q, \quad A(\omega) = \frac{1}{2} \begin{bmatrix} 0 & -\omega_e^T \\ \omega_e & \tilde{\omega}_e \end{bmatrix} \begin{Bmatrix} q_0 \\ q \end{Bmatrix}, \quad q = \begin{Bmatrix} q_0 \\ q \end{Bmatrix} \quad (5.24)$$

which can be integrated by the following constraint satisfying form of the trapezoidal formula

$$\frac{1}{h} (q^{n+1} - q^n) = A(\omega^{n+\frac{1}{2}}) \frac{1}{2} (q^{n+1} + q^n) \quad (5.25)$$

Due to the structure of A, the solution matrix can be analytically inverted such that the discrete orientation update

$$q^{n+1} = \frac{1}{D} \left[I + \frac{h}{2} A(\omega^{n+\frac{1}{2}}) \right] \left[I + \frac{h}{2} A(\omega^{n+\frac{1}{2}}) \right] q^n \quad (5.26)$$

where

$$D = 1 + \frac{h^2}{4} (w_1^2 + w_2^2 + w_3^2)$$

results. The final result is normalized to satisfy the inherent Euler parameter constraint condition. The resulting update (5.26) involves only explicit computations and is readily incorporated into the two-stage explicit integration procedure [9].

6. Illustrative Examples

In order to illustrate the features of the present multibody dynamics analysis capability, four example problems will be described: flexible double pendulum, three-dimensional maneuvering of space crane, parallel computations of a simplified automobile suspension model, and deployment of a beam from a rigid guide.

The first example problem is the dynamics of a double pendulum made of flexible spatial beam members. This problem exhibits both the three-dimensional deformations and large rigid-body motions. It has been demonstrated that, for a free rigid body undergoing only rotational motions, both the energy and the momentum can be conserved by employing a problem-specific algorithm [17]. However, for general multibody systems that involve both the rotational and translational motions in a coupled manner plus complicated system constraints, the most one can realistically hope for is the preservation of the system energy if the system is conservative. The ability of the present capability to conserve the system energy is to be demonstrated by this example.

The double pendulum is modeled with two beams; a spherical joint connects the last node of the first beam to the first node of the second beam and also a spherical joint is attached to the root node of the first beam. It is noted that the joint connection can easily be accounted for from a finite element assemblage which leaves the rotational degrees of freedom free at the hinge locations. The result from this modeling is the same as that obtained using the Lagrange multiplier solver by retaining the joint constraint conditions,

thus establishing the viability of interfacing the spatial beam models with rigid elements. In the first case, the pendulum is subjected to a gravity field in the vertical z-direction and an initial velocity impulse in the horizontal x-y plane such that solely rigid motion is excited. The problem is run for four cases of increasing beam flexibility as follows:

1.	$EA = 1.0 \times 10^4$	$GA = 0.5 \times 10^4$
2.	$EA = 1.0 \times 10^3$	$GA = 0.5 \times 10^3$
3.	$EA = 2.0 \times 10^2$	$GA = 1.0 \times 10^2$
4.	$EA = 1.0 \times 10^2$	$GA = 0.5 \times 10^2$

with the remaining parameters

$$\rho A = 1 \text{ lbm/in} \quad \rho I = .833 \times 10^{-3} \text{ lbm in} \quad l = 1 \text{ in}$$

held constant. The initial velocity impulse, and the spatial trajectories of the mass center of the second beam as projected on the x-y and x-z planes are given in Fig. 6. The trajectory of the first case coincides exactly with a rigid body solution to the problem, and the slight deviation of the trajectories due to the increasing flexibility can be seen. The energy time histories for the problem, given in Fig. 7, verify the high-accuracy level of the present capability as energy is practically conserved. Again, the accurate modeling of the internal forces in the three dimensional environment is demonstrated by the negligible strain energy contribution for all of the four flexible cases. The time integration of the spatial kinematics preserves the balance between the kinetic and potential energies of the problem. Next, the flexible double pendulum is given an initial velocity impulse to excite deformation motion as well as the rigid motion. For this case the parameters used were

$$EA = 1.8 \times 10^8 \text{ lb}, \quad GA = 0.9 \times 10^8 \text{ lb}, \quad EI = 1.4 \times 10^8 \text{ lb in}^2$$

$$\rho A = .98 \text{ lbm/in}, \quad \rho I = 0.67 \text{ lbm in}, \quad l = 120 \text{ in}.$$

The initial velocity profile, the resulting time histories of several deformed configurations and energy time history are given in Fig. 8, which shows the large spatial rotation and deformation capabilities of the present beam formulation. The energy conservation is retained for the computations of spatial deformations.

The second example is a rigid space crane model as shown in Fig. 9. This problem was previously analyzed by Gawronski and Ih [18] by employing a tailored problem-specific formulation. In the present modeling analysis, we model the links to be connected by spherical and revolute joints. Hence, the joint torques and forces ($B^T \lambda$) are computed as part of the problem rather than to be obtained by a post-processing. The motion of the crane is triggered by a specified tip velocity from the initial position to the final position, a nonholonomic constraint condition as shown in Fig. 10. The time histories of torques in the three joints are shown in Fig. 11 (for more details, see, [11]). In addition, during the time integration process, the maneuvering of the crane is controlled under the desired trajectories which are given in two different vertical planes as illustrated in Fig. 12.

The third problem is a double-wishbone automobile suspension system that is analyzed by the parallel algorithm of the present capability. The problem definition and the model

are described in Nikravesh [3] and is shown in Fig. 13. The vehicle is partitioned into six subsystems where six independent processors can be assigned to each of the subsystems so that the null space of the constraint Jacobian matrix can be constructed in parallel. Note that the suspension systems possess four sets of springs and dampers with given locations, spring and damping coefficients. The tires of the vehicle are modeled by using unilateral spring elements. Initially, the vehicle is positioned in a height of one meter from the ground with initial velocities equal to zero. When the vehicle is released, gravity acts as the external loads so that the vehicle experiences a free fall until it hits the ground. Figure 14 illustrates one of the springs that reacts to the ground constraint during one second simulation runtime. Table 1 provides the speedup and computational efficiency vs the number of shared-memory processors. It is noted that a speedup of about 4.5 is achieved when six processors are utilized, thus indicating the potential of parallel computations toward real-time simulations and the arrow-head solution strategy adopted in the present capability.

The final example is the deployment of a flexible beam from a rigid fixture, which simulates paper ejection from a printer or deployment of a solar panel from the stowed satellite structure. In extending the present three-dimensional flexible beam module to model axially moving three-dimensional beams, the present capability has added the following two salient features (see, for details, Downer and Park [19]). First, Hamilton's law of varying action is chosen as a variational frame for the discrete approximation of the governing dynamics of moving beams. As such, the discretization of a complicated convective acceleration is avoided as the use of Hamilton's law requires only the first material time-derivative of the convective displacements. Second, in the finite element approximation of the moving beam, we fix the number of discrete points and allow the length of each discrete beam element to vary.

This extended capability was used to analyze a thin beam deploying from a rigid horizontal guide into a uniform gravitational field and compared to results reported in Mansfield and Simmonds [20], who modeled the beam as an axially moving elastica. Figures 15 and 16 illustrate the basic kinematics and the beam deployment results vs different deployment velocities. Two characteristic parametrizations of the problem are dimensionless weight-to-stiffness ratio $\mu = mgl^4 / EI$ and a dimensionless velocity $\nu = Vl\sqrt{ml / EI}$. In these expressions, m is the mass per unit area of the paper, g is the acceleration of gravity, ℓ is the length of the paper, EI the bending stiffness, and V the constant velocity of the paper ejection. The analysis was performed with $\mu = 50$ and $\nu^2 = 100, 50, 20, 10$ corresponding to $V = 92, 65, 41, 20$ in/sec, respectively. The trajectories of tip of the beam tip for the various deployment speeds, as obtained by the present capability, are shown in Fig. 16 and compared to the static shape of a fully extended cantilevered beam. The present results are comparable to those reported in Mansfield and Simmonds [20].

7. Discussions

A multibody dynamics analysis capability that consists of independent analysis kernel modules and that utilizes a partitioned computational procedure is presented. The present software architecture is designed to run the resulting program efficiently both on sequential as well as parallel computers. The presently fully operational modules are a three-dimensional beam processor, a joint and stabilized constraint processor, an explicit time-integration module, a Hamiltonian plane beam deployment dynamics, a conjugate gradient-based rigid-element parallel solver, and a linearized vibration control synthesizer. Future planned additions include a three-dimensional plate element processor, active controllers for large slewing motions, and parallel flexible multibody solver. It should be emphasized that the primary goal of the present undertaking is to develop a highly accurate multibody dynamics capability for space applications as well as high-precision manufacturing real-time simulations such electronics assembly. This does not, however, preclude the use of the present capability for conventional multibody applications as demonstrated in the analysis of automobile suspension dynamics.

Acknowledgements

The work reported herein was supported by the NASA/Langley Research Center under Grant NAG-1-756, Air Force Office of Scientific Research under Grant F49620-87-C-0074, the NASA Graduate Students Research Program through grant number NGT-50254, from JPL under NASA Headquarters/OAET through Pathfinder Project. The authors wish to thank Drs. Jerry Housner of NASA/Langley Research Center, Dr. S. Wu of Afosr and Dr. B. K. Wada of JPL for their interest and support during the course of the present work.

References

1. Malvern, L.E., *Introduction to the Mechanics of a Continuous Medium*, Prentice-Hall, Inc., Englewood Cliffs, N. J., (1969).
2. Zienkiewicz, O.C., *The Finite Element Method*, 3rd ed., McGraw Hill Book Company, Ltd, London, (1977).
3. Nikraves, P.E., *Computer-Aided Analysis of Mechanical Systems*, Prentice Hall, (1988).
4. Downer, D. D., Park, K. C. and J. C. Chiou, "A Computational Procedure for Multibody Systems Including Flexible Beam Dynamics," to appear in *Computer Methods in Applied Mechanics and Engineering*, 1991; Proc. the 1990 AIAA Dynamics Specialist Conference, Paper No. AIAA-90-1237, Long Beach, Calif., 5-6 April 1990.
5. Downer, J.D., "A Computational Procedure for the Dynamics of Flexible Beams within Multibody Systems," PhD Thesis, University of Colorado, (1990).
6. Cardona, A. and Geradin, M., "A Beam Finite Element Nonlinear Theory with Finite Rotations," *Int. J. Num. Meth. Eng.* 26 (1988) 2403-2438.

7. Rankin, C.C., and Brogan, F.A., "An Element Independent Corotational Procedure for the Treatment of Large Rotations," *J. Pressure Vessel Technology* 108 (1986) 165-174.
8. Park, K.C., and Chiou, J.C., "Stabilization of Computational Procedures for Constrained Dynamical Systems," *Journal of Guidance, Control and Dynamics* 11 (1988) 365-370.
9. Park, K.C., Chiou, J.C., and Downer, J.D., "Explicit-Implicit Staggered Procedure for Multibody Dynamics Analysis," *J. Guidance, Control, and Dynamics* 13 (1990) 562-570.
10. Chiou, J.C., "Constraint Treatment Techniques and Parallel Algorithms for Multibody Dynamic Analysis," PhD Thesis, University of Colorado, (1990).
11. Chiou, J. C., Park, K. C. and C. Farhat, "A Natural Partitioning Scheme for Parallel Simulation of Multibody Systems," to be presented at the 1991 AIAA Structures, Dynamics and Materials Conference, Paper No. AIAA-91-1111, Baltimore, Maryland, 8-10 April 1991.
12. Baumgarte, J.W., "Stabilization of Constraints and Integrals of Motion in Dynamical Systems," *Comp. Meth. Appl. Mech. Engr.* 1 (1972) 1-16.
13. Baumgarte, J.W., "A New Method of Stabilization for Holonomic Constraints," *Journal of Applied Mechanics* 50 (1983) 869-870.
14. Wehage, R.A. and Haug, E.J., "Generalized Coordinate Partitioning for Dimension Reduction in Analysis of Constrained Dynamic Systems," *ASME J. of Mech. Design* 104 (1982) 247-255.
15. Walton, W.C. and Steeves, E.C., "A New Matrix Theorem and its Application for Establishing Independent Coordinates for Complex Dynamical Systems with Constraints," NASA TR-R326 (1984).
16. Lanczos, L., *The Variational Principles of Mechanics*, 4th ed., University of Toronto Press, 1970.
17. Simo, J. C. and Wong, K. K., "Unconditionally Stable Algorithms for Rigid Body Dynamics that Exactly Preserve Energy and Momentum," *Int. j. num. meth. engr.*, 31(1), 19-52(1991).
18. Gawronski, W. and Ih, C-H, C., "3D Rigid Body Dynamic Modeling of Space Crane for Control Design and Analysis," JPL D-6878, Nov. 17, 1989.
19. Downer, D. D. and Park, K. C., "Formulation and Solution of Inverse Spaghetti Problem: Application to Beam Deployment Dynamics," to be presented at the 1991 AIAA Structures, Dynamics and Materials Conference, Paper No. AIAA-91-1113, Baltimore, Maryland, 8-10 April 1991.
20. Mansfield, L. and Simmonds, J.G., "The Reverse Spaghetti Problem: Drooping Motion of an Elastica Issuing from a Horizontal Guide," *J. Appl. Mech.* 54 (1987) 147-150.

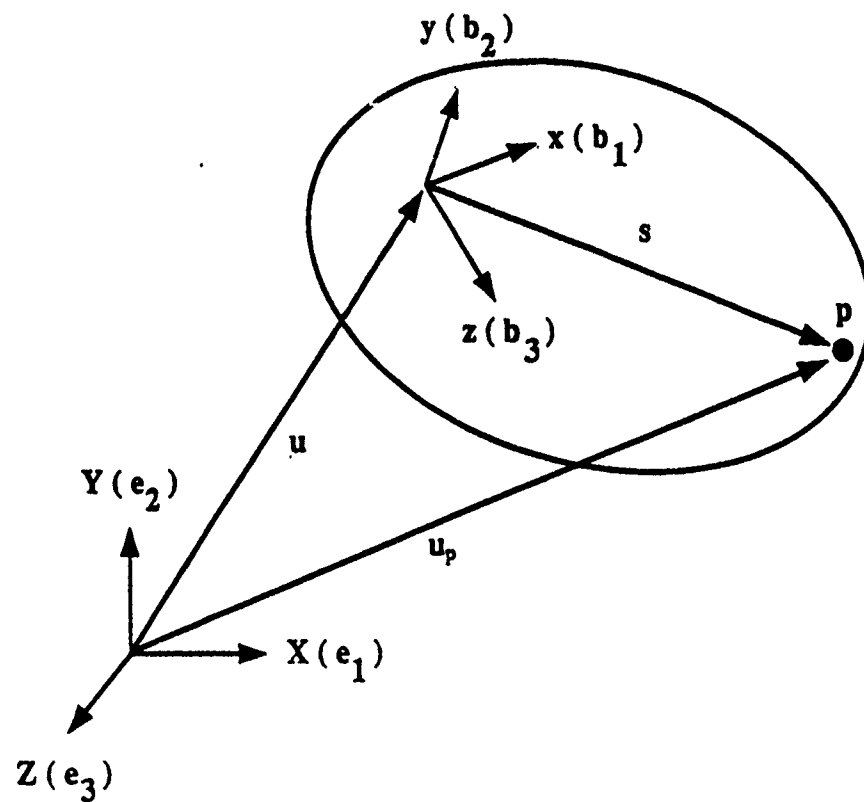


Fig. 1 Translation and Rotation of a Body
in Three-Dimensional Space

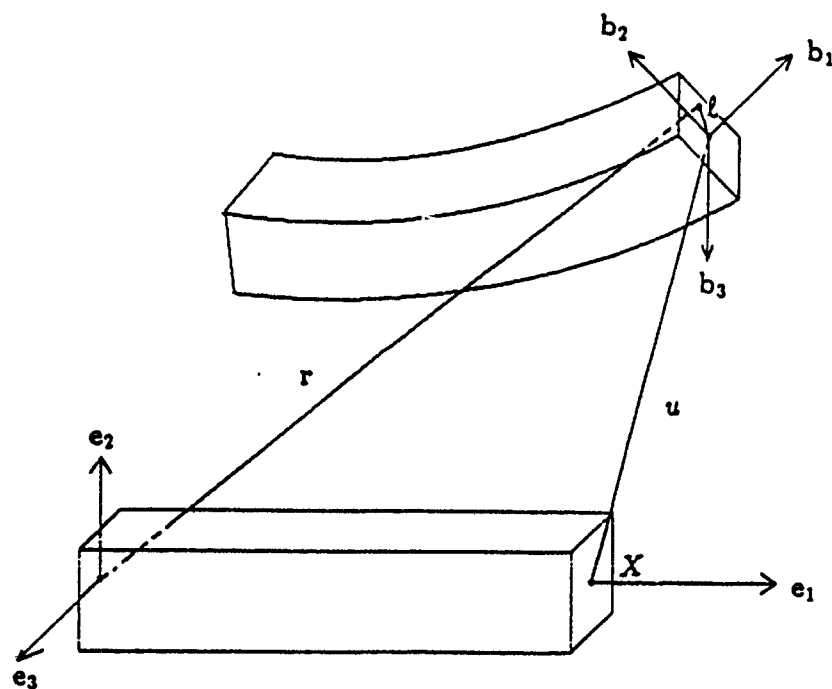


Fig. 2 Spatial Beam Kinematics

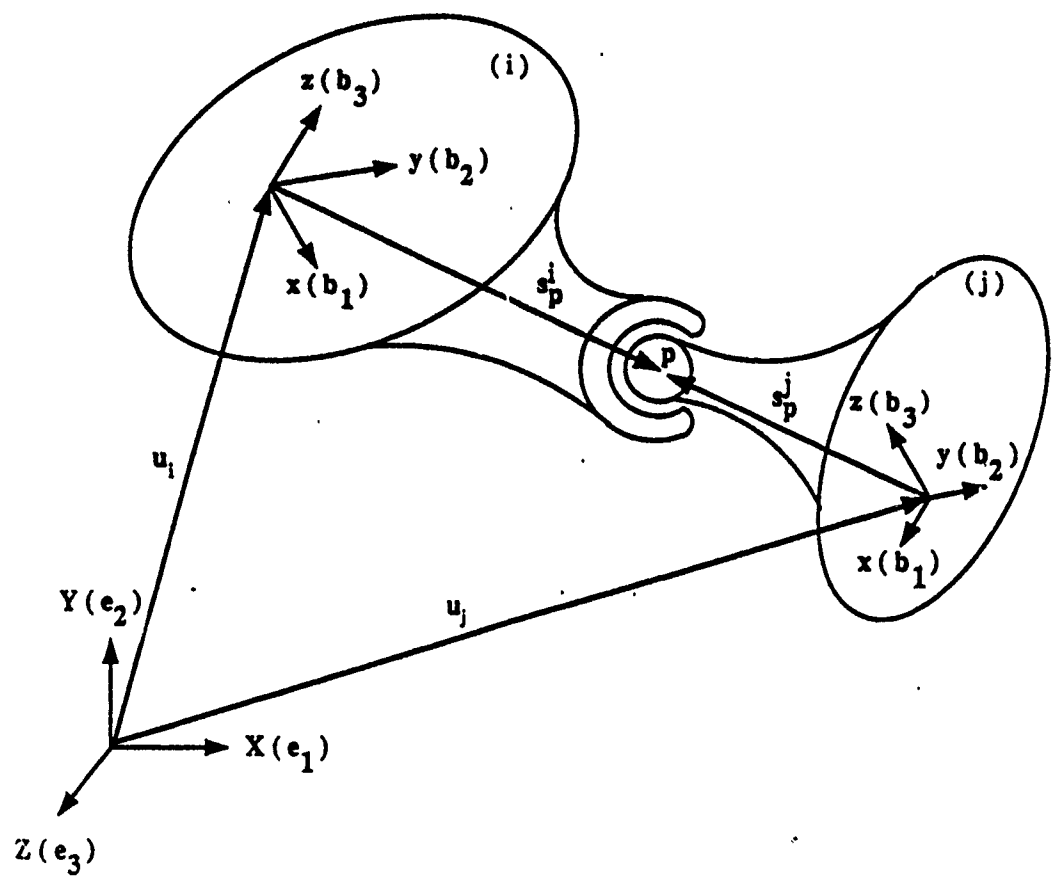


Fig. 3 A Spherical Joint

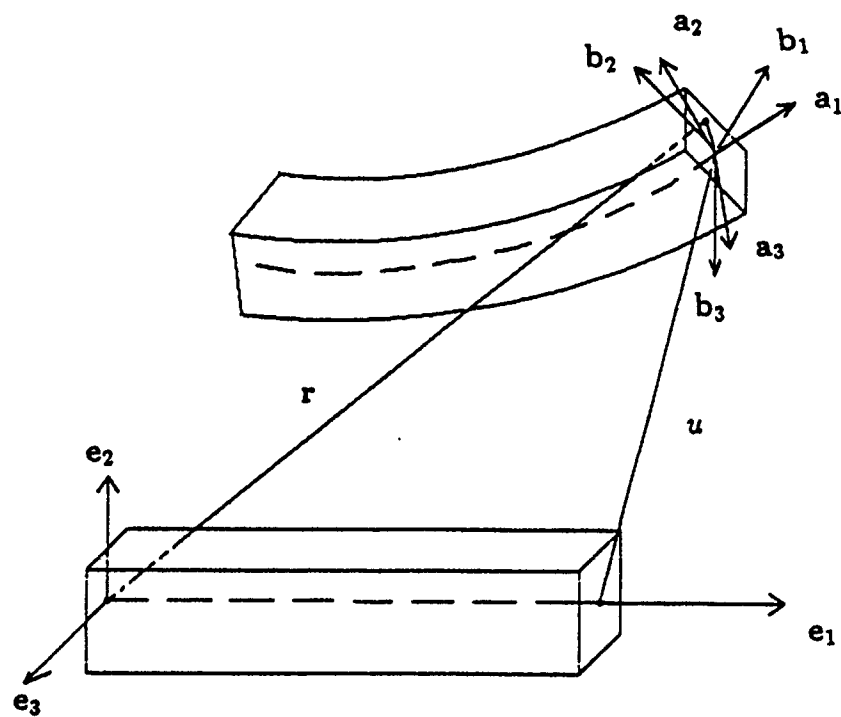


Fig. 4 Convected Reference Frame

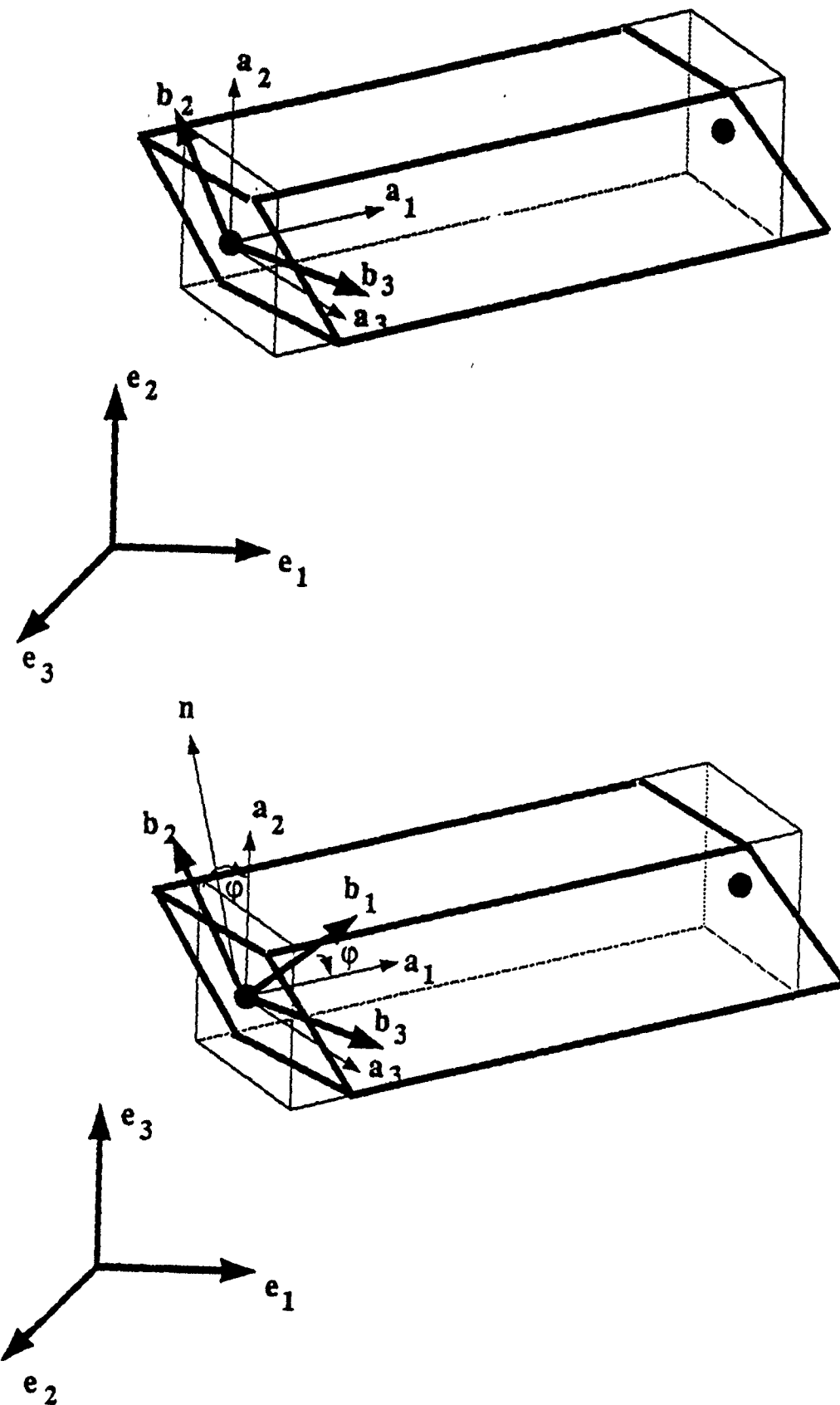
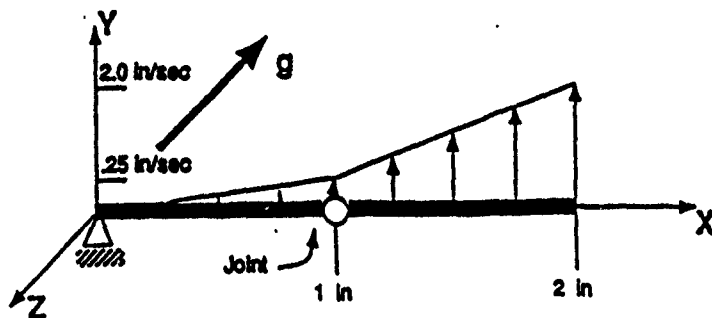


Fig. 5 Reference Frames of Discrete Beam

Initial Beam Position vs. Initial Velocity Profile



Material Properties

$$A = .01 \text{ in}^2$$

$$I = .00833333 \text{ in}^4$$

$$\rho = .259 \text{ lbm/in}^3$$

$$(1) E = 1.0 \times 10^6 \text{ lb/in}^2$$

$$G = .5 \times 10^6 \text{ lb/in}^2$$

$$(2) E = 1.0 \times 10^5 \text{ lb/in}^2$$

$$G = .5 \times 10^5 \text{ lb/in}^2$$

$$(3) E = 2.0 \times 10^4 \text{ lb/in}^2$$

$$G = 1.0 \times 10^4 \text{ lb/in}^2$$

$$(4) E = 1.0 \times 10^4 \text{ lb/in}^2$$

$$G = .5 \times 10^4 \text{ lb/in}^2$$

Mesh

8 Linear Elements / Beam

Fig. 6a Spatial Motion of Double Pendulum: Problem Data

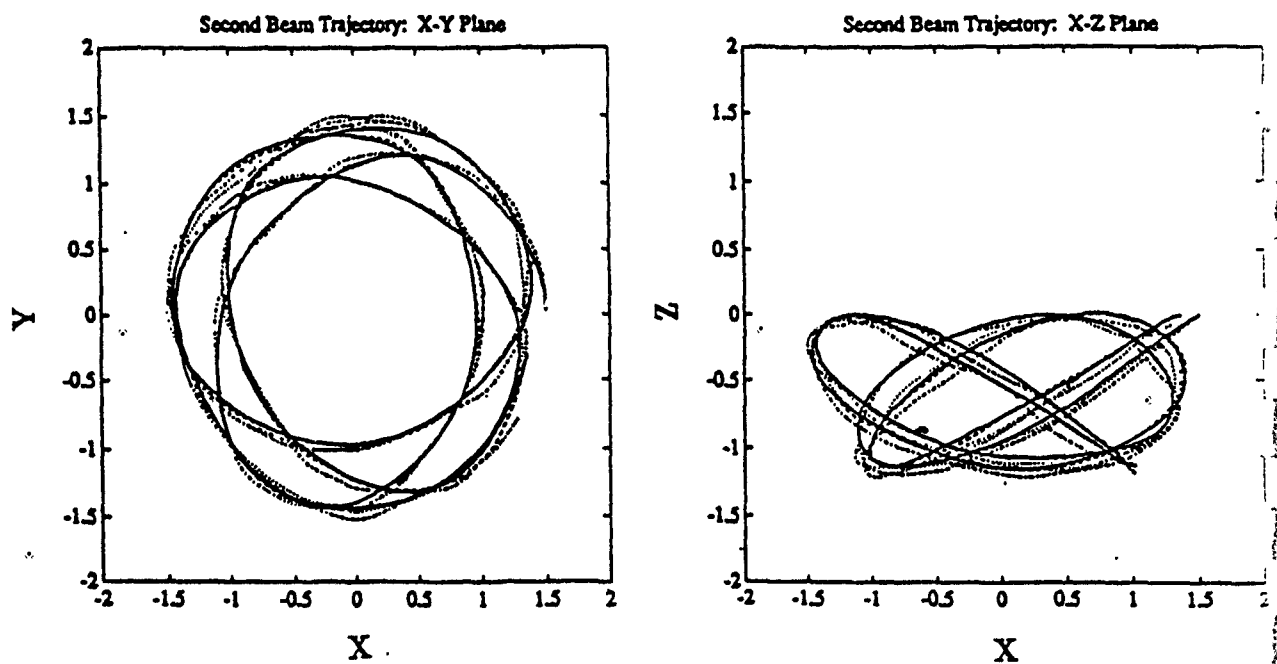


Fig. 6b Spatial Motion of Double Pendulum: Projected Trajectories

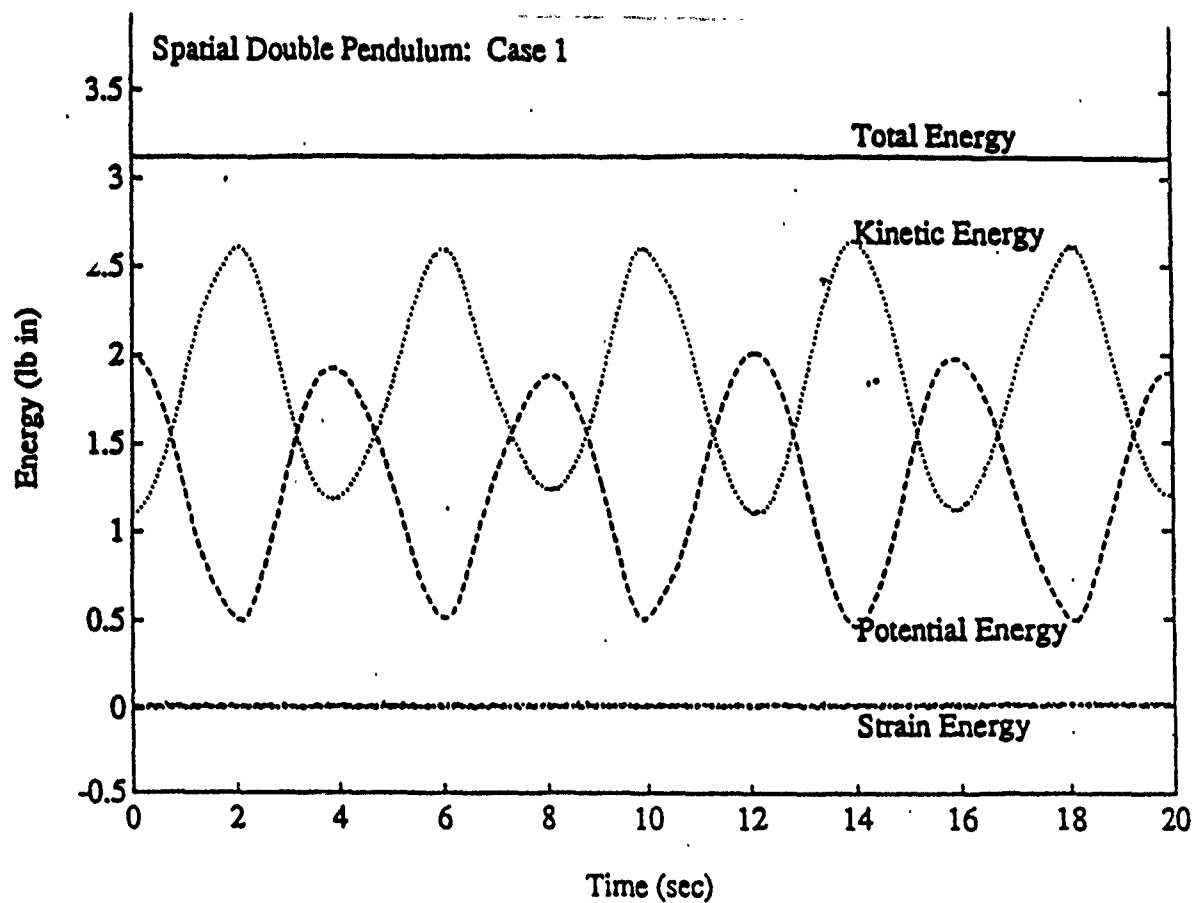


Fig. 7a Spatial Motion of Double Pendulum: Energy History (1)

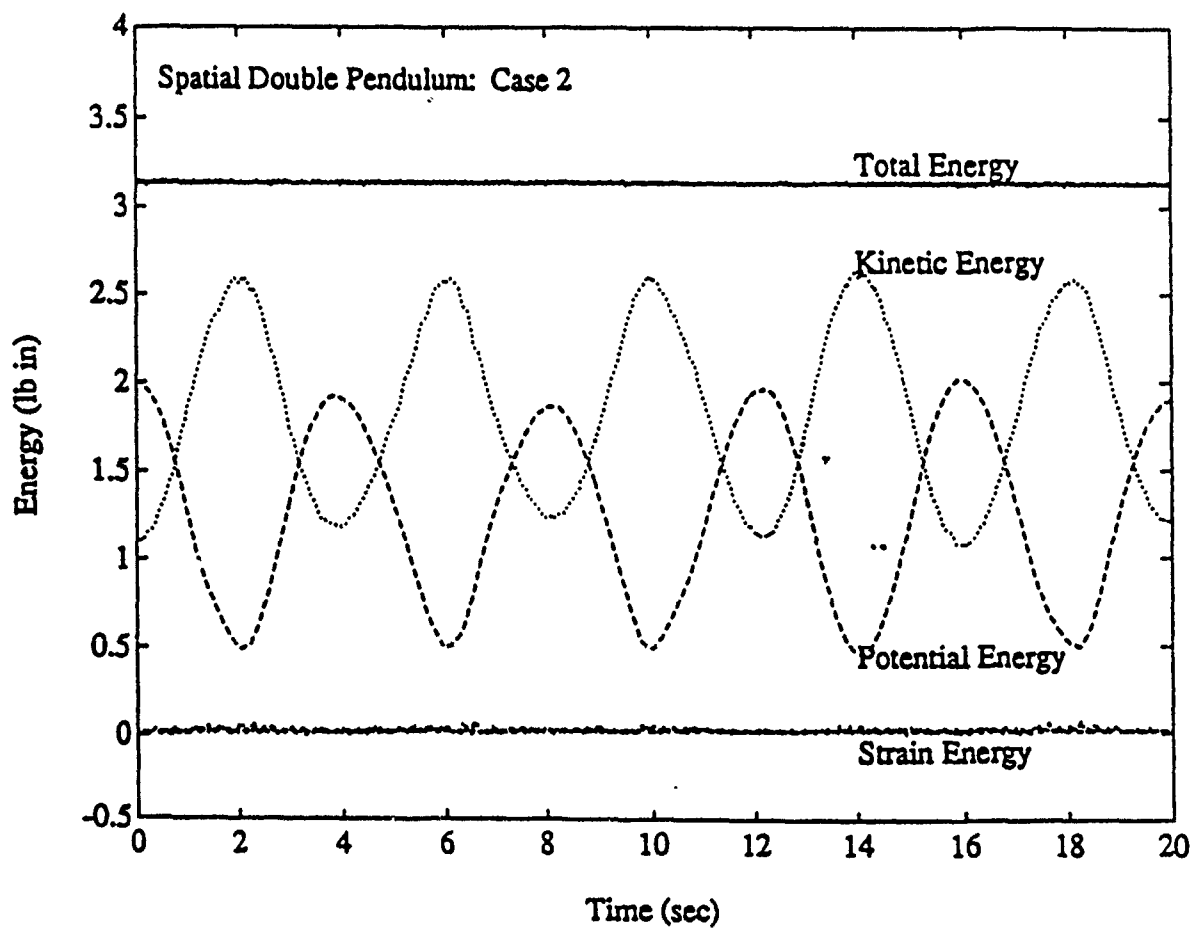


Fig. 7b Spatial Motion of Double Pendulum: Energy History (2)

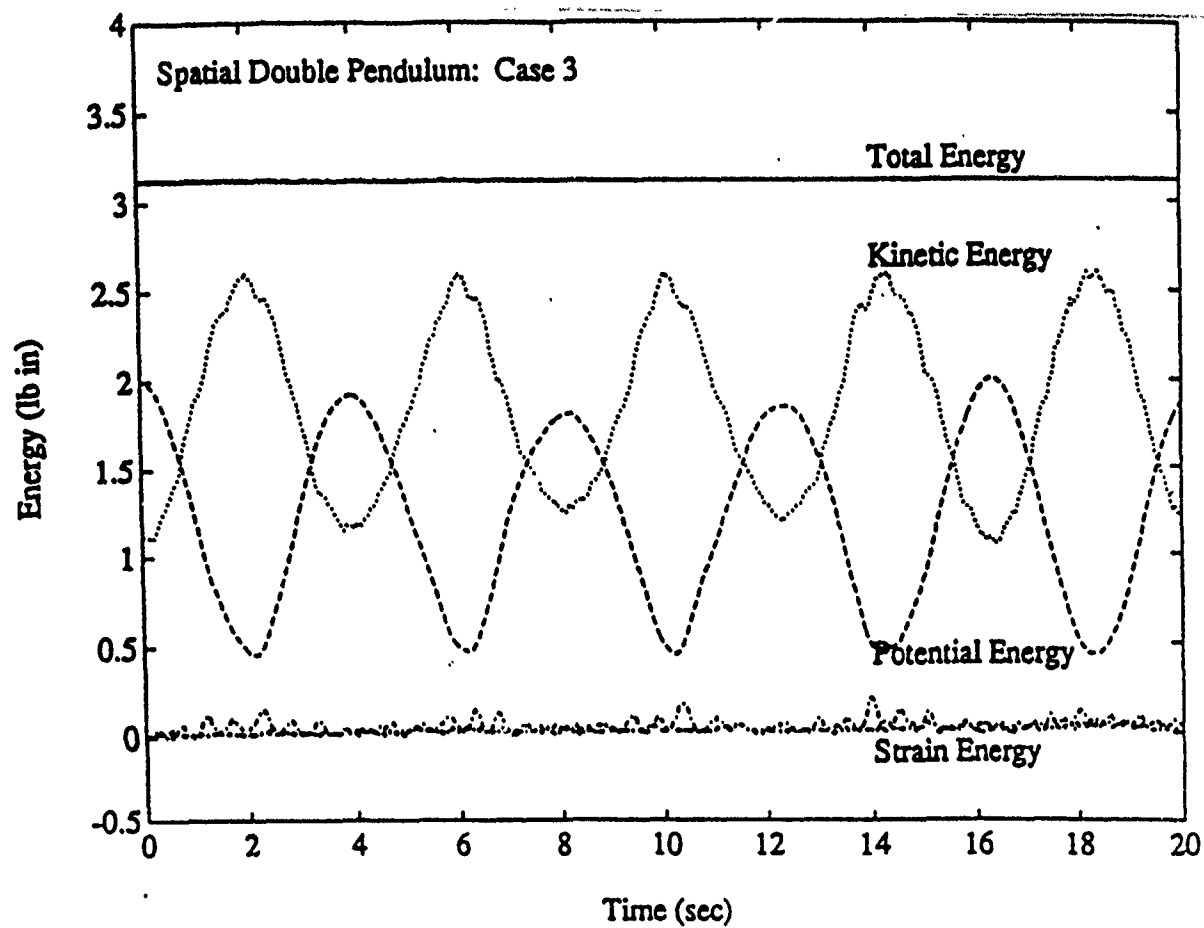
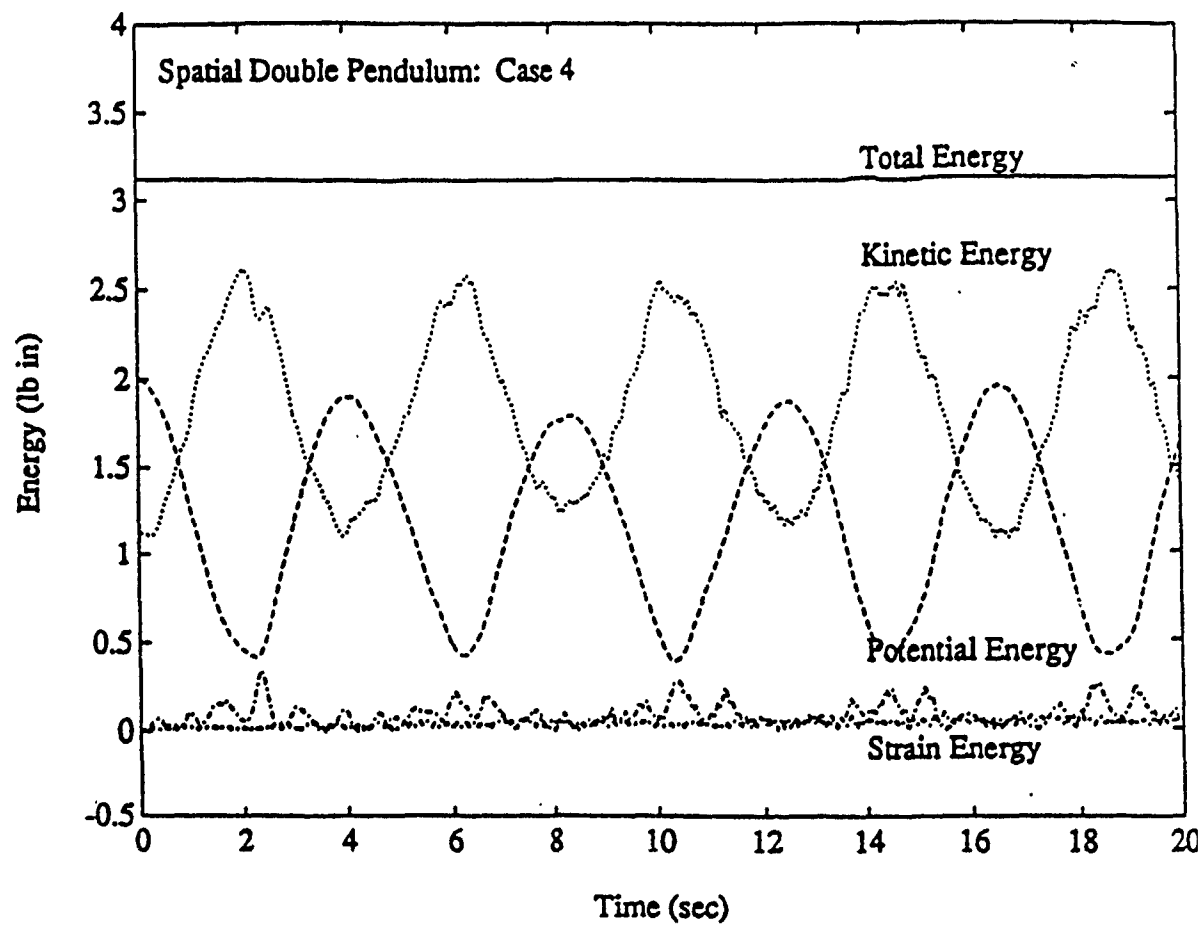
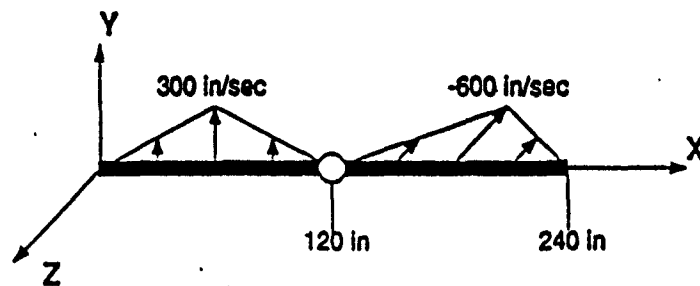


Fig. 7c Spatial Motion of Double Pendulum: Energy History (3)



Initial Beam Position vs. Initial Velocity Profile



Material Properties

$E = 2.0 \times 10^7$ lb/in²
 $G = 1.0 \times 10^7$ lb/in²
 $A = 9.0$ in²
 $I = 6.75$ in⁴
 $\rho = .011$ lbm/in³

Mesh

8 Linear Elements / Beam

Fig. 8a Spatial Motion of Double Pendulum: Problem Data

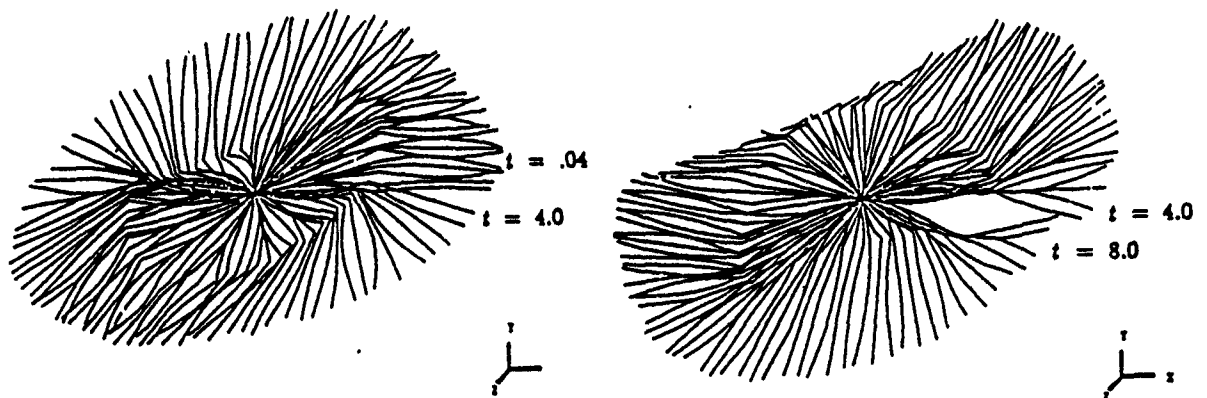


Fig. 8b Spatial Motion of Double Pendulum: Motion History

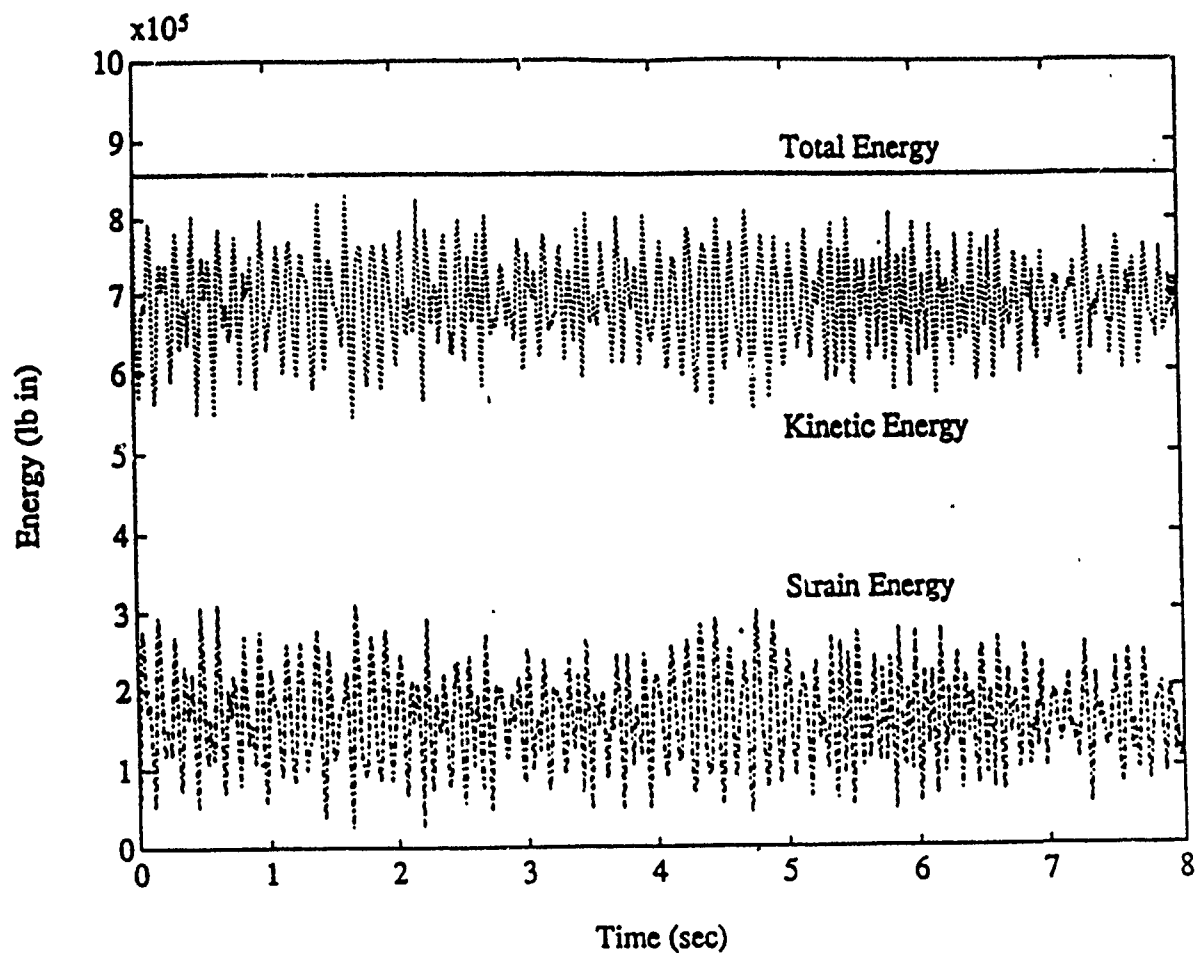


Fig. 8c Spatial Motion of Double Pendulum: Energy History

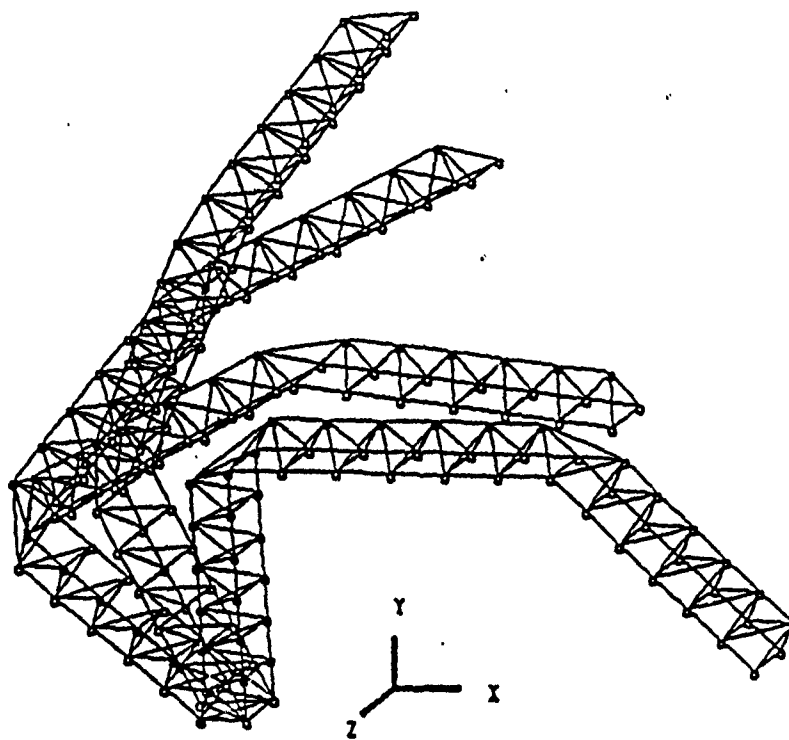


Fig. 9 Crane Configuration and Subsequent Motion

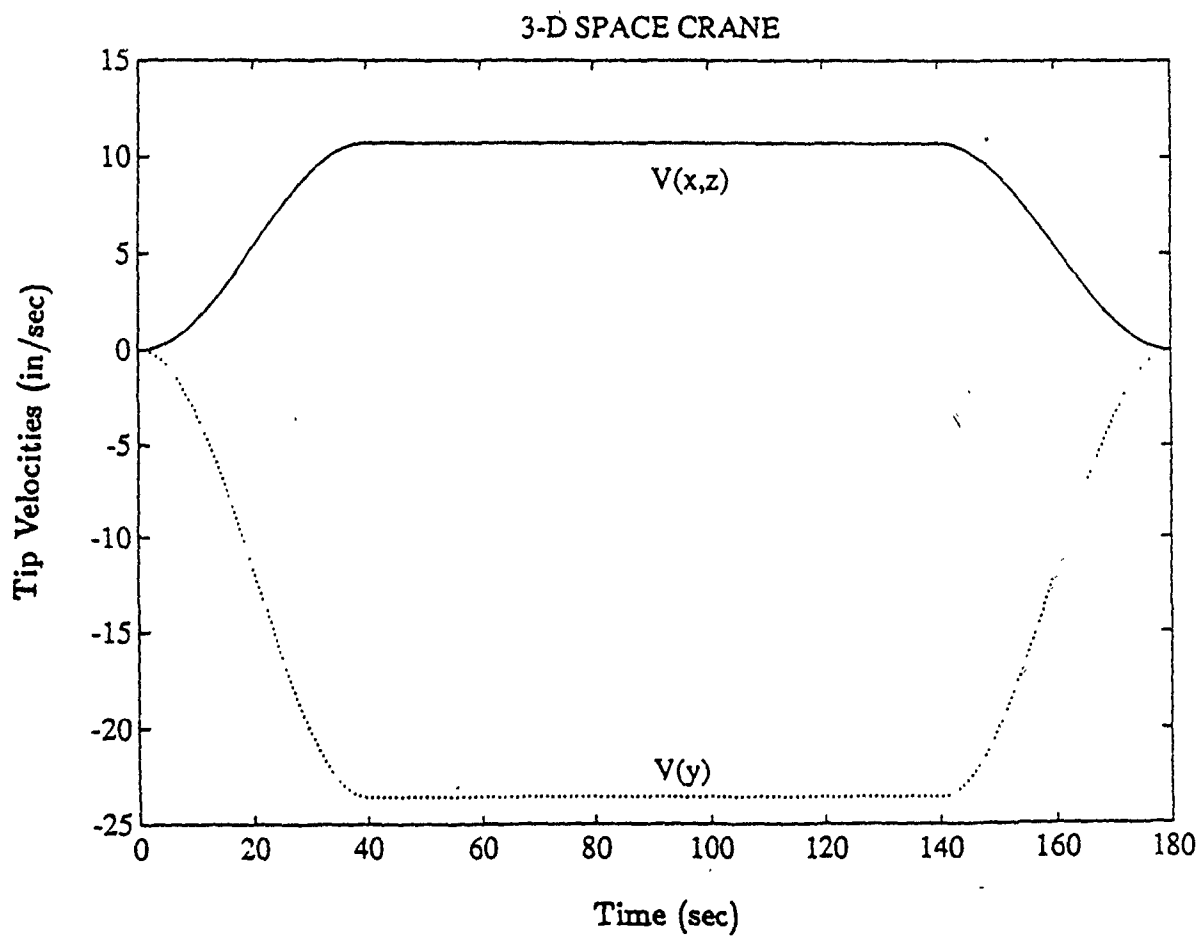


Fig. 10 Tip Velocities for the Rigid Space Crane

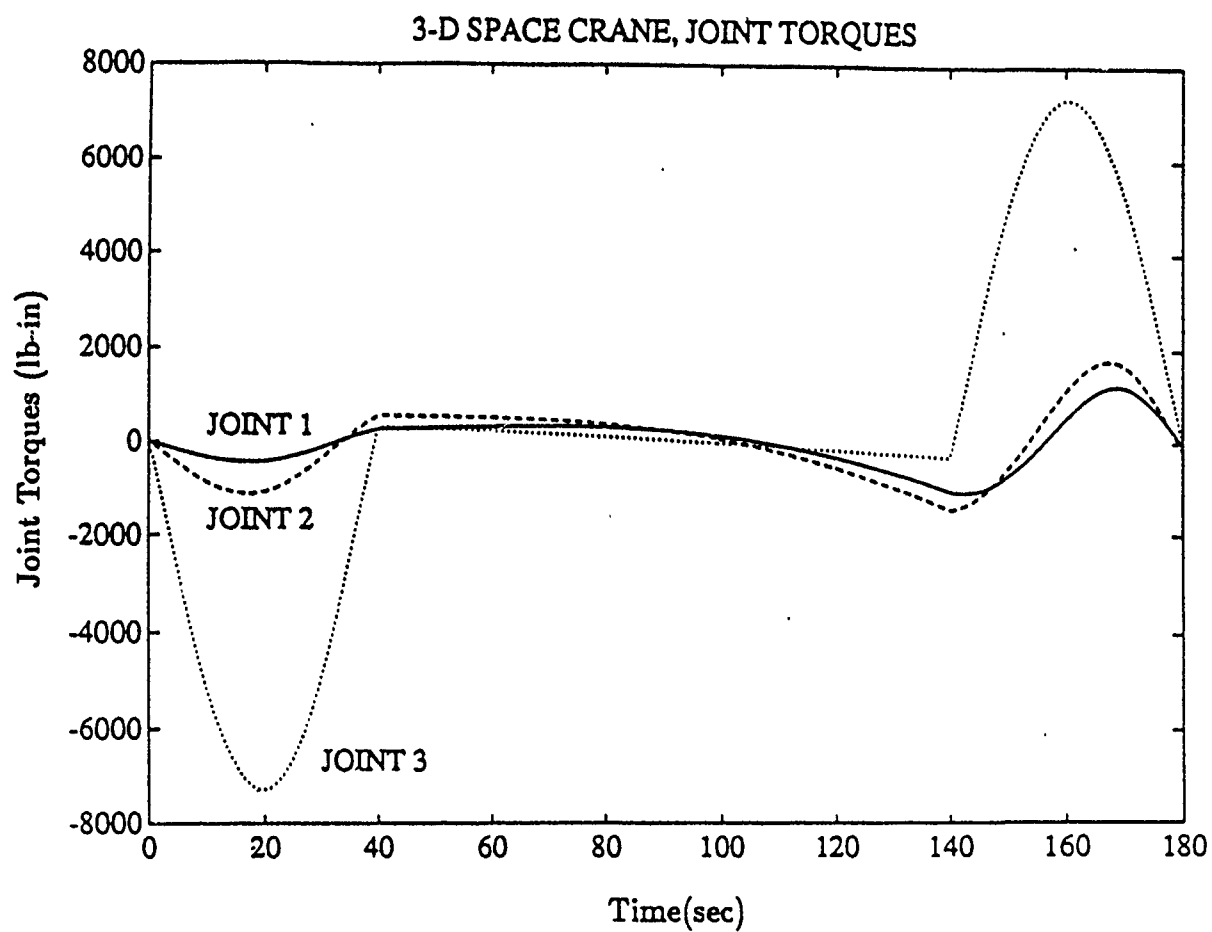


Fig. 11 Time Histories of Torques in the Connecting Joints

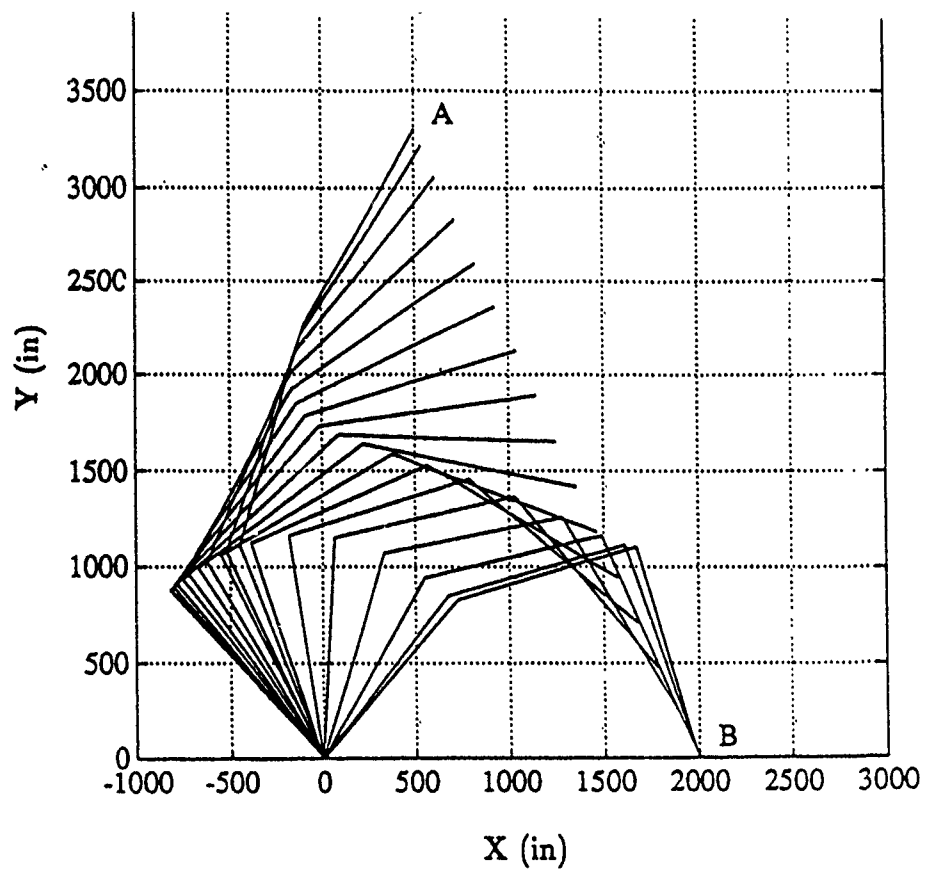


Fig. 12a Manipulator Configuration Along the X-Y Plane

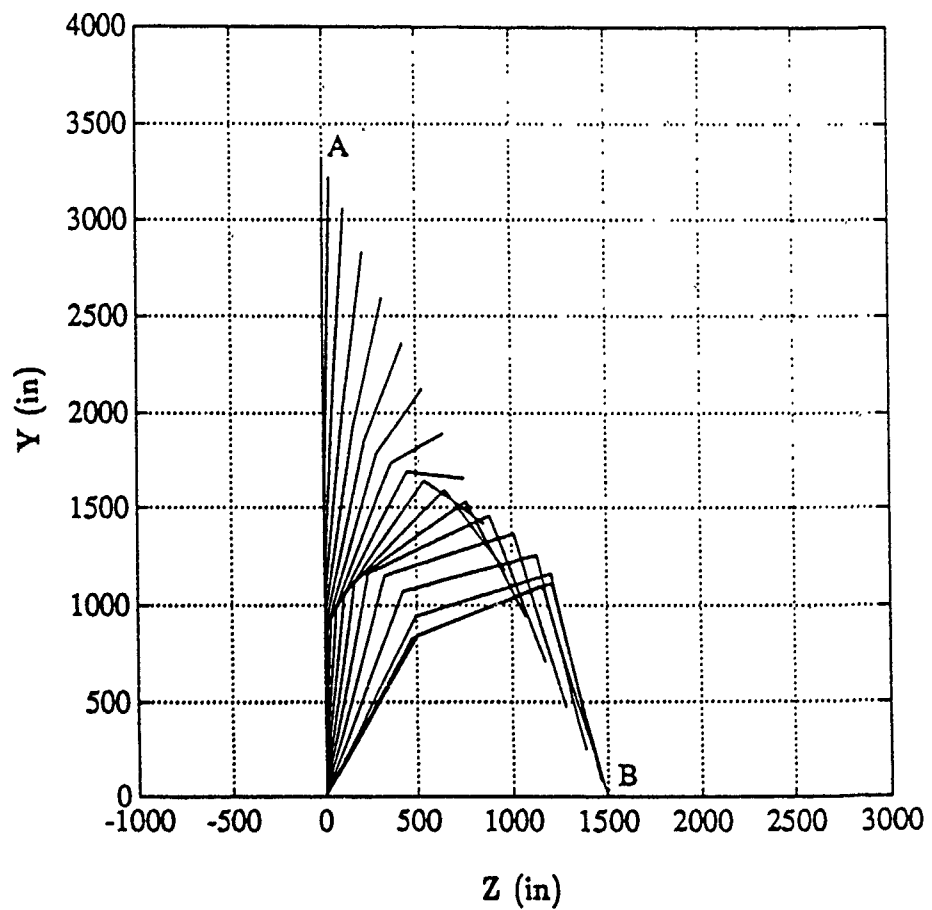
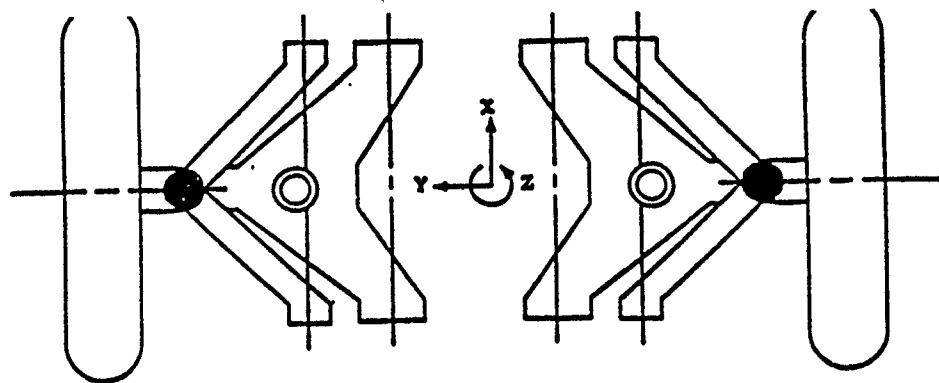
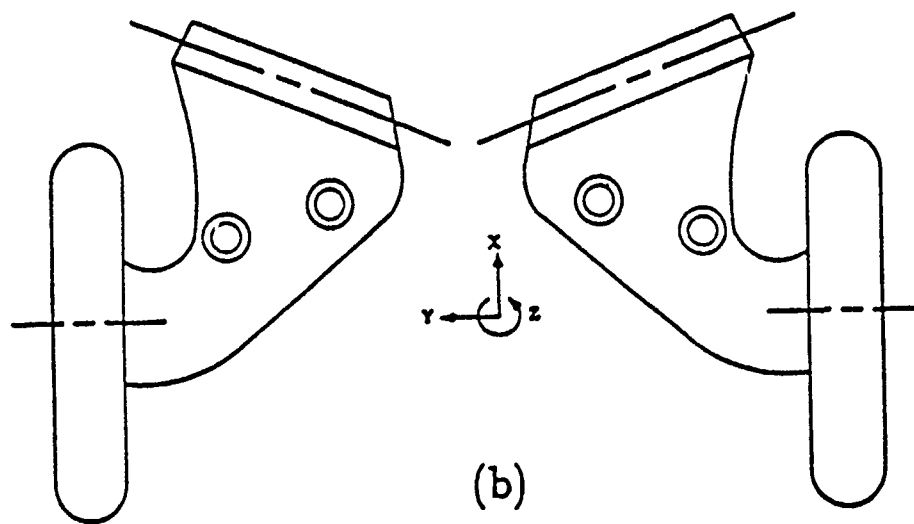
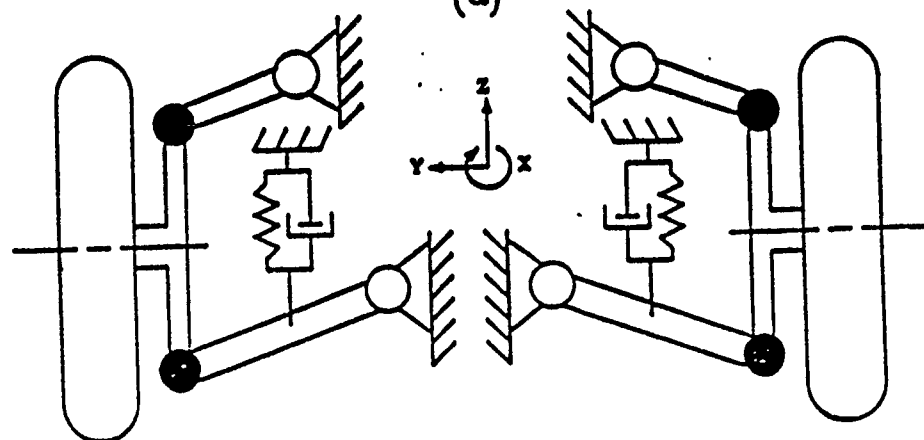


Fig. 12b Manipulator Configuration Along the Z-Y Plane



(a)



(b)

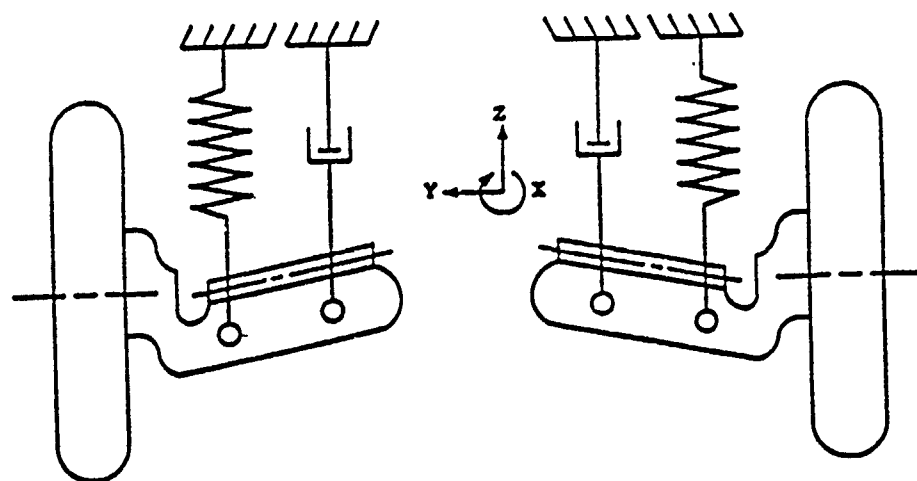


Fig. 13 Automobile Suspension Systems
 (a) Front Suspension Systems: Top and Read Views
 (b) Rear Suspension Systems: Top and Read Views

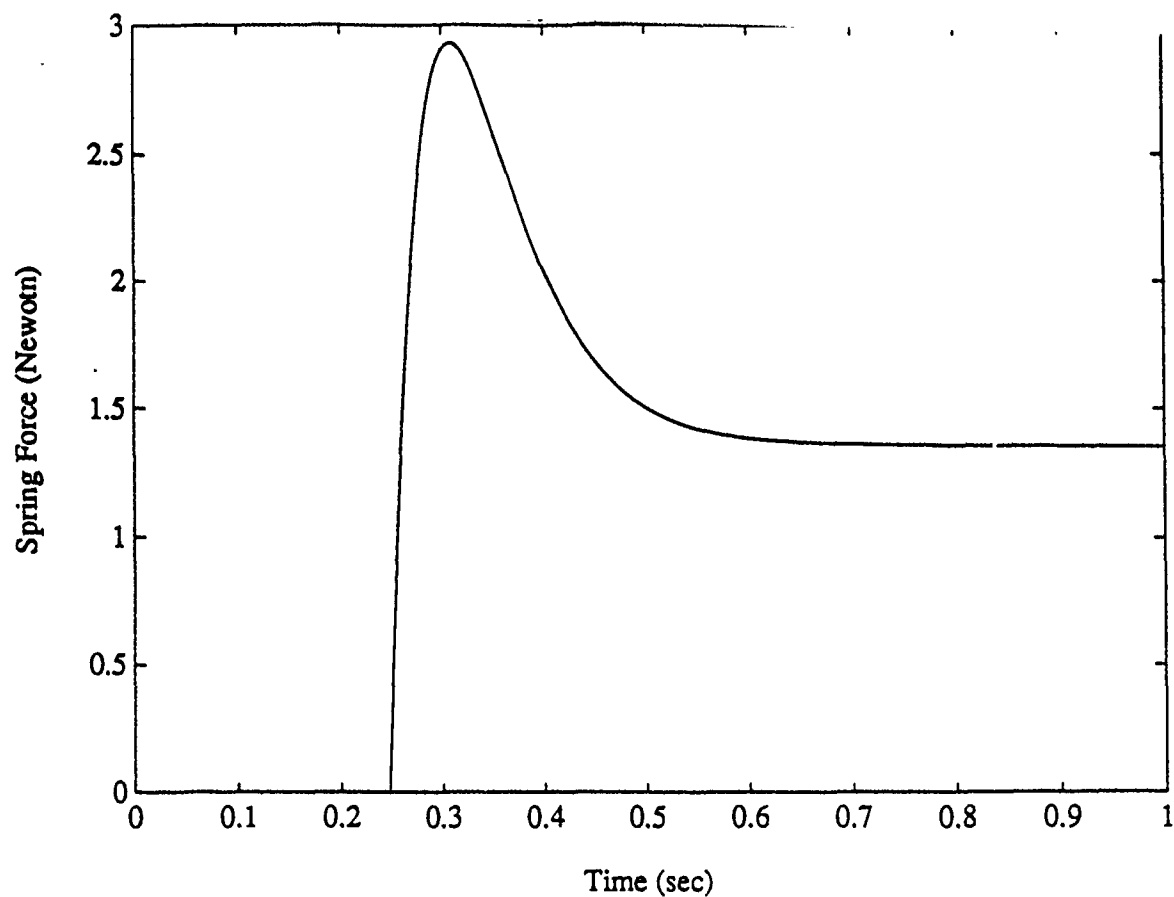


Fig. 14 Force Storage in Springs

NO. of Processors	CPU Time Consumed (sec) Per Time Step	Speed Up	Efficiency
1	7.612	1	1
2	4.133	1.842	0.921
3	2.828	2.692	0.897
4	2.287	3.328	0.832
5	1.901	4.004	0.801
6	1.708	4.476	0.746

Table 1 Performance of the Present Parallel Scheme Using Alliant FX/8

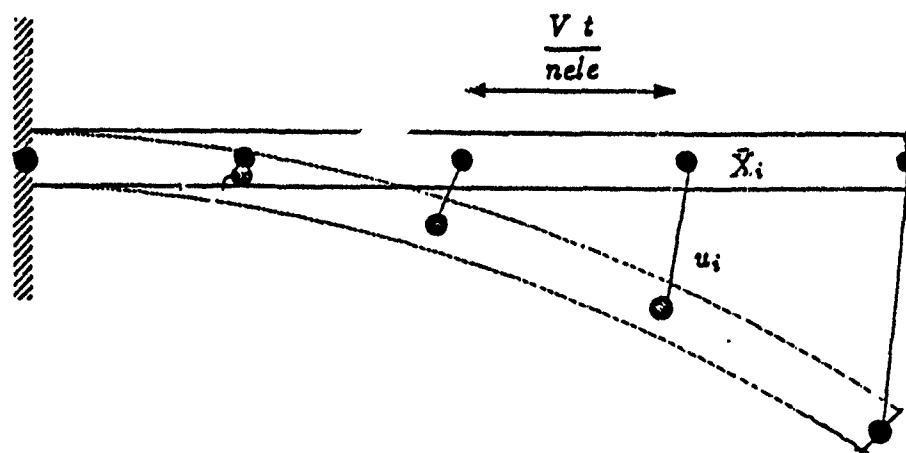


Fig. 15 Deployment Kinematics

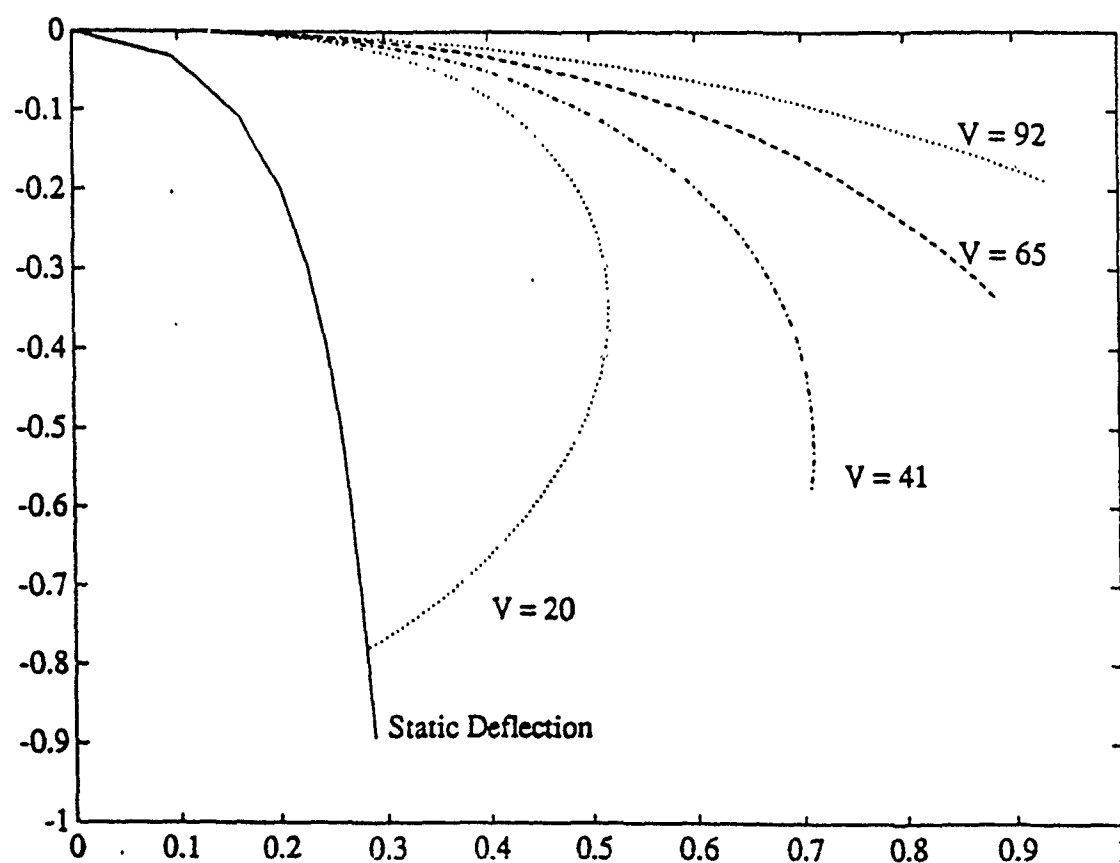


Fig. 16 End Orbits for Various Deployment Speeds (V)

The Core-Congruential Formulation of Geometrically Nonlinear TL Finite Elements

CARLOS A. FELIPPA

LUIS A. CRIVELLI

Department of Aerospace Engineering Sciences and
Center for Space Structures and Controls
University of Colorado
Boulder, Colorado 80309-0429, USA

January 1991

Report No. CU-CSSC-91-01

Invited chapter contributed to *Nonlinear Computational Mechanics —
The State of the Art*, edited by P. Wriggers and W. Wagner
Anniversary volume in honor of Professor E. Stein
to be published by Springer-Verlag, Berlin, 1991

Research supported by Air Force Office of Scientific
Research (AFOSR) under Grant F49620-87-C-0074.

The Core-Congruential Formulation of Geometrically Nonlinear TL Finite Elements

C. A. FELIPPA and L. A. CRIVELLI

Department of Aerospace Engineering Sciences
and Center for Space Structures and Controls
University of Colorado at Boulder
Boulder, CO 80309-0429, USA

Summary

The core-congruential formulation (CCF) of geometrically nonlinear finite elements based on the Total Lagrangian (TL) description is presented. Although the key ideas behind the CCF can be traced back to Rajasekaran and Murray in 1973, it has not subsequently received serious attention. The CCF is first described in general form and then applied to the derivation of geometrically nonlinear truss, plane stress, and Timoshenko beam elements using the Green-Lagrange strain measure. Several advantages of the CCF, notably the physically clean separation of material and geometric stiffnesses, and its independence with respect to the ultimate choice of shape functions and element degrees of freedom, are noted. Two problems involving very large motions solved with the beam element display the range of applicability of this formulation, which transcends the kinematic limitations commonly attributed to the TL description.

1 Introduction

There is an elegant Total Lagrangian (TL) formulation of geometrically nonlinear finite elements that has received little attention in the literature. This will be referred to as the *core-congruential formulation*, or CCF, in the sequel. The key concepts, presented by Rajasekaran and Murray in 1973 [1], evolved from the analysis and reinterpretation of the pioneer work of Mallet and Marcal [2], as well as Murray's previous work in geometrically nonlinear finite element analysis [3]. The discussion of [1] by Felippa [4] provided parametric expressions for the stiffness matrices that appear at various levels of the discrete governing equations.

In 1987 a course in nonlinear finite elements offered by the first author presented the derivation of several elements using the CCF. Preparation of homework assignments and feedback from students helped to streamline the material. Subsequently the doctoral thesis by Crivelli [5] made extensive use of the CCF in the systematic development of a three-dimensional nonlinear Timoshenko beam element capable of undergoing arbitrarily large rotations. This particular application pushed this formulation beyond frontiers

hitherto deemed unpassable by a TL element with rotational degrees of freedom.¹ Experience has shown that the beam formulation exhibits computational robustness lacking in other TL formulations that incorporate arbitrary kinematic approximations *ab initio*. The central idea of the CCF is the construction of TL stiffness matrices for geometrically nonlinear analysis through the scheme

$$\mathbf{K}^{\text{level}} = \int_V \mathbf{G}^T \mathbf{S}^{\text{level}} \mathbf{G} dV, \quad (1)$$

where $\mathbf{S}$ is the core stiffness matrix, $\mathbf{K}$ the physical stiffness in terms of node-displacement degrees of freedom $\mathbf{v}$, $\mathbf{G}$ a core-to-physical-freedom transformation matrix independent of $\mathbf{v}$, V the appropriate integration volume, and in which "level" identifies the *governing equation level at which the stiffness matrix is used*. The three levels of main interest in practice are: energy, force equilibrium, and first-order incremental equilibrium. The core stiffness matrix is expressed in terms of the *displacement gradients* at each material point. Displacement gradients $\mathbf{g}$ make a better choice of core variables than finite strains because for many element types they can be expressed linearly in terms of node displacements $\mathbf{v}$ as $\mathbf{g} = \mathbf{G}\mathbf{v}$, a property that validates (1) for all levels.²

The qualifier "core" emphasizes the *independence of $\mathbf{S}^{\text{level}}$ with respect to discretization decisions* such as element geometry, shape functions, and choice of nodal degrees of freedom. Such a dependence is introduced by the congruential transformation indicated in (1) and the integration over element volumes.

The CCF has several advantages noted later. But perhaps the most important one is the clean separation of physical effects. The importance of this factor should not be underestimated, because physical transparency is the key to success in nonlinear analysis.

As a final general remark, the specialization of (1) to linear analysis does *not* yield directly the well known $\mathbf{B}^T \mathbf{E} \mathbf{B}$ integrand of the standard formulation of linear stiffness matrices. This shows that such an alternative congruential form does not extend naturally to geometrically nonlinear analysis.³

¹ The conventional wisdom is that TL-based elements would utterly fail beyond moderate rotations, and an updated Lagrangian or corotational description is necessary for handling truly large motions. For example, recently Mathiasson *et.al.* [6] concluded that "The TL formulation can only be used in problems with small or moderate displacements."

² If $\mathbf{G}$ depends on $\mathbf{v}$, the transformation in (1) is more complex and varies with level, as discussed in Section 6.

³ Indeed, naive generalization of the $\mathbf{B}^T \mathbf{E} \mathbf{B}$ form to nonlinear analysis led several investigators to incorrect results during the 60's and 70's.

2 Historical Background

In 1968 Mallet and Marcal [2] presented a standard nomenclature for geometrically nonlinear finite element structural analysis based on the Total Lagrangian (TL) description. Consider a discrete, finite element model of a static structural system under dead loading with nodal displacement degrees of freedom collected in array $\mathbf{v}$. Displacements are measured from a fixed reference configuration C^0 to a current configuration C^t . The virtual-work conjugate forces, independent of $\mathbf{v}$, are collected in array $\mathbf{p}$. The system has a total potential energy function $J = U - W$ that is the difference between the strain energy U and the loads potential $W = \mathbf{p}^T \mathbf{v}$. The residual node forces are $\mathbf{r} = \partial J / \partial \mathbf{v}$, and the symbol Δ denotes increment associated with the variation of the current configuration.

Mallet and Marcal expressed the total potential energy, the residual (force-balance) equilibrium equations, and the incremental equilibrium equations as follows:

$$J = U - W = \frac{1}{2} \mathbf{v}^T [\mathbf{K}_0 + \frac{1}{3} \mathbf{N}_1 + \frac{1}{6} \mathbf{N}_2] \mathbf{v} - \mathbf{p}^T \mathbf{v}, \quad (2)$$

$$\mathbf{r} = \frac{\partial J}{\partial \mathbf{v}} = [\mathbf{K}_0 + \frac{1}{2} \mathbf{N}_1 + \frac{1}{3} \mathbf{N}_2] \mathbf{v} - \mathbf{p} = 0, \quad (3)$$

$$\Delta \mathbf{r} = [\mathbf{K}_0 + \mathbf{N}_1 + \mathbf{N}_2] \Delta \mathbf{v} - \Delta \mathbf{p} = 0. \quad (4)$$

Here $\mathbf{K}_0$ is the *linear stiffness matrix* evaluated at the reference configuration, whereas $\mathbf{N}_1$ and $\mathbf{N}_2$ are *nonlinear stiffness matrices*, also evaluated at the reference configuration, that depend linearly and quadratically, respectively, on the node displacements $\mathbf{v}$. The $\mathbf{N}$ matrices were said "to repeat" in the foregoing expressions.

Five years later Rajasekaran and Murray [1] examined more critically the structure of the matrices that appear in the above equations. In that investigation they chose to start from the "core" stiffness matrices corresponding to $\mathbf{K}$, $\mathbf{N}_1$ and $\mathbf{N}_2$ expressed in terms of displacement gradients, and in doing so laid down the main idea of the CCF. Working with specific elements they showed that the nonlinear stiffness matrices $\mathbf{N}_1$ and $\mathbf{N}_2$ are *not uniquely determined*. Indeed (2)–(4) as written are unique only for a single degree of freedom. They did not present, however, a general expression valid for arbitrary elements. This was partly done by Felippa [4], who in the discussion of [1] considered again those equations, rewritten here in a more general and compact form:

$$J = \frac{1}{2} \mathbf{v}^T \mathbf{K}^U \mathbf{v} + (\mathbf{p}^0 - \mathbf{p})^T \mathbf{v}, \quad (5)$$

$$\mathbf{r} = \mathbf{K}^r \mathbf{v} + \mathbf{p}^0 - \mathbf{p} = \mathbf{f} - \mathbf{p} = 0, \quad (6)$$

$$\Delta \mathbf{r} = \mathbf{K} \Delta \mathbf{v} - \Delta \mathbf{p} = 0, \quad (7)$$

in which the notation of this paper — rather than that of [4] — is used. Here $\mathbf{K}^U$,

K^r and K denote the *energy*, *secant* and *tangent* stiffness matrices, respectively⁴. In addition, p^0 is the *prestress force vector*, which vanishes if the reference configuration is stress free and was omitted in [4], and $f = K^r v + p^0$ is the internal force vector. The tangent stiffness is of course fundamental in incremental-iterative solution methods and stability analysis, while the secant stiffness (by itself or in the internal-force form $K^r v + p^0$) is important in pseudo-force methods. The energy stiffness enjoys limited application *per se* but has theoretical importance as source for the other two.

In linear problems $K^U = K^r = K = K_0$ and the three stiffness matrices coalesce. But in nonlinear problems not only do the matrices differ but, as shown in the next section, K^U and K^r may involve arbitrary scalar coefficients. Such parametrized expressions were given in [4] under the following restrictions:

- (R1) K^r is symmetric.
- (R2) The reference configuration is stress free.
- (R3) The finite strain measure is quadratic in the displacement gradients.
- (R4) The transformation between core and physical freedoms is linear.

The following treatment eliminates restrictions (R1) and (R2) altogether, and the other two selectively.

3 Core Stiffness Equations

A conservative, geometrically nonlinear structure under dead loading is viewed as a continuum undergoing finite displacements u . These displacements are measured from a fixed *reference* configuration C^0 to a variable *current* configuration C^t . *No discretization into finite elements is implied at this stage.* We confine our attention to the case in which the structure behavior stays within the linear elastic range, thus implying small deformational strains but arbitrarily large rotations. Corresponding points or *particles* in the reference and current configuration are referred to a fixed Cartesian coordinate system and have the coordinates X_i and x_i ($i = 1, \dots, n_d$), respectively, where n_d is the number of space dimensions. The displacement field components are $u_i = x_i - X_i$.

Let the state of strain at a particle in the current configuration be characterized by n_s strains e_i ($i = 1, 2, \dots, n_s$) collected in an array e , and let the corresponding conjugate stresses be s_i ($i = 1, 2, \dots, n_s$), collected in an array s . Using the summation convention the elastic stress-strain relations are written

$$s_i = s_i^0 + E_{ij} e_j, \quad \text{with} \quad E_{ij} = E_{ji}, \quad \text{or} \quad s = s^0 + Ee, \quad (8)$$

⁴ Energy and secant stiffnesses are not denoted by K^e and K^s because such symbols are used for other purposes in the finite element course noted in the Introduction.

where s_i^0 are stresses in the reference configuration (stresses that remain if $e_i = 0$, also called prestresses) and E_{ij} are elastic moduli arranged as a $n_s \times n_s$ square array in the usual manner.

Let $\mathcal{J}$, $\mathcal{U}$, $\mathcal{W}$, Ψ , Φ and Υ denote the analogues of J , U , W , p , f and r , respectively, at the particle level.⁵ The strain energy density can be expressed as

$$\mathcal{U} = e_i s_i^0 + \frac{1}{2} e_i E_{ij} e_j = \mathbf{e}^T \mathbf{s}^0 + \frac{1}{2} \mathbf{e}^T \mathbf{E} \mathbf{e}. \quad (9)$$

The total strain energy U is obtained by integrating (9) over the structure volume: $U = \int_V \mathcal{U} dV$; the integration taking place — as can be expected in a TL description — over the reference configuration geometry.

Next, introduce the n_g displacement gradients $g_{mn} = \partial u_m / \partial X_n$. These are subsequently identified as g_i ($i = 1, 2, \dots, n_g$) so they can be conveniently arranged in a one-dimensional array $\mathbf{g}$. Following [2], assume that the strains e_i are linked to the displacement gradients through matrix relations of the form

$$e_i = h_i^T \mathbf{g} + \frac{1}{2} \mathbf{g}^T \mathbf{H}_i \mathbf{g}, \quad i = 1, 2, \dots, n_s, \quad (10)$$

where h_i and $\mathbf{H}_i$ are arrays of dimension $n_g \times 1$ and $n_g \times n_g$, respectively, with $\mathbf{H}_i$ symmetric. In [2] and [4] it was assumed that $\mathbf{H}_i$ is independent of $\mathbf{g}$, which is the case for the Green-Lagrange strain measure. This restriction, labeled (R3) in Section 2, will be enforced below except in Section 5. As noted previously, for deriving core equations we regard the displacement gradients $\mathbf{g}$ as degrees of freedom. On substituting (8) and (10) into (9) we obtain the "core counterparts" of (5)–(7), in which $\mathbf{v}$ has become $\mathbf{g}$:

$$\mathcal{J} = \mathcal{U} - \mathcal{W} = \frac{1}{2} \mathbf{g}^T \mathbf{S}^U \mathbf{g} + (\Psi^0 - \Psi)^T \mathbf{g}, \quad (11)$$

$$\Upsilon = \frac{\partial \mathcal{J}}{\partial \mathbf{g}} = \mathbf{S}^r \mathbf{g} + \Psi^0 - \Psi = \Phi - \Psi = 0, \quad (12)$$

$$\Delta \Upsilon = \mathbf{S} \Delta \mathbf{g} - \Delta \Psi = 0. \quad (13)$$

Here $\mathbf{S}^U$, $\mathbf{S}^r$ and $\mathbf{S}$ denote the energy, secant and tangent core stiffness matrices, and Ψ^0 , which is independent of $\mathbf{g}$, is the core counterpart of p^0 . With this notation the first and second variations of the strain energy density can be expressed as

$$\delta \mathcal{U} = \delta \mathbf{g}^T (\mathbf{S}^U \mathbf{g} + \Psi^0) + \frac{1}{2} \mathbf{g}^T \delta \mathbf{S}^U \mathbf{g} = \delta \mathbf{g}^T (\mathbf{S}^r \mathbf{g} + \Psi^0) = \delta \mathbf{g}^T \Phi, \quad (14)$$

$$\delta^2 \mathcal{U} = \delta \mathbf{g}^T \mathbf{S}^r \delta \mathbf{g} + \delta \mathbf{g}^T \delta \mathbf{S}^r \mathbf{g} + (\delta^2 \mathbf{g})^T \Phi = \delta \mathbf{g}^T \mathbf{S} \delta \mathbf{g}. \quad (15)$$

⁵ The first three acquire the meaning of energy densities, whereas Ψ is a dead-loading applied force density independent of $\mathbf{u}$.

These variational equations implicitly determine S^r , Φ and S from S^U and Ψ^0 . If restriction (R4) holds, the term $\delta^2 g \Phi$ drops out.⁶ For convenience introduce the following $n_g \times n_g$ matrices (with summation convention implied):

$$\begin{aligned} S_0 &= E_{ij} h_i h_j, & S_1 &= E_{ij} h_i g^T H_j, & S_1^* &= E_{ij} (h_i^T g) H_j, \\ S_2 &= E_{ij} H_i g g^T H_j, & S_2^* &= E_{ij} (g^T H_i g) H_j, \end{aligned} \quad (16)$$

where parentheses are used to emphasize the grouping of *scalar* quantities such as $g^T H_i g$. It may be then verified⁷ that, if assumptions (R3)-(R4) of Section 2 hold, the core stiffnesses and prestress vector in (12)-(14) possess the general form:

$$\begin{aligned} S^U(\alpha, \beta) &= S_0 + \frac{1}{2}\alpha(S_1 + S_1^T) + (1 - \alpha)S_1^* + \frac{1}{4}\beta S_2 + \frac{1}{4}(1 - \beta)S_2^* + s_i^0 H_i, \\ S^r(\phi, \psi) &= S_0 + \frac{1}{2}S_1 + \phi S_1^T + (1 - \phi)S_1^* + \frac{1}{4}(2 - \psi)S_2 + \frac{1}{4}\psi S_2^* + s_i^0 H_i, \\ S &= S_0 + S_1 + S_1^T + S_1^* + S_2 + \frac{1}{2}S_2^* + s_i^0 H_i, & \Psi^0 &= s_i^0 h_i. \end{aligned} \quad (17)$$

Here α , β , ϕ and ψ are arbitrary scalar coefficients in the sense that $g^T S^U g$ and $S^r g$ are independent of them. In fact,

$$\Phi = S^r g + \Psi^0 = s_i b_i \quad (18)$$

where b_i is defined in (20). The expressions are more general than those given in [4] because restrictions (R1)-(R2) of Section 2 are no longer assumed. Note that S^r becomes symmetric if $\phi = 1/2$.

The "repeatable forms" of equations (2)-(4) are obtained if $\alpha = \beta = \psi = 2/3$ and $\phi = 1/2$, in which case the combinations $S_1 + S_1^T + S_1^*$ and $S_2 + \frac{1}{2}S_2^*$ become the core counterparts of N_1 and N_2 , respectively. But this observation has largely historical interest. More physically relevant are the following combinations:

$$\begin{aligned} S_D &= S_1 + S_1^T + S_2, & S_M &= S_0 + S_D, \\ S_G &= S_1^* + \frac{1}{2}S_2^* + s_i^0 H_i = s_i H_i. \end{aligned} \quad (19)$$

These are the core versions of the *initial-displacement*, *material* and *geometric* stiffness, respectively. The core tangent stiffness is $S = S_0 + S_D + S_G = S_M + S_G$.

⁶ If $g = Gv$ with G independent of v , $\delta^2 g = G \delta^2 v = 0$ because v are independent variables. On the other hand, if displacement gradients are nonlinear functions of node displacements expressible as $g_i = g_i(v_j)$, then

$$\delta g_i = \frac{\partial g_i}{\partial v_j} \delta v_j = G_{ij} \delta v_j, \quad \delta^2 g_i = \frac{\partial^2 g_i}{\partial v_j \partial v_k} \delta v_j \delta v_k + \frac{\partial g_i}{\partial v_j} \delta^2 v_j = F_{ijk} \delta v_j \delta v_k.$$

Thus δg is still $G \delta v$ but $\delta^2 g = (F \delta v) \delta v$, where F is a cubic array.

⁷ The verification was done by hand for [4], and rechecked with the MACSYMA symbolic algebra computer system 14 years later.

4 Spectral Expression of Core Stiffnesses

There is an alternative, more compact expression of the core stiffnesses, not presented in [4], that offers theoretical as well as implementational advantages at the cost of some generality. Define vectors b_i and c_i as

$$e_i = c_i^T g, \quad c_i = h_i + \frac{1}{2} H_i g, \quad b_i = \frac{\partial e_i}{\partial g} = h_i + H_i g. \quad (20)$$

Then the spectral forms⁸ are

$$S^U(1, 1) = S^U|_{\alpha=\beta=1} = E_{ij} c_i c_j^T + s_i^0 H_i, \quad (21)$$

$$S^r(0, 0) = S^r|_{\phi=\psi=0} = E_{ij} b_i b_j^T + s_i^0 H_i, \quad (22)$$

$$S^r(\frac{1}{2}, 1) = S^r|_{\phi=\frac{1}{2}, \psi=1} = E_{ij} c_i c_j^T + \frac{1}{2}(s_i + s_i^0) H_i, \quad (23)$$

$$S = E_{ij} b_i b_j^T + s_i H_i = S_M + S_G. \quad (24)$$

Note that $S^r(\frac{1}{2}, 1)$ is symmetric but $S^r(0, 0)$ is not. It is seen that for energy and secant stiffnesses, compactness is paid in terms of settling for specific coefficient values. The foregoing relations may be obtained by repeated differentiation of $\mathcal{U}$ with respect to g . A lesson from this observation is that direct differentiation of nonlinear matrix relations may conceal the richness of possible expressions as revealed in equations (17).

5 Generalization to $H(g)$

An advantage of the spectral forms is that the complication introduced by H_i being functions of g (as it happens for all nonlinear strain measures except Green-Lagrange's) can be accommodated by redefining some of the previous relations as follows:

$$\begin{aligned} b_i^* &= \frac{\partial e_i}{\partial g} = h_i + H_i^* g, \quad H_i^* = H_i + \frac{1}{2} g^T \frac{\partial H_i}{\partial g}, \\ H_i^{**} &= \frac{\partial^2 e_i}{\partial g \partial g} = \frac{\partial b_i}{\partial g} = H_i + \frac{\partial \left(\frac{1}{2} g^T \frac{\partial H_i}{\partial g} g \right)}{\partial g}, \end{aligned} \quad (25)$$

$$S^r(0, 0) = E_{ij} b_i (c_j^*)^T + s_i^0 H_i^*, \quad S = E_{ij} b_i^* (b_j^*)^T + s_i H_i^{**}.$$

There is no simple form of $S^r(\frac{1}{2}, 1)$ equivalent to (23), and indeed symmetric forms of S^r do not generally exist if strain measures other than Green-Lagrange's are used.

⁸ So called because of the formal similarity of equations (21)–(24) with the spectral decomposition of a matrix as the sum of rank-one matrices.

6 Transformation to Physical Coordinates

Subdivide the continuum into finite elements. Over each individual element⁹ the displacement field $\mathbf{u}^T = (u_1, u_2, u_3)$ is given by $\mathbf{u} = \mathbf{P}\mathbf{v}$, where $\mathbf{v}$ now collects the element node-displacement degrees of freedom and $\mathbf{P}$ is a matrix of shape functions. If the displacement gradients are linear in $\mathbf{v}$, differentiation yields $\mathbf{g} = \mathbf{G}\mathbf{v}$, $\delta\mathbf{g} = \mathbf{G}\delta\mathbf{v}$ and $\delta^2\mathbf{g} = 0$ (cf. footnote 6). Invariance of the strain energy variations δU and $\delta^2 U$ obtained by integrating (14) and (15) over V yields

$$\mathbf{K}^U = \int_V \mathbf{G}^T \mathbf{S}^U \mathbf{G} dV, \quad \mathbf{K}^r = \int_V \mathbf{G}^T \mathbf{S}^r \mathbf{G} dV, \quad \mathbf{K} = \int_V \mathbf{G}^T \mathbf{S} \mathbf{G} dV, \quad \mathbf{p} = \int_V \mathbf{G}^T \Psi dV, \quad (26)$$

where V stands now for the *element* volume, and the last transformation also applies to Ψ^0 . Although the dependency of $\mathbf{S}^{level}$ and Ψ on $\mathbf{g}$ is not made implicit in these equations, it must be remembered that the transformation $\mathbf{g} = \mathbf{G}\mathbf{v}$ also appears there. Because of the ensuing algebraic complexity, numerical integration is generally required unless the gradients are constant over the element.

Often $\mathbf{G}$ is expressed as a chain of transformations, some of which are position dependent and remain inside the integral whereas others are not and may be taken outside. For example, in the bar element studied below, $\mathbf{G} = \mathbf{T}\mathbf{D}$, where $\mathbf{D}$ transforms $\mathbf{g}$ to local node displacements and $\mathbf{T}$ transforms local to global node displacements.

If the relation between $\mathbf{g}$ and $\mathbf{v}$ is nonlinear so that $\mathbf{g} = \mathbf{g}(\mathbf{v})$, then $\mathbf{G} = \partial\mathbf{g}/\partial\mathbf{v}$ depends on $\mathbf{v}$, and transformation rules change with level. This complication occurs in elements with rotational freedoms, such as beams, plates and shells, if finite rotations are exactly treated. Both residual and incremental levels are affected. At the residual level, $\mathbf{K}^r$ cannot be extracted, except artificially, and it is better to work with the internal force vector $\mathbf{f} = \int_V \mathbf{G}^T \Phi dV = \int_V \mathbf{G}^T s_i b_i dV$ directly. At the incremental level, the term $(\delta^2\mathbf{g})^T \Phi$ in (15) must be retained because $\delta^2\mathbf{g}$ does not vanish. Invariance of $\delta^2 U$ yields

$$\mathbf{K} = \int_V (\mathbf{G}^T \mathbf{S} \mathbf{G} + \mathbf{Q}) dV, \quad (27)$$

where $\mathbf{S}$ is that given in (17), and the entries of the symmetric matrix $\mathbf{Q}$ are, from footnote 6 and equation (18):

$$Q_{jk} = Q_{kj} = F_{mjk} \Phi_m = \frac{\partial^2 g_m}{\partial v_j \partial v_k} \Phi_m = \frac{\partial^2 g_m}{\partial v_j \partial v_k} s_i b_{im}, \quad (28)$$

in which b_{im} denotes the m^{th} entry of $\mathbf{b}_i$. It is seen that $\mathbf{Q}$ vanishes in a stress-free state and can be therefore viewed as a correction to the geometric stiffness matrix.

⁹ For simplicity elements will not be identified by an element index in the sequel.

7 Example 1: 3D Bar Element ($n_d = 3$, $n_s = 1$, $n_g = 3$)

The first CCF application example is to a two-node prismatic, straight bar element of reference length L and cross section area A in 3D space. To simplify node sub-scripting notations, Cartesian reference systems and displacement components will be denoted by (X, Y, Z) , (x, y, z) and (u_X, u_Y, u_Z) rather than (X_1, X_2, X_3) , (x_1, x_2, x_3) and (u_1, u_2, u_3) , respectively. The bar node displacements are (v_{Xn}, v_{Yn}, v_{Zn}) , $n = 1, 2$.

In the reference configuration C^0 the bar element is referred to a *local* Cartesian system $(\bar{X}, \bar{Y}, \bar{Z})$, with $\bar{X}$ located along the bar axis, so that the element extends from node 1 at $(0, 0, 0)$ to node 2 at $(L, 0, 0)$. The current configuration is also straight and extends from $(\bar{x}_1 = \bar{v}_{X1}, \bar{y}_1 = \bar{v}_{Y1}, \bar{z}_1 = \bar{v}_{Z1})$ to $(\bar{x}_2 = L + \bar{v}_{X2}, \bar{y}_2 = \bar{v}_{Y2}, \bar{z}_2 = \bar{v}_{Z2})$. The element displacement field in local coordinates may be expressed as

$$\bar{\mathbf{u}} = \begin{Bmatrix} \bar{u}_X \\ \bar{u}_Y \\ \bar{u}_Z \end{Bmatrix} = \begin{bmatrix} 1-\zeta & 0 & 0 & \zeta & 0 & 0 \\ 0 & 1-\zeta & 0 & 0 & \zeta & 0 \\ 0 & 0 & 1-\zeta & 0 & 0 & \zeta \end{bmatrix} \begin{Bmatrix} \bar{v}_{X1} \\ \bar{v}_{Y1} \\ \bar{v}_{Z1} \\ \bar{v}_{X2} \\ \bar{v}_{Y2} \\ \bar{v}_{Z2} \end{Bmatrix} = \mathbf{P}\bar{\mathbf{v}}, \quad (29)$$

where $\zeta = \bar{X}/L$ is a natural axial coordinate varying from 0 to 1. The three displacement gradients that intervene in the definition of nonlinear strains are

$$\mathbf{g} = \begin{Bmatrix} g_1 \\ g_2 \\ g_3 \end{Bmatrix} = \begin{Bmatrix} \partial \bar{u}_X / \partial \bar{X} \\ \partial \bar{u}_Y / \partial \bar{X} \\ \partial \bar{u}_Z / \partial \bar{X} \end{Bmatrix} = \frac{1}{L} \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{v}} = \frac{1}{L} \mathbf{D}\bar{\mathbf{v}} \quad (30)$$

As strain measure we adopt the Green-Lagrange axial strain, defined as

$$\begin{aligned} e \equiv e_1 &= \frac{\partial \bar{u}_X}{\partial \bar{X}} + \frac{1}{2} \left[\left(\frac{\partial \bar{u}_X}{\partial \bar{X}} \right)^2 + \left(\frac{\partial \bar{u}_Y}{\partial \bar{X}} \right)^2 + \left(\frac{\partial \bar{u}_Z}{\partial \bar{X}} \right)^2 \right] = g_1 + \frac{1}{2}(g_1^2 + g_2^2 + g_3^2) \\ &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}^T \begin{Bmatrix} g_1 \\ g_2 \\ g_3 \end{Bmatrix} + \frac{1}{2} \begin{Bmatrix} g_1 \\ g_2 \\ g_3 \end{Bmatrix}^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} g_1 \\ g_2 \\ g_3 \end{Bmatrix} = \mathbf{h}^T \mathbf{g} + \frac{1}{2} \mathbf{g}^T \mathbf{H} \mathbf{g}. \end{aligned} \quad (31)$$

Thus for this choice of strain, $\mathbf{h}^T = [1 \ 0 \ 0]$ and $\mathbf{H}$ is the 3×3 identity matrix. The conjugate stress measure $s_1 \equiv s$ is the second Piola-Kirchhoff (PK2) axial stress. The stress-strain relation is $s = s^0 + Ee$, where s^0 and s are PK2 axial stresses in the reference and current configurations, respectively, and E is Young's modulus.

Since $\mathbf{H}$ is independent of $\mathbf{g}$, to form the core stiffnesses in local coordinates we can use the spectral expressions of Section 4. First construct the vectors

$$\mathbf{c} \equiv \mathbf{c}_1 = \begin{Bmatrix} 1 + \frac{1}{2}g_1 \\ \frac{1}{2}g_2 \\ \frac{1}{2}g_3 \end{Bmatrix}, \quad \mathbf{b} \equiv \mathbf{b}_1 = \begin{Bmatrix} 1 + g_1 \\ g_2 \\ g_3 \end{Bmatrix}, \quad (32)$$

which inserted into (21), (23) and (24) yield

$$\mathbf{S}^U(1,1) = E\mathbf{c}\mathbf{c}^T + s^0\mathbf{H} = E \begin{bmatrix} (1 + \frac{1}{2}g_1)^2 & \frac{1}{2}g_2(1 + \frac{1}{2}g_1) & \frac{1}{2}g_3(1 + \frac{1}{2}g_1) \\ & \frac{1}{4}g_2^2 & \frac{1}{4}g_2g_3 \\ \text{symm} & & \frac{1}{4}g_3^2 \end{bmatrix} + s^0 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (33)$$

$$\mathbf{S}^r(\frac{1}{2},1) = E\mathbf{c}\mathbf{c}^T + s^m\mathbf{H} = E \begin{bmatrix} (1 + \frac{1}{2}g_1)^2 & \frac{1}{2}g_2(1 + \frac{1}{2}g_1) & \frac{1}{2}g_3(1 + \frac{1}{2}g_1) \\ & \frac{1}{4}g_2^2 & \frac{1}{4}g_2g_3 \\ \text{symm} & & \frac{1}{4}g_3^2 \end{bmatrix} + s^m \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (34)$$

$$\mathbf{S} = E\mathbf{b}\mathbf{b}^T + s\mathbf{H} = E \begin{bmatrix} (1 + g_1)^2 & g_2(1 + g_1) & g_3(1 + g_1) \\ & g_2^2 & g_2g_3 \\ \text{symm} & & g_3^2 \end{bmatrix} + s \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (35)$$

where $s^m = \frac{1}{2}(s^0 + s)$. The clean separation into material and geometric (initial-stress) stiffnesses should be noted. As (30) is linear in $\mathbf{v}$, the physical stiffnesses in local coordinates are obtained through (26) in which $V = AL$. For the tangent stiffness this transformation yields

$$\bar{\mathbf{K}} = \frac{1}{L^2} \int_V \mathbf{D}^T \mathbf{S} \mathbf{D} dV = \frac{A}{L} \mathbf{D}^T (E\mathbf{b}\mathbf{b}^T + s\mathbf{H}) \mathbf{D}, \quad (36)$$

with similar expressions for the other local stiffnesses. Given the simplicity of $\mathbf{D}$, $\bar{\mathbf{K}}$ could easily be expressed in closed form, and similarly for $\bar{\mathbf{K}}^U$ and $\bar{\mathbf{K}}^r$. Finally, transformation to displacements u_X, u_Y, u_Z in the global Cartesian system is handled in the usual manner. If $\bar{\mathbf{v}} = \mathbf{T}\mathbf{v}$, the global stiffness matrix is given by

$$\mathbf{K} = \frac{A}{L} \begin{bmatrix} \mathbf{T}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{T}^T \end{bmatrix} \mathbf{D}^T (E\mathbf{b}\mathbf{b}^T + s\mathbf{H}) \mathbf{D} \begin{bmatrix} \mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{T} \end{bmatrix}. \quad (37)$$

For efficient computer implementation, a closed form for $\mathbf{K}$ is easily obtained from (37) by symbolic manipulation and automatically converted to program statements.

8 Example 2: N -Node Plane Stress Element ($n_d = 2, n_s = 3, n_g = 4$)

As second example we consider an arbitrary N -node isoparametric element that models a plane stress problem. The Cartesian reference system and displacement components will be denoted by (X, Y) , (x, y) and (u_X, u_Y) rather than (X_1, X_2) , (x_1, x_2) and (u_1, u_2) , respectively. The element nodes are located at X_n, Y_n , ($n = 1, \dots, N$) in the reference configuration and move to $(x_n = X_n + u_{Xn}, y_n = Y_n + u_{Yn})$, ($n = 1, \dots, N$) in the current configuration.

The element displacement field is expressed as

$$\begin{Bmatrix} u_X \\ u_Y \end{Bmatrix} = \begin{bmatrix} P_1 & 0 & P_2 & \dots & 0 \\ 0 & P_1 & 0 & \dots & P_N \end{bmatrix} \begin{Bmatrix} v_{X1} \\ v_{Y1} \\ v_{X2} \\ \vdots \\ v_{YN} \end{Bmatrix} = \mathbf{P}\mathbf{v}, \quad (38)$$

where $P_n(\xi, \eta)$ are the appropriate isoparametric shape functions. In this case four displacement gradients appear, and these are arranged as

$$\mathbf{g} = \begin{Bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{Bmatrix} = \begin{Bmatrix} \partial u_X / \partial X \\ \partial u_Y / \partial X \\ \partial u_X / \partial Y \\ \partial u_Y / \partial Y \end{Bmatrix} = \mathbf{G}\mathbf{v}, \quad (39)$$

where $\mathbf{G}$ is a sparse $4 \times 2N$ matrix that contains the shape function derivatives $\partial P_n / \partial X$ and $\partial P_n / \partial Y$. The strain measures chosen are the three components e_i ($i = 1, 2, 3$) of the Green-Lagrange strains defined in the usual manner:

$$e_1 = e_{XX} = g_1 + \frac{1}{2}(g_1^2 + g_2^2) = \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix}^T \mathbf{g} + \frac{1}{2} \mathbf{g}^T \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{g}, \quad (40)$$

$$e_2 = e_{YY} = g_4 + \frac{1}{2}(g_3^2 + g_4^2) = \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix}^T \mathbf{g} + \frac{1}{2} \mathbf{g}^T \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{g}, \quad (41)$$

$$e_3 = e_{XY} + e_{YX} = g_2 + g_3 + g_1 g_3 + g_2 g_4 = \begin{Bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{Bmatrix}^T \mathbf{g} + \frac{1}{2} \mathbf{g}^T \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{g}, \quad (42)$$

from which expressions for $\mathbf{h}_i$ and $\mathbf{H}_i$ ($i = 1, 2, 3$) follow. For brevity, only the derivation of the tangent stiffness matrix will be described. Begin by forming the vectors

$$\mathbf{b}_1 = \begin{Bmatrix} 1 + g_1 \\ g_2 \\ 0 \\ 0 \end{Bmatrix}, \quad \mathbf{b}_2 = \begin{Bmatrix} 0 \\ 0 \\ g_3 \\ 1 + g_4 \end{Bmatrix}, \quad \mathbf{b}_3 = \begin{Bmatrix} g_3 \\ 1 + g_4 \\ 1 + g_1 \\ g_2 \end{Bmatrix}. \quad (43)$$

Then from (24) we get the core stiffness

$$\mathbf{S} = E_{ij} \mathbf{b}_i \mathbf{b}_j^T + s_i \mathbf{H}_i = \mathbf{S}_M + \mathbf{S}_G, \quad (44)$$

where $s_i = s_i^0 + E_{ij} e_j$, ($i, j = 1, 2, 3$), are the PK2 stresses in the current configuration.

In full and denoting $a_1 = 1 + g_1$ and $a_4 = 1 + g_4$:

$$S_M = \begin{bmatrix} E_{11}a_1^2 + E_{13}a_1g_3 + E_{33}g_3^2 & E_{11}a_1g_2 + E_{13}(a_1a_4 + g_2g_3) + E_{33}a_4g_3 & & \\ & E_{11}g_2^2 + E_{13}a_4g_2 + E_{33}a_4^2 & & \\ & & \text{symm} & \\ & & & E_{12}a_1g_3 + E_{13}a_1^2 + E_{23}g_3^2 + E_{33}a_1g_3 & E_{12}a_1a_4 + E_{13}a_1g_2 + E_{23}a_4g_3 + E_{33}g_2g_3 \\ & & & E_{12}g_2g_3 + E_{13}a_1g_2 + E_{23}a_4g_3 + E_{33}a_1a_4 & E_{12}a_4g_2 + E_{13}g_2^2 + E_{23}a_4^2 + E_{33}a_4g_2 \\ & & & E_{22}g_3^2 + E_{23}a_1g_3 + E_{33}a_1^2 & E_{22}a_4g_3 + E_{23}(a_1a_4 + 2g_2g_3) + E_{33}a_1g_2 \\ & & & & E_{22}a_4^2 + E_{23}a_4g_2 + E_{33}g_2^2 \end{bmatrix} \quad (45)$$

$$S_G = \begin{bmatrix} s_1 & 0 & s_3 & 0 \\ & s_1 & 0 & s_3 \\ & & s_2 & 0 \\ \text{symm} & & & s_2 \end{bmatrix} \quad (46)$$

The physical tangent stiffness is

$$K = \int_V G^T (S_M + S_G) G dV, \quad (47)$$

which is conveniently evaluated by numerical integration. Since several integrand matrices are quite sparse, for efficiency in the computer implementation the integrand should be evaluated symbolically and converted to program statements before being encapsulated in the Gauss quadrature loop.

9 Example 3: 3D Timoshenko Beam ($n_d = 3$, $n_s = 3$, $n_g = 9$)

As final CCF application we describe the derivation of a new TL Timoshenko beam element capable of undergoing arbitrarily large rotations in three dimensions. Only a brief outline of the derivation is given; full details may be found in Crivelli's thesis [5].

The beam kinematics is based on two assumptions: (1) the TL description, and (2) the Timoshenko beam hypothesis: planes sections remain plane although not necessarily normal to the deformed longitudinal axis. The beam is assumed isotropically elastic, with Young's modulus E and shear modulus $G = \frac{1}{2}E/(1 + \nu)$. The reference configuration of the beam is straight, prismatic and stress free, with cross section A and length L . A local reference frame $\mathbf{n}_i$ is defined on it, with $\mathbf{n}_1$ directed along the longitudinal axis (locus of cross section centroids). Vectors $\mathbf{n}_2$ and $\mathbf{n}_3$ are in the plane of the left-end cross section; these are eventually aligned with principal inertia axes to simplify some expressions. Along these vectors we attach the reference coordinate system (X_1, X_2, X_3) . We further define a set of *moving* frames, denoted by $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$, parametrized by the longitudinal coordinate X_1 . These frames displace rigidly attached to the beam cross sections as shown in Figure 1.

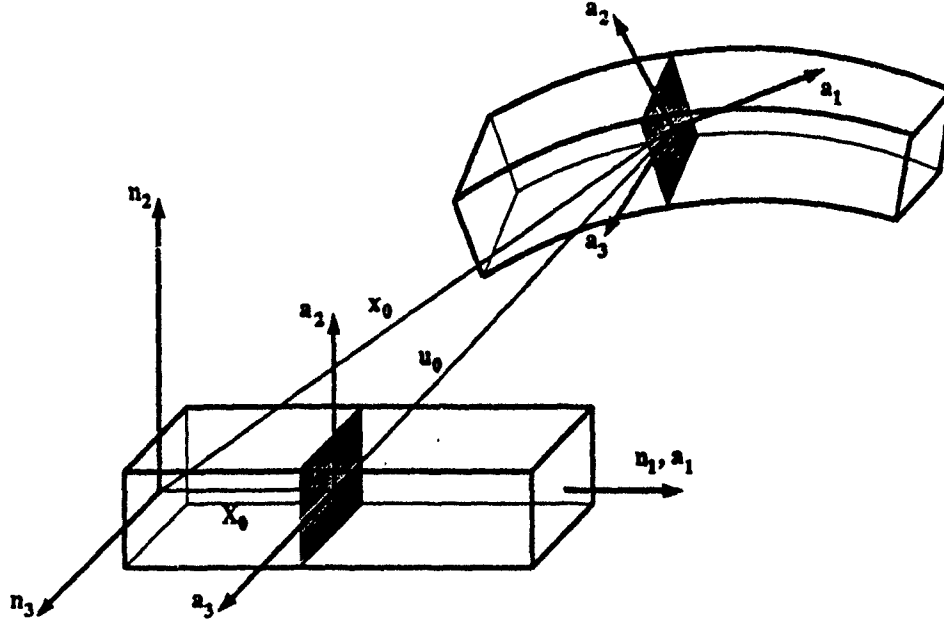


Fig. 1: Reference frames describing beam kinematics.

A beam particle originally at (X_1, X_2, X_3) displaces to

$$\mathbf{x}(\mathbf{X}) = \mathbf{x}_0(X_1) + \mathbf{R}^T(X_1) \boldsymbol{\zeta}(X_2, X_3) \quad (48)$$

where $\mathbf{x}_0$ is the position of the cross section centroid, $\mathbf{R}$ is a 3-by-3 orthogonal matrix function that orients the displaced cross section, and $\boldsymbol{\zeta}$ is the cross-section coordinate system given by $\boldsymbol{\zeta}^T = [0 \ X_2 \ X_3]$. The displacement field is

$$\mathbf{u} = \mathbf{x} - \mathbf{X} = \mathbf{u}_0 + (\mathbf{R}^T - \mathbf{I})\boldsymbol{\zeta}, \quad (49)$$

where $\mathbf{u}_0(X_1) = \mathbf{x}_0(X_1) - \mathbf{X}_0(X_1)$. The skew-symmetric curvature matrix is defined as $\tilde{\boldsymbol{\kappa}} = \mathbf{R} (d\mathbf{R}^T/dX_1)$. The curvature vector $\boldsymbol{\kappa}$ is the 3-vector extracted from $\tilde{\boldsymbol{\kappa}}$ as $\boldsymbol{\kappa} = \text{axial}(\tilde{\boldsymbol{\kappa}})$. The finite strains that contribute to the internal energy are the axial strain e_{11} and the two shear strains γ_{12} and γ_{13} :

$$\begin{aligned} e_{11} &= \mathbf{h}_1^T \mathbf{g}_1 + \frac{1}{2} \mathbf{g}_1^T \mathbf{H} \mathbf{g}_1, \\ \gamma_{12} &= 2e_{12} = \mathbf{h}_2^T \mathbf{g}_1 + \frac{1}{2} \mathbf{g}_2^T \mathbf{H} \mathbf{g}_1 + \mathbf{h}_1^T \mathbf{g}_2 + \frac{1}{2} \mathbf{g}_1^T \mathbf{H} \mathbf{g}_2, \\ \gamma_{13} &= 2e_{13} = \mathbf{h}_3^T \mathbf{g}_1 + \frac{1}{2} \mathbf{g}_3^T \mathbf{H} \mathbf{g}_1 + \mathbf{h}_1^T \mathbf{g}_3 + \frac{1}{2} \mathbf{g}_1^T \mathbf{H} \mathbf{g}_3. \end{aligned} \quad (50)$$

Here the 3×3 matrix $\mathbf{H}$ depends on the strain measured selected (for Green-Lagrange strains, $\mathbf{H} = \mathbf{I}$) and $\mathbf{g}_i$ are the displacement gradient 3-vectors

$$\mathbf{g}_1 = \frac{\partial \mathbf{u}_0}{\partial X_1} + \mathbf{R}^T \tilde{\boldsymbol{\zeta}}^T \boldsymbol{\kappa}, \quad \mathbf{g}_2 = (\mathbf{R}^T - \mathbf{I}) \mathbf{h}_2, \quad \mathbf{g}_3 = (\mathbf{R}^T - \mathbf{I}) \mathbf{h}_3, \quad (51)$$

in which $\mathbf{h}_j$ denotes the 3-vector whose j^{th} component is 1 and the other two are zero.

Introducing the *angular distortion vector* ϕ and its associated skew-symmetric matrix $\tilde{\phi}$ as

$$\phi = Rb_0, \quad b_0 = h_1 + \frac{\partial u_0}{\partial X_1}, \quad \phi = \text{axial}(\tilde{\phi}), \quad (52)$$

we can further split the strains as

$$e_b = \left(\frac{\partial u_0}{\partial X_1} \right)^T c_0, \quad e_f = \zeta^T \tilde{\phi} \kappa + \frac{1}{2} \kappa^T \tilde{\zeta} \zeta^T \kappa, \quad \gamma_{12} = h_2^T \phi + h_2^T \tilde{\zeta}^T \kappa, \quad \gamma_{13} = h_3^T \phi + h_3^T \tilde{\zeta}^T \kappa, \quad (53)$$

where $c_0 = h_1 + \frac{1}{2}(\partial u_0 / \partial X_1)$. Here e_b and e_f measure mean (bar-like) and flexural normal deformations. The two transverse shear strains are $\gamma_2 = h_2^T \phi$ and $\gamma_3 = h_3^T \phi$, while the torsional shear strain is composed of the terms $h_2^T \tilde{\zeta}^T \kappa$ and $h_3^T \tilde{\zeta}^T \kappa$.

The strain energy stored in the current configuration is

$$U = \frac{1}{2} \int_L \int_A \mathcal{U} dA dX_1, \quad \text{with} \quad \mathcal{U} = E e_{11}^2 + \mu G (\gamma_{12}^2 + \gamma_{13}^2), \quad (54)$$

where μ is a shear correction factor.¹⁰ The 9×9 core stiffnesses are expressed in terms of the the 9-component gradient vector $g^T = (g_1^T \quad g_2^T \quad g_3^T)$. Defining $c_i = h_i + \frac{1}{2} H g_i$ and $b_i = h_i + H g_i$ for $i = 1, 2, 3$, and applying the formulas of Section 4 we obtain for the spectral form at the energy level

$$S^U = \begin{bmatrix} ES_1^U + \mu G(S_2^U + S_3^U) & \mu GS_4^U & \mu GS_5^U \\ \mu GS_4^{UT} & \mu GS_1^U & 0 \\ \mu GS_5^{UT} & 0 & \mu GS_1^U \end{bmatrix} \quad (55)$$

where $S_1^U = c_1 c_1^T$, $S_2^U = c_2 c_2^T$, $S_3^U = c_3 c_3^T$, $S_4^U = c_2 c_1^T$ and $S_5^U = c_3 c_1^T$. At the residual level we obtain for the spectral form $S^r = S^r(0, 0)$

$$S^r = \begin{bmatrix} ES_1^r + \mu G(S_2^r + S_3^r) & \mu GS_4^r & \mu GS_5^r \\ \mu GS_4^r & \mu GS_1^r & 0 \\ \mu GS_5^r & 0 & \mu GS_1^r \end{bmatrix} \quad (56)$$

where $S_1^r = b_1 c_1^T$, $S_2^r = b_2 c_2^T$, $S_3^r = b_3 c_3^T$, $S_4^r = b_2 c_1^T$, $S_5^r = b_3 c_1^T$, $S_6^r = b_1 c_2^T$ and $S_7^r = b_1 c_3^T$. The internal force vector conjugate to δg is $\Phi = S^r g = \Phi_\sigma + \Phi_\tau$, in which

$$\Phi_\sigma = E \begin{Bmatrix} b_1 e_{11} \\ 0 \\ 0 \end{Bmatrix}, \quad \Phi_\tau = \mu G \begin{Bmatrix} b_2 \gamma_{12} + b_3 \gamma_{13} \\ b_1 \gamma_{12} \\ b_1 \gamma_{13} \end{Bmatrix}, \quad (57)$$

represent the contribution of the normal and shear stresses, respectively.

¹⁰ This factor is initially assumed to be the same for bending-shear and torsion for simplicity in the foregoing expressions. Distinction can be made if necessary after passing to stress resultants in the over-the-element integration process.

At the incremental level we operate with the second variation (15). Terms $\delta \mathbf{g}^T \mathbf{S}^r \delta \mathbf{g} + \delta \mathbf{g}^T \delta \mathbf{S}^r \mathbf{g}$ provide two components of the core tangent stiffness: $\mathbf{S}_M$ and $\mathbf{S}_{GI}$, which are independent of the choice of element degrees of freedom. On the other hand, the term $(\delta^2 \mathbf{g})^T \Phi$, which has to be retained because the relation between $\mathbf{g}$ and rotational degrees of freedom is nonlinear, leads to a geometric stiffness component $\mathbf{S}_{GD}$, which depends on the parametrization chosen to describe rotations (for example, Euler angles or Euler parameters).

The core material stiffness matrix is

$$\mathbf{S}_M = \begin{bmatrix} ES_1 + \mu G(S_2 + S_3) & \mu GS_4 & \mu GS_5 \\ \mu GS_4^T & \mu GS_1 & 0 \\ \mu GS_5^T & 0 & \mu GS_1 \end{bmatrix}, \quad (58)$$

where $S_1 = \mathbf{b}_1 \mathbf{b}_1^T$, $S_2 = \mathbf{b}_2 \mathbf{b}_2^T$, $S_3 = \mathbf{b}_3 \mathbf{b}_3^T$, $S_4 = \mathbf{b}_2 \mathbf{b}_1^T$ and $S_5 = \mathbf{b}_3 \mathbf{b}_1^T$. The parametrization-independent geometric stiffness is

$$\mathbf{S}_{GI} = \begin{bmatrix} Ee_{11}H & \mu G\gamma_{12}H & \mu G\gamma_{13}H \\ \mu G\gamma_{12}H & 0 & 0 \\ \mu G\gamma_{13}H & 0 & 0 \end{bmatrix}. \quad (59)$$

The computation of $\mathbf{S}_{GD}$ depends on the rotation parametrization; we refer to [5] for details. The core tangent stiffness matrix is the combination $\mathbf{S} = \mathbf{S}_M + \mathbf{S}_{GI} + \mathbf{S}_{GD}$. Completion of the element derivation require the following transformation steps, which are only outlined for brevity.

1. The variations $\delta \mathbf{g}$ are related to variations of a vector $\mathbf{q}$ of generalized coordinates:

$$\delta \mathbf{g} = \begin{bmatrix} \mathbf{I} & \mathbf{R}^T \tilde{\boldsymbol{\zeta}}^T & \mathbf{R}^T \tilde{\boldsymbol{\kappa}} \tilde{\boldsymbol{\zeta}}^T \\ 0 & 0 & \mathbf{R}^T \tilde{\mathbf{h}}_2^T \\ 0 & 0 & \mathbf{R}^T \tilde{\mathbf{h}}_3^T \end{bmatrix} \delta \mathbf{q}, \quad \text{in which} \quad \delta \mathbf{q} = \begin{Bmatrix} \frac{d\delta \mathbf{u}_0}{dX_1} \\ \frac{d\delta \boldsymbol{\Theta}}{dX_1} \\ \delta \boldsymbol{\Theta} \end{Bmatrix}, \quad (60)$$

where $\mathbf{I}$ is the 3×3 identity matrix and $\delta \boldsymbol{\Theta}$ is the axial vector associated with the skew-symmetric matrix $\delta \tilde{\boldsymbol{\Theta}} = \mathbf{R} \delta \mathbf{R}^T$.

2. The variations of $\mathbf{q}$ are related to variations of nodal degrees of freedom $\mathbf{v}$ of the beam element. Euler parameters are chosen as rotational parameters and related to angular coordinates. As usual for Timoshenko beam elements, translational displacements are linearly interpolated. For the rotations, however, the shape functions contain trigonometric terms so that constant curvature states can be exactly represented for arbitrary node rotations. The physical stiffnesses are obtained by the method explained in Section 6. Finally, integrated stress-resultant expressions are simplified by introducing principal and polar moments of inertia.

Procedural details pertaining to these two steps are given in [5]. Two highly nonlinear problems solved with this formulation are reported below.

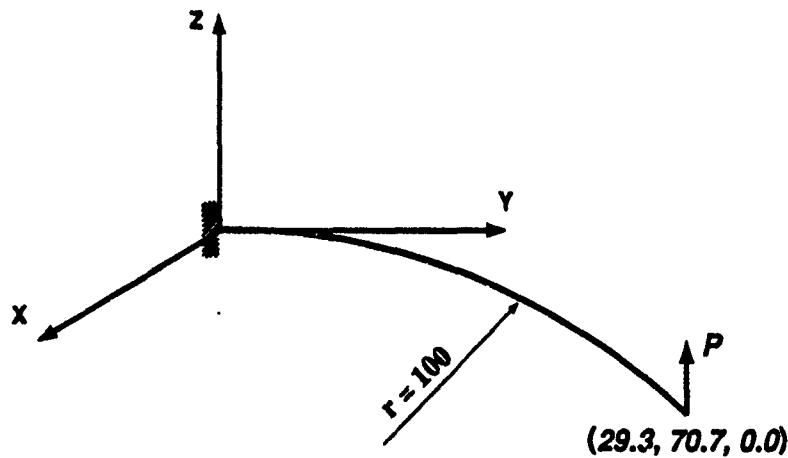


Fig. 2: Curved cantilever bend under tip load: Problem definition.

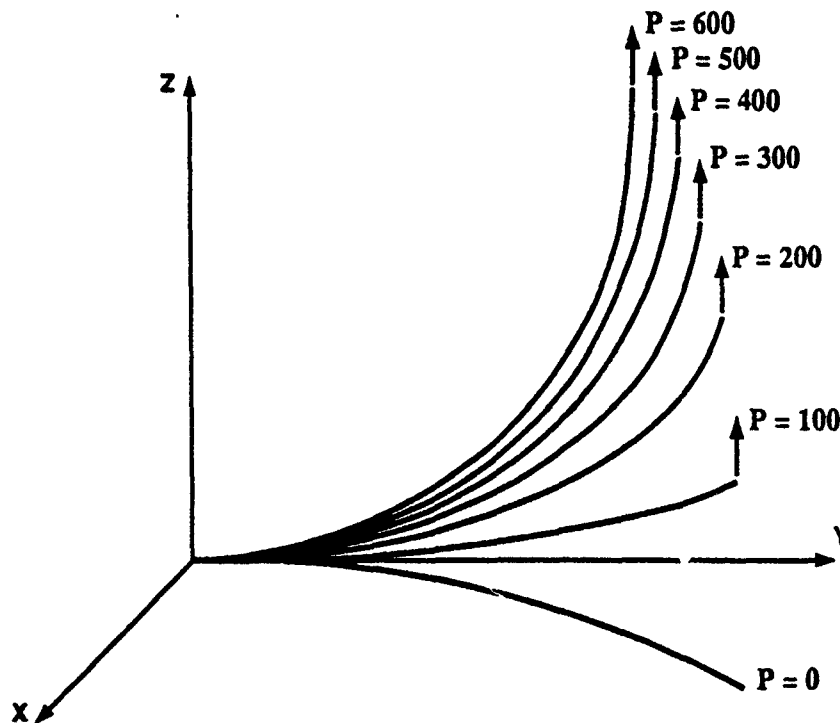


Fig. 3: Curved cantilever bend under tip load: Deflected shapes.

10 Application Problem 1: Large Displacement of a 45° Cantilever Bend

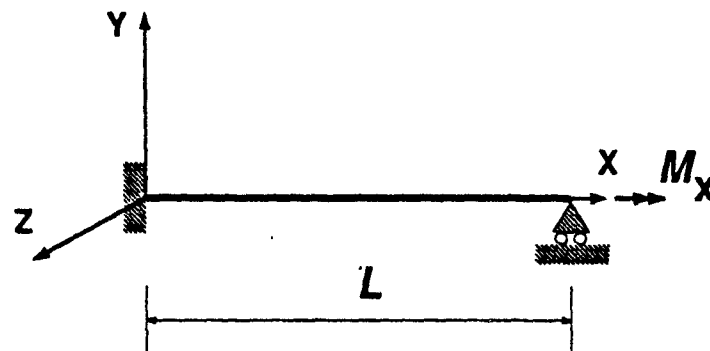
The first problem concerns the large displacement analysis of a 45° cantilever bend initially lying on the horizontal (X,Y) plane. The bend is an arc of a circle of radius $r = 100$ and the beam cross-section is a square with sides of unit length. The beam has modulus $E = 10^7$ and Poisson's ratio $\nu = 0$. It is subjected to an end load $P = 600$ normal to the (X,Y) plane as shown in Figure 2. The model consists of 8 straight beam

elements, and the full load was applied in 6 equal increments. The solution method is incremental-iterative with full Newton iteration used in the corrective phase.

This problem was treated by Bathe and Bolourchi [7] with 3D brick elements, and subsequently by Simo and Vu-Quoc [8] and Cardona [9] with beam elements based on other formulations. Results for the 3 tip displacement components are compared below.

	Load $P = 300$	Load $P = 450$	Load $P = 600$
Bathe <i>et.al.</i> [7]	22.33, 58.84, 40.08	18.62, 53.32, 48.39	15.79, 47.23, 53.37
Simo <i>et.al.</i> [8]	22.50, 59.20, 39.50		15.90, 47.20, 53.40
Cardona [9]	22.14, 58.64, 40.35	18.38, 52.11, 48.59	15.55, 47.04, 53.50
Present	22.31, 58.85, 40.08	18.59, 53.34, 48.39	15.75, 47.25, 53.37

As can be observed the present results compare well with those obtained with 3D elements in [7]. Deflected shapes for selected load levels are shown in Figure 3.



Length	$L = 240 \text{ mm}$
Polar Moment	$I_x = 2.16 \text{ mm}^4$
Second Moments	$I_y = I_z = 0.0833 \text{ mm}^4$
Young's Modulus	$E = 71240 \text{ N/mm}^2$
Shear Modulus	$G = 27190 \text{ N/mm}^2$

Fig. 4: Cable hocking: Problem definition.

11 Application Problem 2: Cable Hocking

The second problem, *cable hocking*, is more computationally demanding. An initially straight cable, modelled as a beam, is subjected to a tip torsional moment. The geometry and physical properties are given in Figure 4. The cable is clamped at one end and supported at the other so that the only motions allowed at that point are axial displacement and torsional rotation about the longitudinal X axis. No rotation is allowed about Y or Z . The purpose of these restrictions is to keep the problem conservative, because

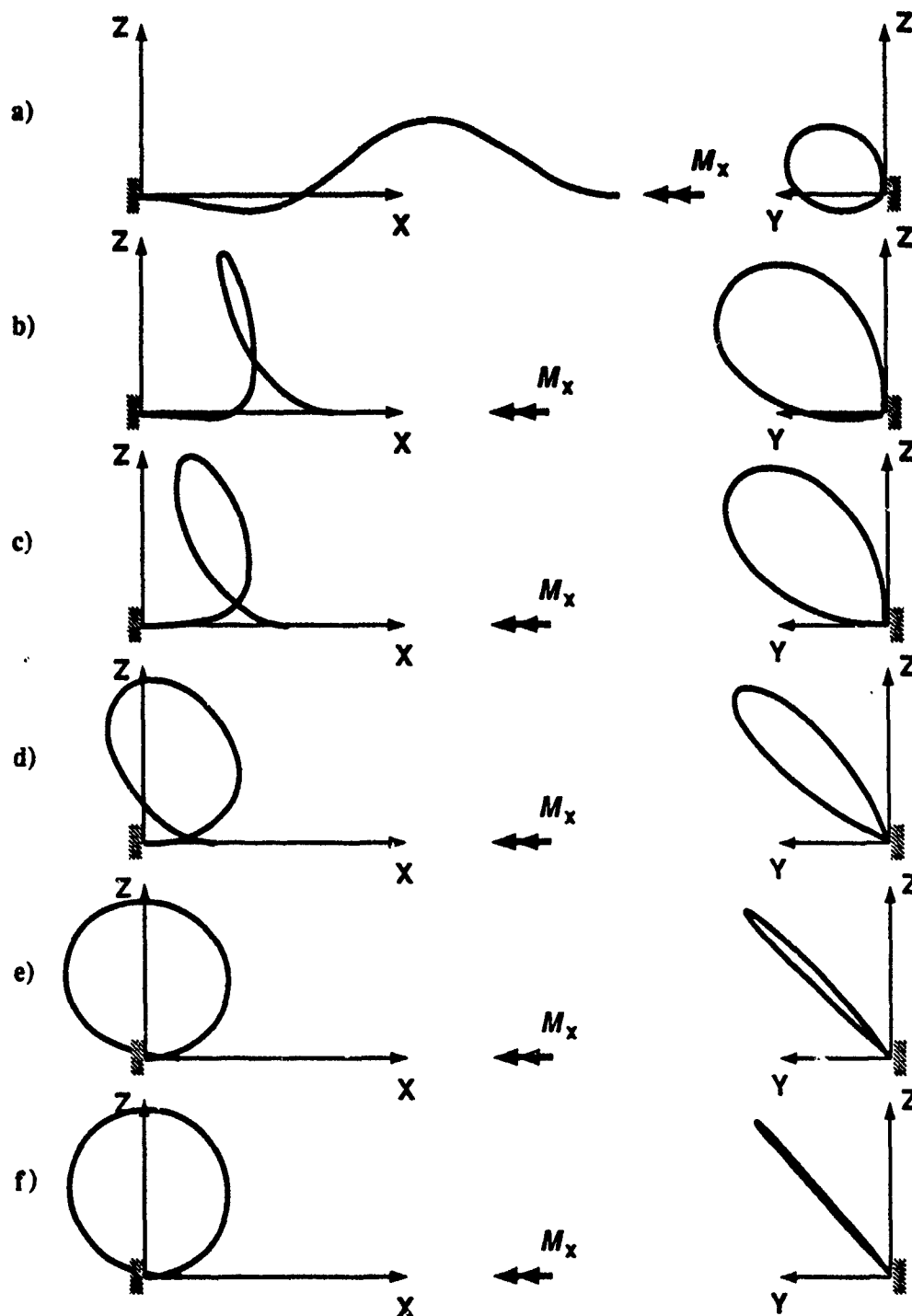


Fig. 5: Cable hocking: deformed shapes at different load levels:
a) After bifurcation, b) $M_x = 50$, c) $M_x = 0$, d) $M_x = -50$,
e) At the limit point, f) $M_x = 0$.

if the torque is allowed to rotate about Y or Z , the problem becomes nonconservative and dynamical methods are required to assess its stability [10,11].

This problem has received a great deal of attention from the engineering community due to its practical importance. The main objective is to estimate the critical applied torque at which the cable departs (bifurcates) from its straight configuration, resulting

in the formation of a loop or hockle. This has direct application to marine cables used in tasks such as lifting objects from the ocean floor, for which structural failure could be disastrous. Under the assumptions of infinitesimal bending deformations, Greenhill obtained an analytical formula to predict the critical torque; see *e.g.* pp. 417–418 of Love [12]. The post-bifurcation response analysis of this problem, however, has not been pursued until recently, as discussed in the research conducted by Nour-Omid and Rankin [13]. This postcritical response has also been analyzed by using the present TL formulation. The structure is discretized by twenty equally-spaced beam elements.

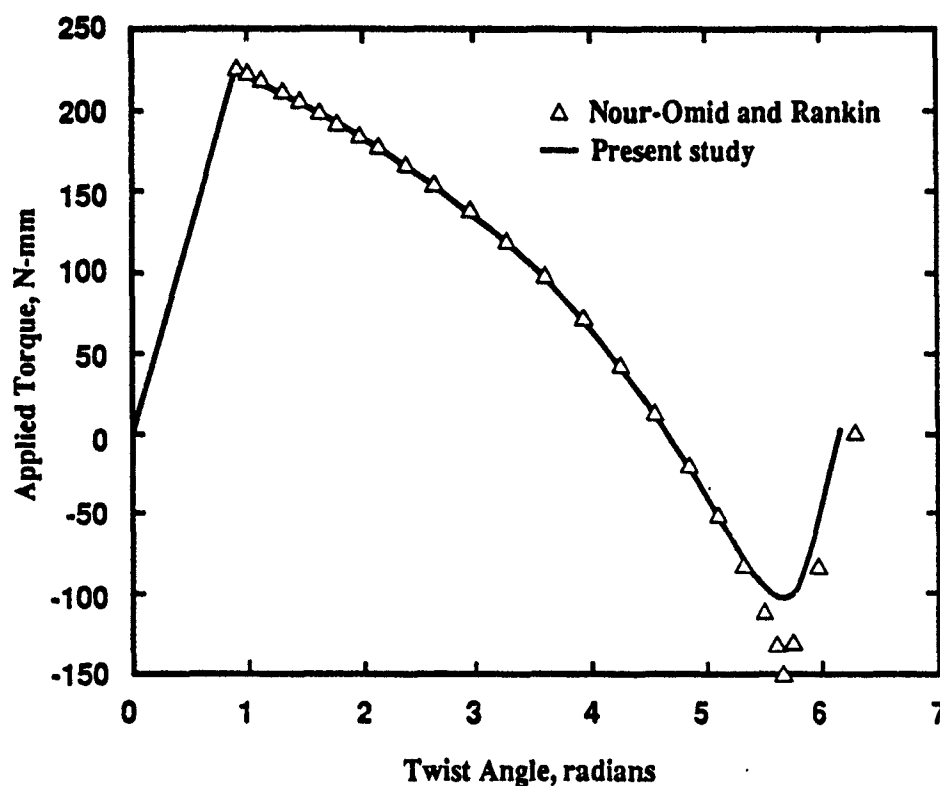


Fig. 6: Cable hocking: Tip rotation vs. applied moment.

The deformed shapes at different load levels are shown in Figure 5, which displays the loop-formation process previously mentioned. The curves on the left and right side show deformed shapes looking along the Y and Z axes, respectively.

If the torque is held under the critical value, the beam twists without lateral deflection. Along this fundamental path the response is linear. At the critical torque a bifurcation point is reached, at which the fundamental path becomes unstable. The beam acquires a helical shape with the free end moving towards the clamped end. This new equilibrium branch is unstable and the cable undergoes large displacements and rotations as the moment decreases. The unloading process continues until reaching a sharp limit point which corresponds to a *negative* value of the applied torque (it should be mentioned that

Nour-Omid and Rankin [13], who use a Hermitian beam element based on corotational description, characterize this critical point as a secondary bifurcation; evidence for categorization is presently inconclusive). After traversing this point, the torque reverses again until the cable reaches a circular-shaped unloaded deformed configuration.

The computed variation of the twist angle at the moving end versus the applied torque is given in Figure 6. Results are compared to those given by Nour-Omid and Rankin.

Acknowledgement

The research reported herein has been supported by the Air Force Office of Scientific Research under Grant F49620-87-C-0074.

References

1. S. Rajasekaran and D. W. Murray, Incremental finite element matrices, *J. Str. Div. ASCE*, 99, pp. 2423-2438, 1973.
2. R. H. Mallet and P. V. Marcal, Finite element analysis of nonlinear structures, *J. Str. Div. ASCE*, 94, pp. 2081-2105, 1968.
3. D. W. Murray, Finite element nonlinear analysis of plates, Ph. D. Dissertation, Dept. of Civil Engineering, University of California, Berkeley, California, 1967.
4. C. A. Felippa, Discussion of [1], *J. Str. Div. ASCE*, 100, pp. 2519-2521, 1974.
5. L. A. Crivelli, A Total-Lagrangian beam element for analysis of nonlinear space structures, Ph. D. Dissertation, Dept. of Aerospace Engineering Sciences, University of Colorado, Boulder, 1990.
6. K. Mathiasson, A. Bengtson and A. Samuelsson, On the accuracy and efficiency of numerical algorithms for geometrically nonlinear structural analysis, in P. G. Bergan, K.-J. Bathe and W. Wunderlich (eds.) *Finite Element Methods for Nonlinear Problems*, Springer-Verlag, Berlin, pp. 3-24, 1986.
7. K. J. Bathe and S. Bolourchi, Large displacement analysis of three-dimensional beam structures, *Int. J. Numer. Meth. Engrg.*, 14, pp. 961-986, 1979.
8. J. C. Simo and L. Vu-Quoc, A three-dimensional finite strain rod model. Part II: Computational aspects, *Comp. Meths. Appl. Mech. Engrg.*, 58, pp. 79-116, 1986.
9. A. Cardona, An integrated approach to mechanism analysis, Ph. D. Dissertation, LTAS, Université de Liège, Belgium, 1989.
10. V. V. Bolotin, *Nonconservative Problems of the Theory of Elastic Stability*, Pergamon Press, New York, 1963.
11. H. Ziegler, *Principles of Structural Stability*, Blaisdell, Massachusetts, 1968.
12. A. E. H. Love, *The Mathematical Theory of Elasticity*, Dover, New York, 1944.
13. B. Nour-Omid and C. C. Rankin, Finite rotation analysis and consistent linearization using projectors, submitted to *Comp. Meths. Appl. Mech. Engrg.*, 1990.

Addendum

This Addendum contains the text of References [1] and [4]. As noted in the paper, these are historically important in the staged development of the core-congruential formulation.

JOURNAL OF THE STRUCTURAL DIVISION

INCREMENTAL FINITE ELEMENT MATRICES

By Sundaramoorthy Rajasekaran¹ and David W. Murray,² M. ASCE

INTRODUCTION

A common technique in geometrically nonlinear finite element analysis is to express the total potential in terms of Lagrangian displacement coordinates, differentiate the potential to obtain the equilibrium equations, and form the differentials in the equilibrium equations to obtain linear incremental equilibrium equations. Mallett and Marcal (3) introduced a standard nomenclature for this procedure by writing the total potential as

$$\Pi = q^T \left[\frac{1}{2} K + \frac{1}{6} N_1 + \frac{1}{12} N_2 \right] q - q^T P \quad (1)$$

the equilibrium equation as

$$\left[K + \frac{1}{2} N_1 + \frac{1}{3} N_2 \right] q - P = \{0\} \quad (2)$$

and the linear incremental equilibrium equation as

$$[K + N_1 + N_2] \Delta q - \Delta P = \{0\} \quad (3)$$

The argument for this notation is that the scalar multiples in Eq. 2 follow from those in Eq. 1, and those in Eq. 3 follow from those in Eq. 2, because of the linear and quadratic dependence of N_1 and N_2 , respectively, on the displacement coordinates, q . Other investigators appear to have accepted this argument and adopted the formalism implied by this notation (1,2,5). The matrices,

Note.—Discussion open until May 1, 1974. To extend the closing date one month, a written request must be filed with the Editor of Technical Publications, ASCE. This paper is part of the copyrighted Journal of the Structural Division, Proceedings of the American Society of Civil Engineers, Vol. 99, No. ST12, December, 1973. Manuscript was submitted for review for possible publication on March 29, 1973.

¹Asst. Prof. of Civ. Engrg., PSG Coll. of Tech., Coimbatore, India.

²Prof. of Civ. Engrg., Univ. of Alberta, Edmonton, Alberta, Canada.

N_1 and N_2 , were designated as "incremental stiffness matrices."

The purpose of this paper is to show that, in general, Eqs. 2 and 3 do not necessarily follow from Eq. 1. The difficulty with the notation arises from the fact that the expressions used to evaluate N_1 and N_2 in Eq. 1 are not unique. There are, however, particular forms of expressions for these matrices for which the formalism of Eqs. 1, 2, and 3 is always valid, and general expressions for which this is true are derived in this paper.

TOTAL POTENTIAL

The engineering components of strain, consistent with Green's strain tensor, may be expressed as

$$\begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix} = \begin{bmatrix} u_x + \frac{1}{2} \{ (u_x)^2 + (v_x)^2 + (w_x)^2 \} \\ v_y + \frac{1}{2} \{ (u_y)^2 + (v_y)^2 + (w_y)^2 \} \\ w_z + \frac{1}{2} \{ (u_z)^2 + (v_z)^2 + (w_z)^2 \} \\ (u_y + v_x) + \{ u_y u_x + v_y v_x + w_y w_x \} \\ (v_z + w_y) + \{ u_z u_y + v_z v_y + w_z w_y \} \\ (w_x + u_z) + \{ u_x u_z + v_x v_z + w_x w_z \} \end{bmatrix} \quad (4)$$

in which u , v , and w = the displacements in the x , y , and z coordinate directions; and a comma denotes partial differentiation with respect to the following coordinate. These six strain components are designated as ϵ_i ($i = 1, 6$). The corresponding stresses may be designated as σ_i ($i = 1, 6$), and for a linear elastic constitutive matrix, C_{ij} , we may write

$$\sigma_i = C_{ij} \epsilon_j \quad (5)$$

in which the summation convention is implied. The strain energy in any linear elastic structure may then be evaluated as

$$U = \frac{1}{2} \int_V \epsilon_i C_{ij} \epsilon_j dV \quad (6)$$

Each strain component in Eq. 4 may be decomposed into two parts and written as

$$\epsilon_i = \epsilon_i^L + \epsilon_i^{NL} \quad (7)$$

in which ϵ_i^L is linearly dependent on displacements; and ϵ_i^{NL} is quadratically dependent on displacements.

The total potential for the structure may, therefore, be expressed through Eqs. 6 and 7, as

$$\Pi = \frac{1}{2} \sum_{n=1}^N \int_V (\epsilon_i^L + \epsilon_i^{NL}) C_{ij} (\epsilon_j^L + \epsilon_j^{NL}) dV - q^T P \quad (8)$$

in which N = the number of elements in the structure; q = the vector of displacement coordinates; and P = the associated vector of generalized forces. Expanding Eq. 8 yields

$$\Pi = \frac{1}{2} \sum_{n=1}^N \int_V C_{ij} (\epsilon_i^L \epsilon_j^L + 2\epsilon_i^L \epsilon_j^{NL} + \epsilon_i^{NL} \epsilon_j^{NL}) dV - q^T P \quad (9)$$

in which the symmetry of C_{ij} has been recognized. The matrices K , N_1 , and N_2 of Eq. 1 arise from the corresponding terms of Eq. 9.

For the purposes of evaluating Eq. 9, each element may be considered individually in a local coordinate system. The operations of transforming from generalized element coordinates to nodal coordinates, from nodal coordinates to system coordinates, integrating over element volumes and assembling the entire structure, do not alter derivations carried out on the infinitesimal level, but simply complicate the notation. Therefore, attention will be focused on a single element.

ELEMENT POTENTIAL

The contribution of only a single element to Eq. 9 may be written, without complicating the notation, as

$$\Pi = \frac{1}{2} \int_V C_{ij} (\epsilon_i^L \epsilon_j^L + 2\epsilon_i^L \epsilon_j^{NL} + \epsilon_i^{NL} \epsilon_j^{NL}) dV - q^T P \quad (10)$$

Let us now express ϵ_i^L and ϵ_i^{NL} in matrix notation, and write Eq. 7 as

$$\epsilon_i = L_i^T d + \frac{1}{2} d^T H_i d \quad (11)$$

in which L_i = a vector; H_i = a symmetric matrix; and d = the vector of displacement gradients contributing to the strains, ϵ_i . The vector, d , is consistent with the nomenclature introduced in Ref. 3.

To fix ideas, consider the truss element of Fig. 1(a). There is only one nonzero strain component so the range of i in Eq. 11 is one, and the subscript can be dropped. For this special case Eq. 11 is (from Eq. 4)

$$\epsilon = u_x + \frac{1}{2} \{ (u_x)^2 + (v_x)^2 + (w_x)^2 \} \quad (12a)$$

$$\text{from which } d^T = \langle u_x \ v_x \ w_x \rangle \quad (12b)$$

$$L^T = \{ 1 \ 0 \ 0 \} \quad (12c)$$

$$H = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \quad (12d)$$

and $\epsilon = L^T d + \frac{1}{2} d^T H d$ (12c)

Eq. 11, in its more general form, represents each strain component by expressions of the type contained in Eqs. 12.

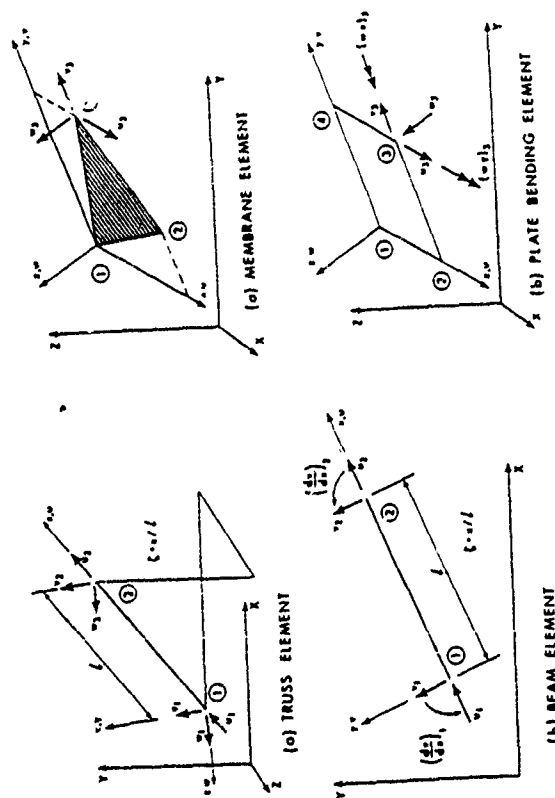


FIG. 1.—Typical "One-Dimensional" Elements

The particular finite element model arises on expressing displacements in terms of nodal, or generalized, coordinates. Thus, we may write symbolically.

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = N q \quad \dots \dots \dots (13)$$

and, differentiating to obtain displacement gradients

$$d = D q \quad \dots \dots \dots (14)$$

For example, for the truss element of Fig. 1(a), Eq. 13 takes the form

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \begin{bmatrix} 1-\xi & \xi & 0 \\ 0 & 1-\xi & \xi \\ 0 & 0 & 1-\xi \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ u_2 \\ v_2 \\ w_2 \end{Bmatrix} \quad \dots \dots \dots (15)$$

in which ξ = the nondimensional coordinate; and $u_1, v_1, w_1, u_2, v_2, w_2$ = the nodal displacements of Fig. 1(a). Eq. 15 defines the N matrix and the displacement coordinate vector, q , of Eq. 13, for this example. Differentiating Eq. 15 to form the displacement gradients in the vector, d , defined in Eq. 12b, results in

$$\begin{Bmatrix} u,x \\ v,x \\ w,x \end{Bmatrix} = \frac{1}{l} \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} q \quad \dots \dots \dots (16)$$

which defines the matrix, D, of Eq. 14, for this example.

Returning to a more general context, substitution of Eq. 11 into Eq. 10 allows the total potential of a single element to be written as

$$\Pi = \frac{1}{2} \int_V C_{ij} d^T \left[L_i L_j^T + L_i d^T H_j + \frac{1}{4} H_i d d^T H_j \right] d dV - q^T P \quad \dots \dots \dots (17)$$

in which the symmetry of C_{ij} and H have been recognized. The potential may also be written in terms of the displacement coordinates, q , of Eq. 14, as

$$\Pi = \frac{1}{2} q^T \int_V C_{ij} D^T \left[L_i L_j^T + L_i d^T H_j + \frac{1}{4} H_i d d^T H_j \right] D dV q - q^T P \quad (18)$$

Either Eqs. 17 or 18 may be used in the following developments but Eq. 17 is adopted because it is slightly simpler in form, and independent of the shape functions, displacement coordinates, or geometric configuration of the element selected. Eqs. 17 or 18 will be written symbolically as

$$\Pi = \int_V I_1 dV + \int_V I_2 dV + \int_V I_3 dV - q^T P \quad \dots \dots \dots (19)$$

in which I_1, I_2 , and I_3 = the integrands defined by identification with the corresponding terms of the preceding equations.

BASIC EQUATIONS

Again following notation similar to, but not identical with, that of Ref. 3, it is desired to write Eq. 17 in the symbolic form

$$\Pi = \int_V d^T \left[\frac{1}{2} \hat{K} + \frac{1}{6} \hat{N}_1 + \frac{1}{12} \hat{N}_2 \right] d dV - q^T P \quad \dots \dots \dots (20)$$

However, it is also desirable that the formalism of Eqs. 1, 2, and 3 apply to the result, so one cannot directly identify the terms in Eq. 17 with those in Eq. 20 without further investigation.

The first term in Eq. 17 may be expressed in only one way and a comparison of Eqs. 17, 19, and 20 indicates that if

$$I_1 = \frac{d^T}{2} \hat{K} d = \frac{1}{2} C_{ij} L_i L_j^T d \quad \dots \dots \dots (21a)$$

for arbitrary d , then

which is appropriate for the potential energy expression, the equilibrium equations.

A similar nonuniqueness occurs in the expression for the third term of Eq. 17 which may again be written in such a way that two different symmetric matrices may arise. The integrand for the third term may be written as

$$I_3 = \frac{C_y}{8} d^T (H_i d^T H_j) d \quad (25a)$$

$$I_3 = \frac{C_y}{8} d^T (d^T H_j d H_i) d \quad (25b)$$

Either of these forms is adequate for the potential energy expression. The right-hand side of Eq. 25a will also repeat in the equilibrium equation, but in order to have a form which will be valid for all three equations it is necessary to take the linear combination consisting of one-third of the sum of the right-hand side of Eq. 25b plus twice the right-hand side of Eq. 25a, i.e., we may write

$$I_3 = \frac{C_y}{12} d^T \left(H_i d^T H_j + \frac{1}{2} d^T H_j d H_i \right) d \quad (25c)$$

$$\text{which yields } \hat{N}_3 = C_y \left(H_i d^T H_j + \frac{1}{2} d^T H_j d H_i \right) \quad (26)$$

For comparison in the following examples we define, from the right side of Eq. 25a, the matrix

$$\hat{N}_3^* = \frac{3}{2} C_y H_i d^T H_j \quad (27)$$

which is a matrix appropriate for the potential energy expression and the equilibrium equations, but is inappropriate for the incremental equilibrium equations.

MATRICES FOR COMPARISON PURPOSES

Eq. 20 represents the total potential for an element, with the strain energy expressed in terms of displacement gradients, and several comments can be made:

1. An equation of the type displayed as Eq. 14 may be used to express the vector of displacement gradients in Eq. 20, in terms of generalized, or nodal, or system coordinates as demonstrated in Eq. 18. The level on which comparisons of these matrices are made is immaterial in a Lagrangian formulation.
2. For the matrices of Eq. 20 defined by Eqs. 21b, 23, and 26, the formalism of Eqs. 1, 2, and 3 is valid, regardless of the Lagrangian coordinates to which the formulation is referenced. However, if, for example, matrices are defined by Eqs. 24 and 27, this formalism is invalid (as demonstrated in Appendix 1).
3. In the following comparisons, matrices will be partially integrated throughout the volume, in order that a meaningful comparison can be carried out. The

$$\hat{K} = C_y L_i L_j^T \quad (21b)$$

and is unique. However the second term in Eq. 17 may be expressed in a variety of ways giving rise to a variety of different matrices in Eq. 20. It should be noted that

$$I_2 = \frac{C_y}{2} d^T (L_i d^T H_j) d \quad (22a)$$

$$I_2 = \frac{C_y}{4} d^T (L_i d^T H_j + H_i d^T L_j^T) d \quad (22b)$$

$$I_2 = \frac{C_y}{2} d^T (d^T L_i H_j) d \quad (22c)$$

In all of the following it must be recognized that $L^T d = d^T L$ is a scalar, and that $d^T H d$ is also a scalar. These scalar forms may be commuted, transposed (recall that H is symmetric), and inserted in arbitrary locations within other matrix products. In addition, since C_y is symmetric, the summation indices may be interchanged at will. Of the three forms of I_2 displayed in Eqs. 22, the right-hand side of Eq. 22a is unsymmetric and regarded as unsuitable. However, the right-hand sides of Eqs. 22b and 22c both yield symmetric matrices. Although either of these forms would prove satisfactory in the potential energy equation, neither of them will repeat in the equilibrium equation upon differentiation of the potential energy equation. A symmetric form which will repeat is obtained by taking one-third of the sum of the right-hand side of Eq. 22c plus twice the right-hand side of Eq. 22b. In other words, if the integrand of the second term of Eq. 17 is expressed as

$$I_2 = \frac{C_y}{2} d^T (L_i d^T H_j) d;$$

$$I_2 = \frac{1}{3} \left\{ 2 \frac{C_y}{4} d^T (L_i d^T H_j + H_i d^T L_j^T) d + \frac{C_y}{2} d^T (d^T L_i H_j) d \right\};$$

$$I_2 = \frac{1}{6} C_y d^T (L_i d^T H_j + d^T L_i H_j + H_i d^T L_j^T) d \quad (22d)$$

then $\hat{N}_1$, of Eq. 20, may be defined as

$$\hat{N}_1 = C_y (L_i d^T H_j + d^T L_i H_j + H_i d^T L_j^T) \quad (23)$$

and this form of $\hat{N}_1$ will repeat in both the equilibrium equation and the incremental equilibrium equation on carrying out the differentiation. A technique for verifying this statement is demonstrated in Appendix 1.

For comparison in the following examples we define, from the right-hand side of Eq. 22b, the matrix

$$\hat{N}_1^* = \frac{3}{2} C_y (L_i d^T H_j + H_i d^T L_j^T) \quad (24)$$

number of spatial integrations that has been carried out will be indicated behind the matrix [i.e., $\hat{N}_1(1)$, $\hat{N}_1(2)$, $\hat{N}_1(3)$ will indicate that the matrix, $\hat{N}_1$, has been integrated in 1, 2, and 3 spatial directions, respectively].

ILLUSTRATIVE EXAMPLES

Example No. 1—Truss Elements.—The relevant displacement assumptions and matrices associated with a large displacement formulation of the truss element of Fig. 1(a) are as follows. Note that $\hat{N}_1^e \neq \hat{N}_1$ and $\hat{N}_2^e \neq \hat{N}_2$.

The strain description is

$$\epsilon = u_x + \frac{1}{2} \{(u_x)^2 + (v_x)^2 + (w_x)^2\} = L^T d + \frac{1}{2} d^T H d \quad (28)$$

$$\text{in which } d^T = \langle u_x \ v_x \ w_x \rangle \quad (29)$$

$$\text{and } L^T = \langle 1 \ \cdot \ \cdot \rangle \quad (30)$$

$$H = \begin{bmatrix} 1 & \cdot & \cdot \\ \cdot & 1 & \cdot \\ \cdot & \cdot & 1 \end{bmatrix} \quad (31)$$

The displacement description is

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \begin{bmatrix} 1 - \zeta & \cdot & \cdot \\ \cdot & 1 - \zeta & \cdot \\ \cdot & \cdot & 1 - \zeta \end{bmatrix} \begin{Bmatrix} \xi \\ \eta \\ \zeta \end{Bmatrix} \quad (32)$$

$$\text{in which } q^T = \langle u \ v \ w \ u_x \ v_x \ w_x \rangle \quad (33)$$

$$\text{so that } D = \frac{1}{l} \begin{bmatrix} -1 & \cdot & \cdot \\ \cdot & -1 & \cdot \\ \cdot & \cdot & -1 \end{bmatrix} \quad (34)$$

The adequate matrices are

$$\hat{K}(3) = EA/LL^T = EA \begin{bmatrix} 1 & \cdot & \cdot \\ \cdot & 1 & \cdot \\ \cdot & \cdot & 1 \end{bmatrix} \quad (35)$$

$$\hat{N}_1(3) = EA \{ Ld^T H + Hd^T L + HdL^T \} \quad (36)$$

$$\hat{N}_1(3) = EA \begin{bmatrix} 3u_x & v_x & w_x \\ v_x & u_x & \cdot \\ w_x & \cdot & u_x \end{bmatrix} \quad (37)$$

$$\text{and } \hat{N}_2(3) = EA \left(H d d^T H + \frac{1}{2} d^T H d H \right)$$

FINITE ELEMENT MATRICES

ST12

$$= EA \begin{bmatrix} \Delta + (u_x)^2 & u_x v_x & u_x w_x \\ v_x u_x & \Delta + (v_x)^2 & v_x w_x \\ w_x u_x & w_x v_x & \Delta + (w_x)^2 \end{bmatrix} \quad (37)$$

$$\text{in which } \Delta = \frac{1}{2} \{(u_x)^2 + (v_x)^2 + (w_x)^2\} \quad (38)$$

The inadequate matrices are

$$\hat{N}_1^e(3) = \frac{3}{2} EA \begin{bmatrix} 2u_x & v_x & w_x \\ v_x & \cdot & \cdot \\ w_x & \cdot & \cdot \end{bmatrix} \quad (39)$$

$$\hat{N}_2^e(3) = \frac{3}{2} EA \begin{bmatrix} (u_x)^2 & u_x v_x & v_x w_x \\ v_x u_x & (v_x)^2 & v_x w_x \\ w_x u_x & w_x v_x & (w_x)^2 \end{bmatrix} \quad (40)$$

Example No. 2—In-Plane Bending.—This is the example of Ref. 3. The element is shown in Fig. 1(b) and subjected to the uniaxial strain

$$\epsilon = u_x - y v_{xx} + \frac{1}{2} (v_x)^2 \quad (41)$$

The reference coordinates, which may be used if desired, are the generalized coordinates, β , associated with a linear polynomial expansion of u and a cubic polynomial expansion of v . Equations and matrices for in-plane bending are as follows:

The stress description is

$$\epsilon = u_x - y v_{xx} + \frac{1}{2} (v_x)^2 = L^T d + \frac{1}{2} d^T H d \quad (42)$$

$$\text{in which } d^T = \langle u_x \ v_x \ v_{xx} \rangle \quad (43)$$

$$L^T = \langle 1 \ \cdot \ -y \rangle \quad (44)$$

$$\text{and } H = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \quad (45)$$

The displacement description is

$$\begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{bmatrix} 1 & x & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & x & x^2 \end{bmatrix} q \quad (46)$$

$$\text{in which } q^T = \langle \beta_1 \ \beta_2 \ \beta_3 \ \beta_4 \ \beta_5 \ \beta_6 \rangle \quad (47)$$

$$\text{so that } D = \begin{bmatrix} 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & 2x & 3x^2 \\ \cdot & \cdot & \cdot & \cdot & 2 & 6x \end{bmatrix} \quad (48)$$

The adequate matrices are

$$\hat{\mathbf{K}}(2) = E \int_A \mathbf{L} \mathbf{L}^T dA = E \begin{bmatrix} A & & \\ & \ddots & \\ & & I \end{bmatrix} \quad (49)$$

$$\hat{\mathbf{N}}_1(2) = E \int_A (\mathbf{L} \mathbf{d}^T \mathbf{H} + \mathbf{d}^T \mathbf{L} \mathbf{H} + \mathbf{H} \mathbf{d} \mathbf{L}^T) dA \approx EA \begin{bmatrix} v_x & \\ v_x & u_x \end{bmatrix} \quad (50)$$

$$\begin{aligned} \text{and } \hat{\mathbf{N}}_2(2) &= E \int_A \left(\mathbf{H} \mathbf{d} \mathbf{d}^T \mathbf{H} + \frac{1}{2} \mathbf{d}^T \mathbf{H} \mathbf{d} \mathbf{H} \right) dA \\ &= \frac{3}{2} EA \begin{bmatrix} \cdot & \cdot & \cdot \\ (v_x)^2 & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \quad (51) \end{aligned}$$

The inadequate matrices are

$$\hat{\mathbf{N}}_1^*(2) = \frac{3}{2} E \int_A (\mathbf{L} \mathbf{d}^T \mathbf{H} + \mathbf{H} \mathbf{d} \mathbf{L}^T) dA = \frac{3}{2} EA \begin{bmatrix} \cdot & v_x & \\ v_x & \cdot & \\ \cdot & \cdot & \cdot \end{bmatrix} \quad (52)$$

$$\text{and } \hat{\mathbf{N}}_2^*(2) = \frac{3}{2} E \int_A \mathbf{H} \mathbf{d} \mathbf{d}^T \mathbf{H} dA = \frac{3}{2} EA \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & (v_x)^2 & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \quad (53)$$

Note that $\hat{\mathbf{N}}_1^* \neq \hat{\mathbf{N}}_1$, but that Ref. 3 has used the proper form of this matrix. Note that for this particular problem $\hat{\mathbf{N}}_2^* = \hat{\mathbf{N}}_2$. This is a coincidence and results from the fact that $\mathbf{H} \mathbf{d} \mathbf{d}^T \mathbf{H} = \mathbf{d}^T \mathbf{H} \mathbf{d} \mathbf{H}$. It is perhaps this circumstance that led to the conclusion that Eqs. 2 and 3 follow from Eq. 1. That this is not, in general, true follows from Example No. 1.

While the in-plane bending problem leads to a particularly simple form of the incremental matrices, more general problems of thin walled beams do not lead to these simple forms. It is of interest that Ref. 1 uses a form, $\hat{\mathbf{N}}_1$ inappropriate for either Eqs. 2 or 3, and a form of $\hat{\mathbf{N}}_2$ inappropriate for Eq. 3.

Example No. 3.—Membrane Elements.—Ref. 3 also contains the derivation of incremental stiffnesses for an anisotropic membrane element. The relevant equations and matrices, in the present notation, are displayed as follows for a general form of element in the coordinate system of Fig. 2(a), and assuming $C_{13} = C_{31} = C_{23} = C_{32} = 0$:

The strain description is

$$\epsilon_1 = \epsilon_x = u_x + \frac{1}{2} (w_x)^2 \quad (54a)$$

$$\epsilon_2 = \epsilon_y = v_y + \frac{1}{2} (w_y)^2 \quad (54b)$$

$$\epsilon_3 = \gamma_{xy} = u_y + v_x + w_x w_y \quad (54c)$$

$$\text{so that } \epsilon_i = \mathbf{L}_i^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \mathbf{H}_i \mathbf{d} \quad (54d)$$

$$\text{in which } \mathbf{d}^T = \langle u_x, u_y, v_x, v_y, w_x, w_y \rangle \quad (55)$$

$$\text{and } \mathbf{L}_1^T = \langle 1 \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \rangle \quad (56)$$

$$\mathbf{L}_2^T = \langle \cdot \quad 1 \quad \cdot \quad \cdot \quad \cdot \quad \cdot \rangle \quad (57)$$

$$\mathbf{L}_3^T = \langle \cdot \quad \cdot \quad 1 \quad \cdot \quad \cdot \quad \cdot \rangle \quad (58)$$

$$\mathbf{H}_1 = \text{diag} \langle \cdot \quad \cdot \quad 1 \quad \cdot \quad \cdot \quad \cdot \rangle \quad (59)$$

$$\mathbf{H}_2 = \text{diag} \langle \cdot \quad \cdot \quad \cdot \quad 1 \quad \cdot \quad \cdot \rangle \quad (60)$$

$$\text{and } \mathbf{H}_3 = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 \end{bmatrix} \quad (61)$$

The adequate matrices are

$$\hat{\mathbf{K}}(1) = C_{ij} \int_{-1/2}^{1/2} \mathbf{L}_i \mathbf{L}_j^T dz = \begin{bmatrix} C_{11} & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & C_{33} & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & C_{33} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & C_{33} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & C_{22} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \quad (62)$$

(sym)

$$\hat{\mathbf{N}}_1(1) = C_{ij} \int_{-1/2}^{1/2} (\mathbf{L}_i \mathbf{d}^T \mathbf{H}_j + \mathbf{d}^T \mathbf{L}_i \mathbf{H}_j + \mathbf{H}_i \mathbf{d} \mathbf{L}_j^T) dz \quad (63)$$

$$\mathbf{N}_1(1) = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} C_{11} w_x & C_{12} w_y \\ C_{33} w_x & C_{33} w_y \\ C_{33} w_x & C_{33} w_y \\ C_{33} w_x & C_{33} w_y \\ C_{12} w_x & C_{22} w_y \\ (C_{11} u_x + C_{12} v_x) & C_{33} (u_y + v_x) \\ (C_{22} v_x + C_{12} u_x) & \cdot \end{bmatrix} \quad (64)$$

(symmetric)

$$\text{and } \hat{N}_2(I) = C_{ij} \int_{-1/2}^{1/2} \left(H_i d^T H_j + \frac{1}{2} d^T H_j d H_i \right) dz \quad (65)$$

$$\hat{N}_2(I) = t \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{matrix} A \\ B \\ C \end{matrix} \quad (66)$$

$$\text{in which } \left. \begin{aligned} A &= \frac{3}{2} C_{11} w_x^2 + \frac{C_{12}}{2} w_y^2 + C_{33} w_z^2 \\ B &= 2 C_{33} w_x w_y + C_{12} w_x w_z \\ C &= \frac{3}{2} C_{22} w_y^2 + C_{33} w_z^2 + \frac{C_{12}}{2} w_x^2 \end{aligned} \right\} \quad (67)$$

The inadequate matrices are

$$\hat{N}_1^*(I) = \frac{3}{2} \int_{-1/2}^{1/2} C_{ij} (L_i d^T H_j + H_i d^T L_j^T) dz \quad (68)$$

$$\hat{N}_1^*(I) = \frac{3}{2} t \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{matrix} C_{11} w_x \\ C_{33} w_y \\ C_{33} w_z \\ C_{33} w_x \\ C_{22} w_y \\ C_{22} w_z \end{matrix} \quad (69)$$

$$\text{and } \hat{N}_2^*(I) = \frac{3}{2} C_{ij} \int_{-1/2}^{1/2} H_i d^T H_j dz \quad (70)$$

$$\hat{N}_2^*(I) = \frac{3}{2} t \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{matrix} (C_{11} w_x^2 + C_{33} w_z^2) w_x \\ (C_{11} w_x^2 + C_{33} w_z^2) w_y \\ (C_{11} w_x^2 + C_{33} w_z^2) w_z \\ (C_{12} + C_{33}) w_x w_y \\ (C_{12} + C_{33}) w_x w_z \\ (C_{12} + C_{33}) w_y w_z \end{matrix} \quad (71)$$

Note that $\hat{N}_1^* \neq \hat{N}_1$ but that Ref. 3 uses the appropriate form. Note, however, that the form of $\hat{N}_2$ in Ref. 3 is inappropriate to permit the formalism implied by the paper, and, indeed, remains inappropriate even after the additional terms have been inserted in the closure to this paper (4).

Example No. 4—Plate Bending Elements.—The relevant equations and matrices for plate bending elements are shown as follows for the coordinate system of Fig. 2(b). Note that all incremental matrices follow directly from those of the membrane element.

The strain description is

$$\epsilon_1 = \epsilon_x = u_x - z w_{,xx} + \frac{1}{2} (w_{,x})^2 \quad (72)$$

$$\epsilon_2 = \epsilon_y = v_y - z w_{,yy} + \frac{1}{2} (w_{,y})^2 \quad (73)$$

$$\epsilon_3 = \gamma_{xy} = u_y + v_x - 2z w_{,xy} + w_{,x} w_{,y} \quad (74)$$

$$\text{so that } \epsilon_i = L_i^T d + \frac{1}{2} d^T H_i d \quad (75)$$

in which $i = 1, 2, 3$ and

$$d^T = (u_x, u_y, v_x, v_y, w_x, w_y, w_{,xx}, w_{,xy}, w_{,yy}) \quad (76)$$

$$L_i^T = \begin{pmatrix} 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \quad (77)$$

$$L_2^T = \begin{pmatrix} \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \quad (78)$$

$$L_3^T = \begin{pmatrix} \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \quad (79)$$

$$H_i = \begin{bmatrix} [H_i^{(m)}]_{6 \times 6} & [0]_{6 \times 3} \\ [0]_{3 \times 6} & [0]_{3 \times 3} \end{bmatrix} \quad (80)$$

in which (m) indicates the membrane matrix of the previous example.

The adequate matrices are

$$\hat{K}(I) = C_{ij} \int_{-1/2}^{1/2} L_i L_j^T dz = \begin{bmatrix} [\hat{K}_{(0)}^{(m)}]_{6 \times 6} & [0]_{6 \times 3} \\ [0]_{3 \times 6} & \frac{1}{12} \begin{bmatrix} C_{11} & C_{12} \\ C_{12} & C_{22} \\ \cdot & \cdot & 4C_{33} \end{bmatrix} \end{bmatrix} \quad (81)$$

$$\hat{N}_1(I) = \begin{bmatrix} [\hat{N}_{(0)}^{(m)}]_{6 \times 6} & [0]_{6 \times 3} \\ [0]_{3 \times 6} & [0]_{3 \times 3} \end{bmatrix}$$

in which (m) indicates the membrane matrices of the previous example.

The inadequate matrices are

$$\hat{N}_1^*(I) = \begin{bmatrix} [\hat{N}_{(0)}^{*(m)}]_{6 \times 6} & [0]_{6 \times 3} \\ [0]_{3 \times 6} & [0]_{3 \times 3} \end{bmatrix} \quad (82)$$

in which (m) indicates the membrane matrix.

Ref. 5 presents incremental matrices for the rectangular plate bending element with 16 out-of-plane degrees-of-freedom and bilinear in-plane displacement patterns. Note that the definition of the matrix elements in this reference, i.e.

$$\frac{\partial^2 n_{1e}}{2 \partial q_i \partial q_j} = \frac{\partial^2 U_3}{\partial q_i \partial q_j}; \quad \frac{1}{3} n_{2e} = \frac{\partial^2 U_4}{\partial q_i \partial q_j} \quad (83)$$

in which U_3 and U_4 = the components of strain energy cubically and quartically dependent on displacements, removes all ambiguity in the method of constructing these matrices. However, one must question the desirability of being forced to carry out this differentiation as a requisite to constructing the matrices.

SUMMARY AND CONCLUSIONS

Expressions for deriving incremental matrices to preserve the validity of Eqs. 1, 2, and 3 have been presented, which are applicable to any elastic system in which the strain components may be expressed as in Eq. 11. The matrices have been explicitly displayed for a truss element, in-plane flexural element, membrane type elements, and plate-flexure type elements. It has been pointed out that some investigators appear unaware of the restrictions necessary on the form of incremental stiffness matrices in order to ensure that the formalism of Eqs. 1, 2, and 3 is valid.

APPENDIX I.—EQUILIBRIUM EQUATIONS

The potential for an element has been written symbolically in Eq. 19. For equilibrium

$$\delta \Pi = \delta q^T \left\{ \frac{\partial \Pi}{\partial q_n} \right\} = 0 \quad (84)$$

For I_2 , expressed by Eq. 22b, we have

$$\delta I_2 = \delta \left[\frac{C_{ij}}{4} d^T (L_i d^T H_j + H_i d L_j^T) d \right] \quad (85a)$$

$$\begin{aligned} \delta I_2 = & \frac{C_{ij}}{4} \delta d^T (L_i d^T H_j + H_i d L_j^T) d + \frac{C_{ij}}{4} d^T (L_i d^T H_j + H_i d L_j^T) \delta d \\ & + \frac{C_{ij}}{4} d^T (L_i \delta d^T H_j + H_i \delta d L_j^T) d \quad (85b) \end{aligned}$$

$$= \frac{C_{ij}}{4} \delta d^T (2L_i d^T H_j + 4H_i d L_j^T) d \quad (85c)$$

in which the last equality follows from a consideration of the properties itemized after the presentation of Eq. 22c. However, the last form is not symmetric. It may be made symmetric by writing as

$$\delta I_2 = \frac{C_{ij}}{2} \delta d^T (L_i d^T H_j + L_j^T d H_i + H_i d L_j^T) d \quad (86)$$

Thus, the symmetric matrix term in Eq. 85a (which motivated the definition of Eq. 24) cannot be made to repeat in the equilibrium equations but gives rise to the form in Eq. 86 (which motivated the definition of N_1 in Eq. 23). If the algebra in Eqs. 85 is carried out for the symmetric form of Eq. 86, the matrix will repeat.

By algebra identical in nature to that in Eqs. 85, it is found that the definition for N_2 in Eq. 27 will repeat in the equilibrium equations but will not repeat in the incremental equilibrium equations. It does however give rise to a form which motivates the definition of N_3 in Eq. 26. If this symmetric form is now inserted in the potential energy expression the matrix repeats after the required manipulations.

APPENDIX II.—REFERENCES

1. Barsour, R. S., "A Finite Element Formulation for the General Stability Analysis of Thin-Walled Members," thesis presented to Cornell University, at Ithaca, N.Y., in 1970, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.
2. Gallagher, R. H., Lien, S., and Mau, S. T., "A Procedure for Finite Element Plate and Shell Pre- and Post-Buckling Analysis," Third Conference on Matrix Methods in Structural Mechanics, Wright-Patterson AFB, Ohio, Oct., 1971.
3. Mallett, R. H., and Marcal, P. V., "Finite Element Analysis of Nonlinear Structures," *Journal of the Structural Division, ASCE*, Vol. 94, No. ST9, Proc. Paper 6115, Sept., 1968, pp. 2081-2105.
4. Mallett, R. H., and Marcal, P. V., closure and errata to "Finite Element Analysis of Nonlinear Structures," *Journal of the Structural Division, ASCE*, Vol. 96, No. ST1, Proc. Paper 6990, Jan., 1970, pp. 132-133.
5. Yang, H. T. Y., "Flexible Plate Finite Element on Elastic Foundation," *Journal of the Structural Division, ASCE*, Vol. 96, No. ST10, Proc. Paper 7604, Oct., 1970, pp. 2083-2101.

APPENDIX III.—NOTATION

The following symbols are used in this paper:

- A = area of cross section;
 C_{ij} = linear elastic constitutive coefficients;
 D = matrix defining displacement gradients from displacement coordinates;
 d = vector of displacement gradients;
 E = modulus of elasticity;
 H_i = symmetric matrix defining nonlinear strains;
 I = moment of inertia;
 I_1, I_2, I_3 = integrands;
 L_i = vector defining linear strains from displacement gradients;
 l = length of one-dimensional element;
 K = assembled zero degree stiffness matrix;
 K = integrand for zero degree stiffness matrix;
 N_1, N_2 = assembled first and second degree stiffness matrices;
 N_1^*, N_2^* = integrands for first and second degree stiffness matrices;
 N_1^*, N_2^* = inappropriate integrands for stiffness matrices;
 $P, \Delta P$ = vector of generalized forces and its increment;

$q, \Delta q$ = vector of displacement coordinates and its increment;

U = strain energy;

u, v, w = local displacements;

x, y, z = local coordinates;

ϵ_i = strain component;

$\epsilon_i^L, \epsilon_i^{NL}$ = linear and nonlinear strain components;

ζ = nondimensional coordinate;

Π = total potential; and

σ_i = stress component.

INCREMENTAL FINITE ELEMENT MATRICES^b

Discussion by Carlos A. Felippa,^c A. M. ASCE

The authors should be commended for drawing attention to the computational pitfalls to which the naive application of Eqs. 1, 2, and 3 may lead the unwary investigator. However, the writer feels that the impact of the paper may be diluted by the informal mathematical derivations and the interweaving of much previously known material. The purpose of this discussion is to state the problem in precise mathematical terms.

The Fréchet-Taylor expansion of the elastostatic Lagrangian potential of Eq. 1 with argument $q + \lambda p$ (in which $p = \Delta q$ is a generalized coordinate variation vector; and $\lambda =$ a perturbation coefficient) may be written, assuming conservative generalized forces P :

$$\Pi(q + \lambda p) = \Pi(q) + \lambda p^T g + \frac{1}{2} \lambda^2 p^T \Delta g + 0(\lambda^3) \dots \dots \dots (87)$$

$$\Pi(q + \lambda p) = q^T(Aq - P) + \lambda p^T(Gq - P) + \frac{1}{2} \lambda^2 p^T(Sp - \Delta P)$$

$$+ 0(\lambda^3) \dots \dots \dots (88)$$

in which g = the energy gradient vector; G = the Jacobian matrix of Π ; and S = the Hessian matrix of Π (tangent stiffness matrix). The most general expression of the matrices in Eq. 88 for which A , G , and S are symmetric is

$$2A = K + \frac{\alpha}{2}(K_1 + K_1^T) + (1 - \alpha)K_1^* + \frac{\beta}{4}K_2 + \frac{1 - \beta}{4}K_2^* \dots \dots \dots (89a)$$

$$G = K + \frac{1}{2}(K_1 + K_1^T + K_1^*) + \frac{2 - \gamma + \beta\gamma}{4}K_2 + \frac{(1 - \beta)\gamma}{4}K_2^* \dots \dots \dots (89b)$$

$$S = K(K_1 + K_1^T + K_1^* + K_2 + \frac{1}{2}K_2^*) \dots \dots \dots (89c)$$

$$\text{in which } K = \int_V D^T C_{ij} L_i L_j^T D dV \dots \dots \dots (90a)$$

$$K_1 = \int_V D^T C_{ij} L_i d^T H_j D dV \dots \dots \dots (90b)$$

$$K_1^* = \int_V D^T C_{ij} (L_i^T d) H_j D dV \dots \dots \dots (90c)$$

$$K_2 = \int_V D^T C_{ij} H_i d^T H_j D dV \dots \dots \dots (90d)$$

$$K_2^* = \int_V D^T C_{ij} (d^T H_i d) H_j D dV \dots \dots \dots (90e)$$

and in which α , β , γ = arbitrary coefficients (if the restriction that G be symmetric is removed, two more arbitrary parameters appear). Note that the Hessian matrix, S , as well as the quadratic forms, $q^T A q$ and $p^T G p$, are independent of α , β , and γ .

The "repeatable" forms

$$\left. \begin{aligned} \dot{N}_1 &= K_1 + K_1^T + K_1^* \\ \dot{N}_2 &= K_2 + \frac{1}{2}K_2^* \end{aligned} \right\} \dots \dots \dots (91)$$

of Eqs. 23 and 26 are obtained if $\alpha = \beta = 2/3$ and $\gamma = 2$, as a trivial computation shows. However, there is no particular reason for selecting such parameters in the generation of A or G , or both. From a computational standpoint, other combinations not complying with Eqs. 1, 2, and 3 may be selected as well.

Note that the so-called "geometric" or "initial-stress" stiffness matrix

$$K_G = K_1^* + \frac{1}{2}K_2^* = \int_V D^T \sigma_i H_i D dV \dots \dots \dots (92)$$

repeats in Eqs. 1, 2, and 3 if $\alpha = (2\gamma - 1)/2\gamma$ and $\beta = (\gamma - 1)/\gamma$ for arbitrary $\gamma \neq 0$. The so-called "initial displacement" stiffness matrix

$$K_D = K_1 + K_1^T + K_2 \dots \dots \dots (93)$$

repeats if $\alpha = (\gamma + 1)/\gamma$ and $\beta = 2\alpha$ for arbitrary $\gamma \neq 0$. But no combination of α , β , γ exists for which both K_G and K_D repeat. The repetition of K_G and K_D would be of more interest than that of N_1 and N_2 whenever nonlinear stiffness evaluation subroutines are constructed to produce K , K_G , and K_D separately, as is often the case in practice.

APPENDIX.—NOTATION

The following symbols are used in this discussion:

A	=	metric matrix of strain energy potential form;
G	=	metric matrix of strain energy gradient form;
$g, \Delta g$	=	energy gradient vector and its increment;
K_1, K_1^*, K_2, K_2^*	=	matrices defined in text;
K_D	=	initial-displacement stiffness matrix;
K_G	=	geometric or initial-stress stiffness matrix;
p	=	unitary incremental displacement vector;
S	=	incremental stiffness matrix (Hessian matrix of total potential);
α, β, γ	=	arbitrary coefficients; and
λ	=	perturbation parameter scaling p .

^bDecember, 1973, by Sundaramoorthy Rajasekaran and David W. Murray (Proc. Paper 10226).

^cResearch Scientist, Struct. Mech. Lab., Lockheed Palo Alto Research Lab., Palo Alto, Calif.

Partitioned Solution Procedure for Control-Structure Interaction Simulations

K. C. Park*

University of Colorado, Boulder, Colorado 80309

and

W. Keith Belvin†

NASA Langley Research Center, Hampton, Virginia 23665.

A partitioned computational procedure is presented for the simulation of control-structure interaction systems by employing three modular software packages: a second-order structural dynamics analyzer, a second-order observer module, and a first-order stabilized active control force generator. This paper focuses on modular programming of the procedure, techniques for enhancing accuracy and for stabilizing the partitioned interaction equations, and time discretization of the stabilized systems of equations. A stability analysis has been performed using a set of model interaction equations, which indicates that the computational stability of the procedure is governed by the highest frequency of the controller and the strength of its position feedback parameter and not by that of the structural system. Comparison of the computational efficiency of the present procedure with a first-order conventional solution procedure indicates that the present procedure offers a substantial efficiency improvement. The effectiveness of the present procedure has been demonstrated by several example problems.

I. Introduction

REALISTIC analysis of active control-structure interaction (CSI) problems poses a formidable challenge both to the dynamist and the control scientist. The emerging need for real-time simulation and control of CSI systems will engender a new activity called "computational controls" to meet this challenge. The prevailing practice in CSI simulations is by and large limited to a reduced-order model of structures based on modal representations together with a limited number of sensors and actuators.¹⁻⁶ Specifically, the control laws are expressed in terms of the generalized coordinates and their time derivatives, which are computed using state estimators if full state feedback control is used. The resulting closed-loop system equations are usually cast in first-order form; this has been the prevailing practice since modern control theory has been almost exclusively developed for the first-order form.⁷⁻¹⁰

Typically, simulation tasks for CSI problems involve several computational elements and discipline-oriented models such as structural dynamics, control law synthesis, state estimation, actuator and sensor dynamics, thermal analysis, liquid sloshing and swirling, environmental disturbances, and maneuvering thrusts and torques. Because each of these computational elements can be large, it is usually not practical to assemble these computational elements into a single set of equations of motion and perform the analysis in its totality, which will be referred to as the "simultaneous solution approach." First, the equation size of the total system can be simply too large for many existing computers. Second, the solution of the coupled interaction equations may destroy the

sparsity of the attendant matrices, thus requiring excessive computations and storage space. Most important of all, any changes in the model or in the computational procedures will affect many of the required analysis software modules and hence require a painstaking software verification effort.

The computational procedure that is described in the present paper has been motivated to alleviate the aforementioned difficulties that exist in the simultaneous solution approach. First, software development of any new capability is costly and time consuming; thus, if at all possible, it is preferable to use existing single-field analysis modules to conduct the coupled-field interaction analysis. Second, the tasks for model generation and methods development of each field are best accomplished by relying on the experts of each single-field discipline. To accommodate both the software considerations and the single-field expertise, a partitioned (or divide-and-conquer) analysis procedure is proposed for control-structure interaction analysis for direct output feedback systems.^{11,12} The procedure abandons the conventional way of treating the CSI problems as one entity. Instead, it treats the structure (or plant), the observer, and the controller/observer interaction terms as separate entities. Thus, the CSI problem is recognized as a coupled-field problem, and a divide-and-conquer strategy is adopted for the development of a real-time computational procedure. A similar concept has been successfully applied to other interaction analyses such as fluid-structure interactions,¹³ multistructural interaction systems,¹⁴ Earth dam and pore-fluid interactions,¹⁵ and multi-body systems with constraints.¹⁶

The proposed partitioned analysis procedure hinges on three software and computational aspects. First, because large-scale simulation of interdisciplinary problems must rely on an efficient and versatile data management system, the data manager is used to handle the necessary interprocessor communications among the single-discipline analysis modules. Second, at each discrete time increment, the equations of motion for each discipline are solved separately by considering the interaction terms as external disturbances or applied forces. Third, when necessary, computational stabilization and accuracy improvements are introduced through augmentations and/or equation modifications. It is important to note that such partitioned solutions of each discipline equations can be carried out either on a sequential or parallel machine

Received Feb. 3, 1989, presented as Paper 89-1238 at the 30th Structures, Structural Dynamics, and Materials Conference, Mobile, AL, April 3-5, 1989; revision received Sept. 18, 1989. Copyright © 1989 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

*Professor of Aerospace Engineering, Center for Space Structures and Control. Member AIAA.

†Aerospace Engineer, Spacecraft Dynamics Branch. Member AIAA.

if certain message passing and memory-conflict issues are handled appropriately.

Recently, the computational need and the physical insight that can be accrued from the second-order dynamic equations have motivated several investigators in the control community to address the second-order state estimation and control law issues. They include the analysis of the second-order system equations^{17,18} and CSI design,¹⁹⁻²⁶ among others.

The objective of the paper is thus to describe the algorithmic nature of the partitioned CSI solution procedure, its implementation aspects and computational stability and accuracy characteristics. To this end, the paper is organized as follows. The discrete equations of motion for CSI problems are presented in Sec. II, which includes the discrete equations of motion for structures subjected to control forces, an observer model, and a general control law.

A review of a conventional first-order-based solution procedure is presented in Sec. III, which is termed herein simultaneous solution procedure to contrast with the proposed partitioned procedure. Section IV introduces a parabolic stabilization in order to derive a differential equation for the interaction terms (i.e., the control force and the state estimation error term) so that the interaction terms are obtained by solving differential equations rather than by back substitutions. It turns out that such a stabilization (or an equation augmentation) improves not only computational stability but numerical accuracy as well.

Time discretization of the coupled CSI equations is carried out in Sec. V by employing the implicit midpoint formula (or the trapezoidal rule). It is shown that a general canonical form of the equations of motion for structures and the second-order form of state estimators leads to a computationally attractive discretization. A computational stability analysis of the present partitioned CSI solution procedure is elaborated in Sec. VI, which indicates that the stability of the present partitioned procedure is governed by the highest frequency of the control force and its position feedback strength parameter irrespective of the frequency magnitude of the structural system.

The computational efficiency of the present procedure is compared with the conventional procedure in Sec. VII. It is shown that the present procedure offers a substantial efficiency improvement over the conventional procedure for most practical CSI problems. Applications of the present procedure are given by way of several example analyses; results are offered in Sec. VIII. Finally, concluding remarks and further remaining challenges toward making the real-time CSI analysis and simulations a routine practice are discussed in Sec. IX.

II. Equations of Motion for Control-Structure Interaction Systems

A typical control-structure interaction system can be represented as shown in Fig. 1 (see, e.g., Kwakernaak and Sivan⁹). The discrete equations of motion for control-structure interaction systems may be described by

Structure:

$$M\ddot{q} + D\dot{q} + Kq = f + Bu + Gw \quad (1a)$$

$$q(0) = q_0, \quad \dot{q}(0) = \dot{q}_0 \quad (1a)$$

Sensor output:

$$z = Hx + v \quad (1b)$$

Estimator:

$$\dot{\tilde{x}} = A\tilde{x} + Ef + Bu + L\gamma, \quad \tilde{x}(0) = 0 \quad (1c)$$

Control force:

$$u = -F\tilde{x} \quad (1d)$$

Estimation error:

$$\gamma = z - (H_d\tilde{q} + H_v\dot{\tilde{q}}) \quad (1e)$$

where

$$x = \begin{Bmatrix} q \\ \dot{q} \end{Bmatrix}, \quad \tilde{x} = \begin{Bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{Bmatrix}$$

and

$$H = [H_d \ H_v], \quad L = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix}, \quad E = \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ M^{-1}B \end{bmatrix}, \quad F = [F_1 \ F_2]$$

In the preceding equations, M is the mass matrix, D is the damping matrix, K is the stiffness matrix, $f(t)$ is the applied force, B is the actuator location matrix, G is the disturbance location matrix, q is the generalized displacement vector, w is a disturbance vector, and the superscript dot denotes time differentiation. In Eq. (1b), z is the measured sensor output. The matrix H_d is the matrix of displacement sensor locations and H_v is the matrix of velocity sensor locations. The vector v is measurement noise. The state estimator in Eq. (1c) is assumed either to be based on the Kalman filter^{7,8} or based on a Luenberger observer²⁷ if the system is deterministic. The superscript $\sim$ denotes the estimated states. The actuator output u is a function of the state estimator variables $\tilde{q}$ and $\dot{\tilde{q}}$, and F_1 and F_2 are control gains determined for example by pole-zero placement or from the solution of an optimal control problem. The observer is governed by L , the filter gain matrix. For the special case where L_1 is the null matrix (i.e., $\tilde{q} = \dot{\tilde{q}}$), a second-order state estimator can be expressed as

$$M\ddot{\tilde{q}} + D\dot{\tilde{q}} + K\tilde{q} = f + Bu + ML_2\gamma \quad (2)$$

The effect of the preceding simplification on the observer stability and convergence is discussed in detail in Refs. 26 and 28.

III. Simultaneous Solution Approach

It should be emphasized that the objective of this paper is to directly solve the coupled equations given in Eqs. (1) in their natural forms. That is, to exploit the second-order form of the structure equation (and also the observer equation if $L_1 = 0$). To justify this objective, the simultaneous solution approach for numerical simulation of Eq. (1) is examined first.

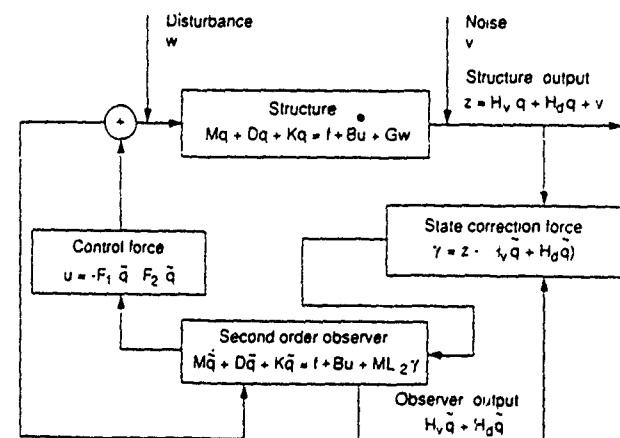


Fig. 1 Typical control/structure interaction system.

The numerical solution of Eqs. (1) by the simultaneous solution approach begins with appropriate initial conditions: the feedback gain F and the filter gain L . The structure equation is written in first-order form

$$\dot{x} = Ax + Ef + \bar{B}u + \bar{G}w \quad (3)$$

where

$$\bar{G} = \begin{bmatrix} 0 \\ M^{-1}G \end{bmatrix}$$

The control gains and observer gains can be synthesized independently by noting that the stability of the structural system and the observer error stability are uncoupled. Introducing the error equation by the deterministic form of Eqs. (1) as

$$\bar{e} = x - \hat{x} = \begin{bmatrix} q - \hat{q} \\ \dot{q} - \hat{\dot{q}} \end{bmatrix} \quad (4)$$

and eliminating u yields

$$\begin{bmatrix} \dot{\bar{e}} \\ \bar{e} \end{bmatrix} = \begin{bmatrix} A - \bar{B}F & \bar{B}F \\ 0 & A - LH \end{bmatrix} \begin{bmatrix} x \\ \bar{e} \end{bmatrix} + \begin{bmatrix} E \\ 0 \end{bmatrix} f + \begin{bmatrix} \bar{G} \\ 0 \end{bmatrix} w \quad (5)$$

The stability of Eq. (5) is governed by the stability of $[A - \bar{B}F]$ and $[A - LH]$. Thus, the control gain F is suitably chosen from the matrix $[A - \bar{B}F]$ and the observer gain L from the matrix $[A - LH]$.

Subsequently, the simultaneous solution approach eliminates u and z from Eqs. (1a) and (1c) and then solves the observer-based closed-loop equations:

$$\begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} = \begin{bmatrix} A & -\bar{B}F \\ LH & A - \bar{B}F - LH \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \begin{bmatrix} E \\ E \end{bmatrix} f + \begin{bmatrix} \bar{G} \\ 0 \end{bmatrix} w \quad (6)$$

The embedding effects of both the controller and the state observer result in an unsymmetric and nonsparse system matrix of dimension $(4N \times 4N)$, where N is the number of structural degrees of freedom. Solution of Eq. (6) would require considerable software modification of existing structural dynamics analysis programs for large-scale CSI simulation purposes. In addition to losing the computational advantages associated with the finite element based CSI equation, the simultaneous solution approach requires the control law to be embedded into the observer model. If the control law includes actuator, sensor, and/or controller dynamics, additional states must be added to the observer. This greatly complicates the observer model and requires significant software development for each class of control law dynamics. The difficulties associated with the simultaneous solution approach have prompted development of a partitioned solution approach for the CSI equations as described in the next section.

IV. Partitioned Solution Procedure

To mitigate the software and algorithmic difficulties associated with the asymmetric embedding of the controller and the state observer into the closed-loop equations as discussed in the previous section, a preliminary partitioned solution procedure¹² was proposed for the solution of direct output feedback systems, but no stability analysis was offered therein. The present exposition extends the basic concepts offered in Ref. 12 to include the case of dynamic compensators in a form of computationally advantageous second-order observers. However, for completeness, the first-order observer solution procedure is included in the Appendix.

In essence, the proposed partitioned solution procedure numerically integrates the structural equations of motion (1a) and the observer equation (1c) by treating the control force u and the estimation error γ as if they are applied terms in the right-hand sides. In this way, simulation of control-structure

interaction systems using the proposed partitioned solution procedure to be described can be carried out by a judicious employment of three software modules: the structural analyzer to obtain q , the state estimator to obtain $\hat{q}$, and the solver for the control force u and the state estimation error γ as indicated in Fig. 2. Thus, the partitioned procedure becomes computationally efficient and can preserve software modularity by exploiting the symmetric matrix form in the left-hand sides of Eqs. (1a) and (1c).

However, computations of the control force u and the state estimation error γ by Eqs. (1d) and (1e), respectively, can lead not only to an accumulation of errors but often can give rise to numerical instability. Hence, to make the proposed partitioned solution procedure robust, it is imperative to stabilize the partitioned solution process and/or numerically to filter the solution errors in computing u and γ . This is addressed in the next section.

Stabilization for Computations of Control Force and Estimation Error

To appreciate the nature of error accumulations in computing u and γ by employing Eqs. (1d) and (1e), respectively, let us consider the following discrete predictors:

$$u^{n+1/2} \approx u^n = -(F_1 \hat{q}^n + F_2 \ddot{q}^n) \quad (7a)$$

$$\gamma^{n+1/2} \approx \gamma^n = z^n - (H_d \hat{q}^n + H_v \dot{\hat{q}}^n) \quad (7b)$$

where the superscripts designate the discrete time intervals, $t^n = nh$ and $t^{n+1/2} = n + 1/2h$ with a constant stepsize of h .

It is observed that the predicted control force $u^{n+1/2}$ will possess errors in $(\hat{q}^{n+1/2} - \hat{q}^n)$ and $(\ddot{q}^{n+1/2} - \ddot{q}^n)$ magnified by F_1 and F_2 , unless the control gains have a built-in stabilizing filter against these errors. The computation of the estimation error $\gamma^{n+1/2}$ will have a similar error accumulation unless the filter gain L is designed to cope with the prediction errors. Among several possible stabilization strategies, we will employ a parabolic stabilization as follows.

First, we time differentiate Eq. (1c) to obtain

$$u = -F_1 \dot{\hat{q}} - F_2 \ddot{\hat{q}} \quad (8)$$

Substituting $\ddot{\hat{q}}$ from Eq. (2) into the preceding equation, one obtains

$$\dot{u} + F_2 M^{-1} B u = -F_2 (M^{-1} \dot{p} + L_2 \gamma) - F_1 \dot{\hat{q}} \quad (9)$$

where the generalized rate of momentum $\dot{p}$ is given by

$$\dot{p} = (f - D \dot{\hat{q}} - K \hat{q}) \quad (10)$$

The parabolic stabilization that led to Eq. (9) for computing the control law is sometimes called an equation augmentation procedure as it has not altered any of the basic governing of Eqs. (1) except one time differentiation of u assuming $\dot{u}$ exists. However the assumption is later removed through time discretization as will be shown later in the paper.

It is noted that the homogeneous part of Eq. (9) has the filtering effect of the form $(sI + F_2 M^{-1} B)^{-1}$ in parlance of classical control theory where s is the Laplace transform operator, thus achieving the required stabilization. From the computational viewpoint, although $F_2 M^{-1} B$ is in general a full matrix, its size is relatively small as the size of u is proportional to the number of actuators placed on the structure.

Similarly, for the observer estimation error γ , one can stabilize its computation by adopting the following augmented equation:

$$\dot{\gamma} + H_v L_2 \gamma = \dot{z} - H_v M^{-1} (\dot{p} + B u) - H_d \dot{\hat{q}} \quad (11)$$

Again the matrix $H_v L_2$ is in general not sparse; however its

size is only proportional to the number of measurements output from the structure.

Stabilized Partitioned Equations and Solution Process

The adoption of the second-order observer and the preceding stabilization thus replace Eqs. (1c), (1d), and (1e) by Eqs. (2), (9), and (11), respectively, as summarized here.

Structure:

$$\begin{aligned} M\ddot{q} + D\dot{q} + Kq &= f + Bu + Gw \\ q(0) &= q_0, \quad \dot{q}(0) = \dot{q}_0 \end{aligned} \quad (12a)$$

Sensor output:

$$z = Hx + v \quad (12b)$$

Estimator:

$$\begin{aligned} M\ddot{\tilde{q}} + D\dot{\tilde{q}} + K\tilde{q} &= f + Bu + ML_2\gamma \\ \tilde{q}(0) &= 0, \quad \dot{\tilde{q}}(0) = 0 \end{aligned} \quad (12c)$$

Control force:

$$\ddot{u} + F_2 M^{-1} Bu = -F_2 (M^{-1} \ddot{p} + L_2 \gamma) - F_1 \dot{\tilde{q}} \quad (12d)$$

Estimation error:

$$\dot{\gamma} + H_0 L_2 \gamma = \dot{z} - H_0 M^{-1} (\ddot{p} + Bu) - H_d \ddot{\tilde{q}} \quad (12e)$$

The partitioned solution process that implements the preceding equation can be summarized by the following steps:

- 1) Obtain $\tilde{q}^{n+1/2} = \tilde{q}^n + \delta \dot{\tilde{q}}^n$ and $\dot{\tilde{q}}^{n+1/2} = \dot{\tilde{q}}^n$.
- 2) Solve for $u^{n+1/2}$ and $\gamma^{n+1/2}$ from Eqs. (12d) and (12e).
- 3) Solve for $\tilde{q}^{n+1/2}$, $\dot{\tilde{q}}^{n+1}$, and $\ddot{\tilde{q}}^{n+1}$ from Eq. (12c).
- 4) Solve for $q^{n+1/2}$, $\dot{q}^{n+1}$, and $\ddot{q}^{n+1}$ from Eq. (12a).
- 5) Increment ($n = n + 1$) and (time = time + h).

A flowchart describing the preceding computational steps is depicted by Fig. 2. These computational steps are implemented into three modules as shown in Fig. 2. Note that steps 1-5 would be repeated until the transient response simulation is complete. (Note, if no observer is used, step 2 would only require calculation of $u^{n+1/2}$ and step 3 would be omitted.) The methods for calculating the control and observer interaction forces differ depending on the type of simulation required. This aspect together with the time discretization, prediction of the state estimation vectors, and computational details for implementing the present partitioned solution procedure will be dealt with in the next section.

V. Time Discretization of Control-Structure Interaction Equations

Direct time integration of the structural equations (12a) and the observer equations (12c) can be performed by a variety of techniques. A theoretically exact solution can be obtained using a matrix exponential approach. However, such an approach engenders numerical roundoff when applied to large-dimensioned problems and involves fully populated matrices that can significantly limit the number of simultaneous equations which can be solved. A number of approximate techniques for numerical integration of coupled ordinary differential equations also exist. The approximate methods generally interpolate and/or extrapolate the dynamic states with implicit and/or explicit time-stepping formulas. By preserving the sparsity and symmetry of the coefficient matrices, the approximate methods can be applied to very large systems of simultaneous equations. Thus approximate numerical integration formulas are used in this paper.

Approximate methods must be examined from both stability and accuracy considerations. Numerical stability of explicit integration formulas require a step size inversely

proportional to the highest frequency of the system. The small step size necessary for explicit integration stability generally satisfies the accuracy requirements. Conversely, some fully implicit formulas are unconditionally stable; thus the step size must be chosen based on accuracy considerations. Because larger step sizes permit more efficient solution of the control-structure interaction equations, implicit integration formulas are employed in this study.

It should be mentioned that, when massively parallel computations become widely available, the programming simplicity of explicit formulas become so attractive that one may prefer explicit to implicit formulas for control-structure interaction simulations. This is particularly true when reduced-order models are used that filter high-frequency responses.

Among a plethora of implicit numerical integration formulas, we employ the following unconditionally stable midpoint implicit formulas:

$$q^{n+1/2} = q^n + \delta \dot{q}^{n+1/2}, \quad \delta = h/2 \quad (13a)$$

$$\dot{q}^{n+1/2} = \dot{q}^n + \delta \ddot{q}^{n+1/2} \quad (13b)$$

$$q^{n+1} = 2q^{n+1/2} - q^n \quad (13c)$$

where h is the time-step size. The selection of the midpoint implicit formula (or the trapezoidal rule) for the present application is due to its minimal frequency distortion and no numerical damping characteristics.²⁹⁻³¹ Experience has shown that it is very important to minimize numerical damping when studying the control-structure interactions.

Integration of Structural Equations

Implicit time discretization of Eq. (12a) by Eq. (13) yields the following difference equation for the structure:

$$Sq^{n+1/2} = g^{n+1/2} \quad (14a)$$

$$S = M + \delta D + \delta^2 K \quad (14b)$$

$$\begin{aligned} g^{n+1/2} &= \delta^2 (f^{n+1/2} + Bu^{n+1/2}) + M(q^n + \delta \dot{q}^n) + \delta Dq^n \\ &\quad + \delta^2 Gw^{n+1/2} \end{aligned} \quad (14c)$$

$$q^{n+1} = 2q^{n+1/2} - q^n \quad (14d)$$

$$\dot{q}^{n+1/2} = (q^{n+1/2} - q^n)/\delta, \quad \dot{q}^{n+1} = 2\dot{q}^{n+1/2} - \dot{q}^n \quad (14e)$$

It is observed that the only unknown in the right-hand side of the preceding difference equation is $u^{n+1/2}$, which is treated as an applied forcing term in solving for the states q^{n+1} and $\dot{q}^{n+1}$, thus, leading to a modular implementation as a unique feature in the present partitioned solution procedure.

Integration of Observer Equations

Time discretization of Eq. (12c) yields the following equations for the observer:

$$S\tilde{q}^{n+1/2} = \tilde{g}^{n+1/2} \quad (15a)$$

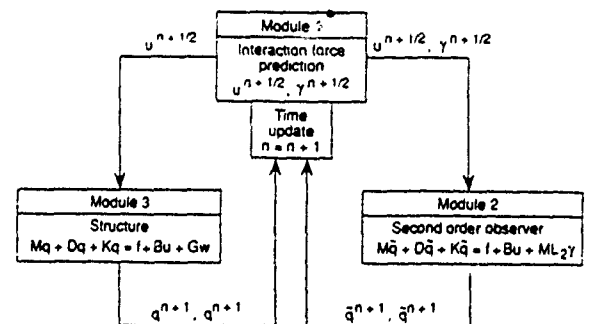


Fig. 2 Flow chart of partitioned solution procedure.

$$S = M + \delta D + \delta^2 K \quad (15b)$$

$$\begin{aligned} \bar{g}^{n+1/2} = & \delta^2 (\gamma^{n+1/2} + B u^{n+1/2} + M L_2 \gamma^{n+1/2}) \\ & + M(\bar{q}^n + \delta \dot{\bar{q}}^n) + \delta D \bar{q}^n \end{aligned} \quad (15c)$$

$$\bar{q}^{n+1} = 2\bar{q}^{n+1/2} - \bar{q}^n \quad (15d)$$

$$\dot{\bar{q}}^{n+1/2} = (\bar{q}^{n+1/2} - \bar{q}^n)/\delta, \quad \dot{\bar{q}}^{n+1} = 2\dot{\bar{q}}^{n+1/2} - \dot{\bar{q}}^n \quad (15e)$$

From Eqs. (15) it is seen that $u^{n+1/2}$ and $\gamma^{n+1/2}$ are required to numerically solve for the states $\bar{q}^{n+1}$ and $\dot{\bar{q}}^{n+1}$. As in the case of the structural analyzer of Eqs. (14), these two unknown vectors are treated as applied disturbance terms so that the observer states can be implemented into a modular package. Note that both in Eqs. (14) and (15) the attendant matrix S possesses the same sparsity and symmetry that exists in the mass, damping, and stiffness matrices. Hence, linear equation solution techniques that exploit symmetry and sparsity may be used advantageously. Such a solution capability is widely available in existing structural analysis software systems.

Integration of Control Force and State Estimation Error Equations

In advancing the time marching for the structural and the state estimation equations, it is recalled that we have assumed that both $u^{n+1/2}$ and $\gamma^{n+1/2}$ are already available from the solution module of the control force and state estimation error equations. We will show in this subsection that $u^{n+1/2}$ and $\gamma^{n+1/2}$ can be obtained by predicting only the state estimation vector $\bar{q}^{n+1/2}$.

In doing so, unlike the structural and observer equations, we have decided to time discretize both the control force and state estimation error equations as a coupled set of equations. Carrying out the necessary discretization by the midpoint implicit formula [Eqs. (13)] yields

$$\bar{S} r^{n+1/2} = \bar{g}^{n+1/2} \quad (16a)$$

$$\bar{S} = \begin{bmatrix} I + \delta F_2 M^{-1} B & \delta F_2 L_2 \\ \delta H_v M^{-1} B & I + \delta H_v' \end{bmatrix} \quad (16b)$$

$$\bar{g}^{n+1/2} = \begin{Bmatrix} 0 \\ z^{n+1/2} \end{Bmatrix} - \begin{Bmatrix} F_1 \\ H_d \end{Bmatrix} \bar{q}^{n+1/2} - \begin{Bmatrix} F_2 \\ H_v \end{Bmatrix} (\bar{q}^n + M^{-1} \dot{\bar{p}}^{n+1/2}) \quad (16c)$$

$$r^{n+1/2} = \begin{Bmatrix} u^{n+1/2} \\ \gamma^{n+1/2} \end{Bmatrix} \quad (16d)$$

It is noted that $\bar{S}$ is not sparse; thus the partitioned solution approach is mainly oriented toward systems where the number of states ($2N$) is more than the number of actuators m and measurements r (i.e., $r + m < 2N$). The solution of Eqs. (16) at the half time step makes both $u^{n+1/2}$ and $\gamma^{n+1/2}$, thus permitting the solution of both the structural displacement and the state estimation vectors with the control and state estimation error terms on the right-hand sides of Eqs. (14) and (15). Since a prediction step is involved, one typically iterates to achieve a converged solution for $\bar{q}^{n+1/2}$. However, as will be shown in Sec. VIII, the stabilization procedure used to obtain Eqs. (9) and (11) [or Eqs. (12d) and (12e)] produces very accurate results without iterating to obtain $\bar{q}^{n+1/2} = \bar{q}_p^{n+1/2}$ in most instances. To solve for $u^{n+1/2}$ when an observer is used, it is necessary to predict $\bar{q}^{n+1/2}$

$$\bar{q}_p^{n+1/2} = \bar{q}^n + \delta \dot{\bar{q}}^n \quad (17)$$

and use $z^{n+1/2}$ to solve Eq. (16) for $u^{n+1/2}$ and $\gamma^{n+1/2}$. (Note that $z^{n+1/2}$ implies measuring the system output at the half time step. If this is not possible, using $z_p^{n+1/2} = z^n$ will produce a phase shift in the observer state of δ in time.)

VI. Stability and Accuracy of Partitioned Solution Procedure

Computational stability analysis of partitioned procedures for a general coupled system is still in an evolving stage as discussed in Refs. 14 and 32. Hence, the analysis herein applies the relevant results from Refs. 14 and 32 in the present stability analysis of the partitioned CSI solution procedure. The partitioned CSI solution procedure presented in Eqs. (14–16), although discretized by implicit time integration formulas, may suffer from computational instability as it involves extrapolations to obtain $u^{n+1/2}$ and $\gamma^{n+1/2}$. A complete stability analysis of the partitioned solution procedure for the coupled structural dynamics, observer, and controller equations is difficult to perform unless the observer characteristics H, L and the controller characteristics B, F are specified. Hence, the analysis that follows is restricted to an ideal observer, i.e., $\gamma = 0$ so that only Eq. (14) and Eq. (16a) are considered.

Stability Analysis of Model Control-Structure Interaction

To assess the computational stability of the present partitioned solution procedure, we construct a model single-degree-of-freedom interaction equation as follows. First, neglecting structural damping, a modal structural equation of motion can be expressed as

$$\ddot{y} + \omega^2 y = -u \quad (18)$$

where y is a generalized coordinate and ω is its associated frequency.

Second, the model controller is assumed to consist of both the position and velocity feedback with appropriate weights given by

$$u = \eta \omega_c^2 y + \zeta \omega_c \dot{y}, \quad \omega_{\min} \leq \omega_c \leq \omega_{\max} \quad (19)$$

where ω_c is the feedback frequency that ranges from the minimum to the maximum of the structural frequency contents, and η and ζ are positive scalar coefficients that signify the strength of the position and the velocity feedback, respectively.

Combining Eq. (18) with the stabilized form of Eq. (19), we have the model interaction equation

$$\ddot{y} + \omega^2 y = -u \quad (20a)$$

$$\dot{u} + \zeta \omega_c u = \eta \omega_c^2 \dot{y} - \zeta \omega_c \omega^2 y \quad (20b)$$

Thus, the model interaction equations given by Eqs. (20) represent the case of full-state feedback. They do not, however, reflect the mode-to-mode coupling that can occur in reduced-order feedback controller. Nevertheless, an analysis of the computational stability using the preceding model interaction equations should shed insight on the overall stability of the present partitioned solution procedure.

Application of the partitioned CSI solution procedure [Eqs. (14–16)] with $\gamma = 0$ to solve the preceding model equations yields

$$y_p^{n+1/2} = y^n + \delta \dot{y}^n \quad (21a)$$

$$(1 + \delta \zeta \omega_c) u_p^{n+1/2} = (\eta \omega_c^2 - \delta \zeta \omega_c \omega^2) y_p^{n+1/2} + \zeta \omega_c \dot{y}^n \quad (21b)$$

$$(1 + \delta^2 \omega^2) y^{n+1/2} = -\delta^2 u_p^{n+1/2} + y^n + \delta \dot{y}^n \quad (21c)$$

$$y^{n+1} = 2y^{n+1/2} - y^n, \quad \dot{y}^{n+1/2} = (y^{n+1/2} - y^n)/\delta$$

$$\dot{y}^{n+1} = 2\dot{y}^{n+1/2} - \dot{y}^n \quad (21d)$$

$$u^{n+1} = \eta \omega_c^2 y^{n+1} + \zeta \omega_c \dot{y}^{n+1} \quad (21e)$$

Computational stability of the model form of CSI partitioned equations (20) can be assessed by seeking a nontrivial solu-

tion of the preceding difference equation (21) by

$$\begin{Bmatrix} u^{n+1} \\ y^{n+1} \end{Bmatrix} = \lambda \begin{Bmatrix} u^n \\ y^n \end{Bmatrix} \quad (22)$$

such that

$$|\lambda| \leq 1 \quad (23)$$

for stability.

Substituting Eq. (22) into Eqs. (21) and eliminating y , one obtains

$$J \begin{Bmatrix} u \\ y \end{Bmatrix}^n = 0 \quad (24)$$

where

$$J = \begin{bmatrix} \delta(1 + \delta\zeta\omega_c) \cdot (\lambda + 1)^2 & 4\lambda(\delta^2\zeta\omega_c\omega^2 - \delta\eta\omega_c^2) + 2\zeta\omega_c(1 + \lambda) \\ \delta^2(\lambda + 1)^2 & (1 + \delta^2\omega^2) \cdot (\lambda + 1)^2 - 4\lambda \end{bmatrix}$$

To test the stability requirement of Eq. (23) on the characteristic equation, i.e., $\det J = 0$, one transforms $|\lambda| \leq 1$ into the entire left-hand plane of the z plane by

$$\lambda = \frac{1+z}{1-z}, \quad |\lambda| \leq 1 \Leftrightarrow \operatorname{Re}(z) \leq 0 \quad (25)$$

Carrying out the necessary algebra we have from $\det J(z) = 0$ the following z -polynomial equation:

$$(\delta^3\zeta\omega_c\omega^2 - \delta^2\eta\omega_c^2 + 1)z^2 + (\delta\zeta\omega_c)z + \delta^2(\eta\omega_c^2 + \omega^2) = 0 \quad (26)$$

A test of the polynomial equation (26) for possible positive real roots by the Routh-Hurwitz criterion (see, e.g., Gantmacher³³) indicates that the partitioned procedure as applied to the model coupled equations (18) and (19) gives a stable solution provided

$$(\delta^3\zeta\omega_c\omega^2 - \delta^2\eta\omega_c^2 + 1) \geq 0 \quad (27)$$

Note that if there is no position feedback (i.e., $\eta = 0$), the model interaction equations solved by the present partitioned solution procedure [Eqs. (14-16)] yields unconditionally stable solutions as Eq. (27) is automatically satisfied. Hence, a more critical stability assessment can be made by assuming no velocity feedback (i.e., $\zeta = 0$) for which we have stability from Eq. (27)

$$h \leq 2/\sqrt{\eta\omega_c} \quad (28)$$

The preceding stability analysis on the model interaction equations permits us to make the following observations. First, Eq. (28) indicates that feedback frequency ω_c and the strength of the position feedback η dictate the computational stability and not the structural frequency ω . In other words, the position feedback dictates the allowable step size for stability. Thus, the highest frequency of the controller governs stability, not the highest frequency of the structure. Since most controllers are designed with reduced-order structure models that ignore high-frequency dynamics, the present solution procedure is not unduly restricted by stability.

Second, if velocity feedback is present, the allowable step size for stability increases until $\zeta \geq \sqrt{4\eta^3/27}$ at which point the solution becomes unconditionally stable.

It should be noted that, instead of the stabilized form of control force [Eq. (12d) or (20b)], if the scalar form of Eq. (7a) is used in the preceding stability analysis, the resulting stability limit is given by

$$h \leq \min(2/\zeta\omega_c, 2\zeta/\eta\omega_c) \quad (29)$$

Assuming $\zeta \ll 1$, the first term in the preceding condition allows a sufficiently large step size. However, since $\zeta/\eta \approx 1$ for a balanced control law, it imposes a step size restriction $h \approx 2/\omega_c$, which approaches the limit imposed by a typical explicit integration formula. This proves the advantage of the present stabilized partitioned solution equation (12) solely from the computational stability viewpoint.

Although not elaborated herein, a stability analysis that includes an observer model and the state estimation error equation has been conducted with the following parameter choices:

$$L_2 = [l_{21} \quad l_{22}], \quad H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (30)$$

in conjunction with the structural model already used in Eq.

(20). The analysis result yields the following step size restriction

$$h \leq \min(2/\zeta\omega_c, 2/\sqrt{\eta\omega_c}, 2/\sqrt{l_{21}}) \quad (31)$$

It should be noted that l_{21} corresponds to the Kalman filter gain magnitude, which can be adjusted to be sufficiently small compared with ω^2 as can be assessed from Eq. (5b). Hence, provided $l_{21} < \omega_c$, the condition given by Eq. (28) is seen to govern the maximum stable step size by the present partitioned solution procedure.

For the general multidimensional case governed by Eqs. (14-16), one observes that the stiffness proportional control force in practice reaches only a fraction of the total internal force ($u = \eta Kq$, $\eta \ll 1$). Hence, even for a distributed stiffness proportional control configuration where $\omega_c \rightarrow \omega_{\max}$, the stable step size given by Eq. (28) should be much larger than the maximum stable step size of a typical explicit integration algorithm (say, $h_{\max} \leq 2/\omega_{\max}$). Therefore, the computational efficiency of the present partitioned solution procedure is established.

Accuracy Assessment

In addition to the stability consideration, the step size must also be chosen based on accuracy considerations. Although the midpoint implicit integration formulas produce no artificial damping, there does exist a frequency distortion. The apparent frequency to true frequency ratio F_a is given by

$$F_a = \frac{N_s}{2\pi} \tan^{-1} \left(\frac{2\pi}{N_s[1 - (\pi^2/N_s^2)]} \right) \quad (32)$$

where N_s is the number of steps per cycle, $N_s = (2\pi)/(h\omega)$. Figure 3 shows the frequency distortion as a function of the number of steps per cycle. There must be 18 steps per cycle to produce less than 1% frequency error. Thus, the step size based on 1% frequency error is

$$h \leq 2\pi/N_s\omega = 0.349/\omega \quad (33)$$

Comparing Eq. (28) to Eq. (33) one finds $\eta \geq 32.84$ before stability governs the step size and not the 1% allowable frequency distortion. Thus, for most practical considerations, accuracy of the present integration formulas rather than stability govern the selection of the integration step size h .

VII. Computational Efficiency

Simulation of the CSI equations in second-order form is prompted in part by potential increases in computational

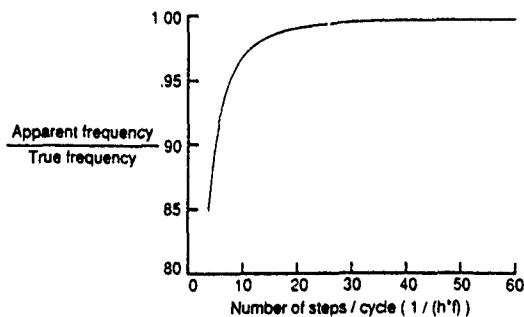


Fig. 3 Frequency distortion using midpoint implicit integration.

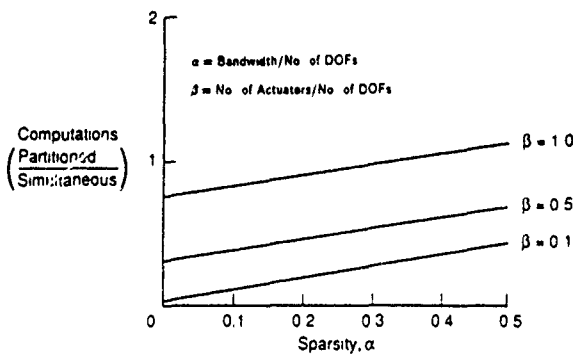


Fig. 4 Computational efficiency of partitioned solution procedure.

efficiency. The computational benefits of matrix sparsity and symmetry on the left-hand side (LHS) of the second-order finite element based equations are reduced by the additional computations needed to maintain the control and state estimation error terms on the right-hand side (RHS) of the equations. Therefore, we will assess the relative efficiency of the simultaneous solution procedure of Eq. (6) and the partitioned solution procedure of Eqs. (14-16). For simplicity, the computational efficiency is estimated based on an autonomous, deterministic system, $f = w = v = 0$. The number of floating point operations (FLOP) are approximated to an accuracy of order N , where N is the number of degrees of freedom.

The simultaneous solution procedure given by Eq. (6) may be solved either by a set of first-order differential equation solvers or by a matrix exponential approach. Since the size of the solution matrix is $4N \times 4N$, in addition to factoring of the solution matrix once at the beginning of the computation for time-invariant cases, a total of $16N^2$ storage locations $16N^2$ FLOP are required at each time step.

When one adopts the partitioned solution procedure [Eqs. (14-19)] as S is a sparse and symmetric matrix, considerable storage and computational savings can be attained. If the matrix element bandwidth of S is proportional to N by α

$$\text{bandwidth} = \alpha N; \quad 0 \leq \alpha \ll N$$

then, only αN^2 storage locations are needed. An $L-U$ decomposition of the S matrix (which requires $[\alpha^2 N^3 + 3\alpha N^2]/2 + O(N)$ FLOP) is performed once for time-invariant systems. Then, at each time step, the RHS is calculated, and a lower and upper triangular solution is performed to compute the states. The partitioned solution procedure requires $[12\alpha N^2 + (r+m)^2 + 4rN + 4mN] + O(N)$ FLOP, where m is the number of actuators and r is the number of measurements. The number of sensors and actuators is usually small compared with the system-order N . The value of α for which the partitioned solution approach becomes more computationally attractive than the simultaneous solution approach must be determined. If the assumption is made that the number of actuators m and the number of measurements r are

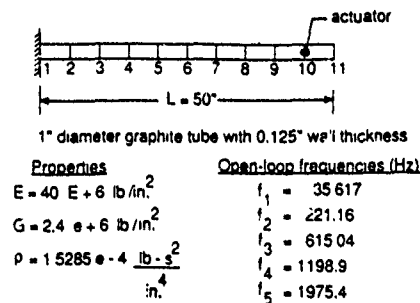


Fig. 5 Cantilever beam example.

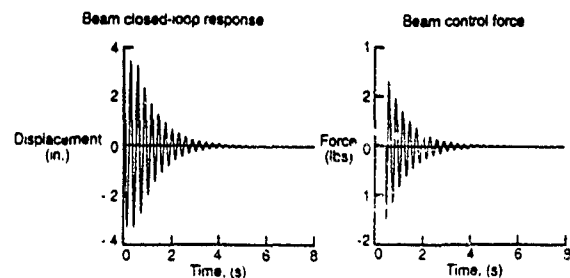


Fig. 6 Closed-loop response of example beam.

proportional to N by β

$$r = m = \beta N$$

the values of α and β for which the second-order solution is more efficient than the first-order solution are readily obtained. Figure 4 shows the efficiency ratio of the simultaneous and the partitioned solution procedures for various values of α and β . For practical cases where $\alpha \ll 1$ and $\beta < 1$, the partitioned solution approach is much more efficient.

It should be noted that the FLOP counts assessed previously are only approximate as their precise value depends on the cleverness of the implementation. Also, additional computational savings are possible if more efficient data structures such as profile storage schemes are employed for the mass, damping, and stiffness matrices.

VIII. Numerical Experiments

The time-discretized CSI equations [Eqs. (14-16)] have been implemented into three modules: the structural analyzer, the state estimator, and the solver for the interaction terms. The examples herein have concentrated on the structural analyzer and the control force interaction term. Additional examples with state estimation are presented in Refs. 26 and 28.

The first numerical example is a beam with 10 finite elements and an actuator placed on node 10 as shown in Fig. 5. The actuator gains have been determined using a single mode in the control law. The beam is initially bent according to its first mode shape, and the controller must regulate the response to 1% of its initial amplitude within 0.5 s. Figure 6 illustrates the tip displacement response and the control force u vs time. To assess the effectiveness of the present partitioned solution procedure, the response has been computed using both the simultaneous and the present technique for $N_s = 20$ and $N_e = 10$. The simultaneous technique does not use stabilization [Eq. (9)] for the computation of $u^{n+1/2}$. We have also computed the control force by postprocessing Eq. (1d) instead of the augmented form of Eq. (9). Figure 7 compares the control force error computed by the present stabilized Eq. (12d) vs that by direct substitution into the unaltered Eq. (1d) for step sizes of 20 and 10 samples per response cycle. It is noted that the present partitioned solution procedure yields

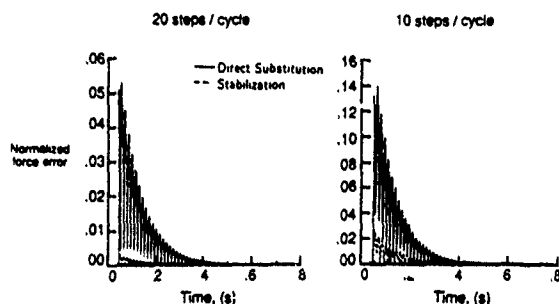


Fig. 7 Errors in computing control force for beam problem.

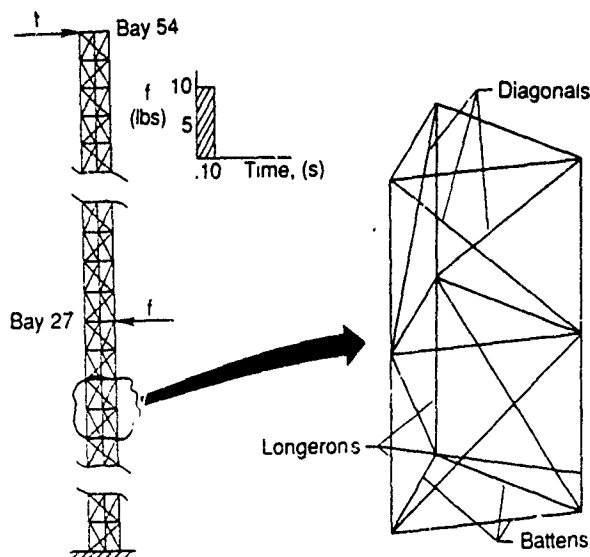


Fig. 8 Truss-beam example.

far better accuracy than the conventional direct substitution method.

Of course, one can eliminate the solution errors associated with the conventional simultaneous solution scheme. However, this requires solving the CSI equation in its entirety as given by Eq. (6), which cannot be carried out in a modular software environment.

To illustrate the procedure on more realistic structures, we have performed transient response analyses on the truss beam shown in Fig. 8. The truss was modeled by finite elements with one Timoshenko element from joint to joint. The model had 990 degrees of freedom. The truss is described in more detail in Ref. 25. A modal space control law with seven modes was used to suppress vibrations of the beam induced by the loading shown in Fig. 8. Both position and rate feedback were used to reduce the vibration amplitude to 0.025 in. within 10 s after active control was initiated.

Figure 9 shows the truss response along with the performance of the second-order observer of Eq. (15). Notice that the observer that is turned on concurrently with the actuator faithfully follows the structural response within an acceptable time lapse. Figure 10 demonstrates the errors in computing the control force by the stabilized Eq. (12d) and by directly substituting into the unaltered Eq. (1d). The computation of the control force by the stabilized Eq. (12d) is usually more than an order of magnitude more accurate than using the original form of Eq. (1d). This high accuracy removes the necessity of iterating during the prediction step.

Most important, the proposed technique, which exploits the second-order form of the structure equation, can be used to solve this 990-degree-of-freedom problem without resorting to a truncated modal model. Thus simulation can be efficiently performed with "truth" models to verify performance and robustness of control laws developed from reduced order

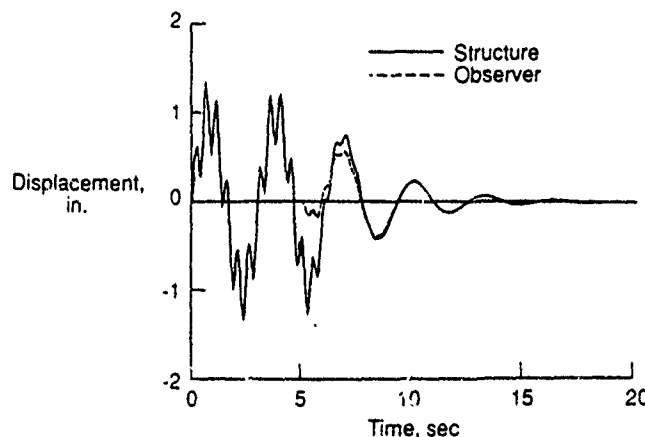


Fig. 9 Truss-beam closed-loop response.

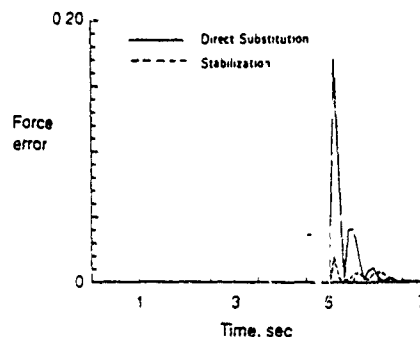


Fig. 10 Truss-beam control force prediction error.

models. It is this feature that will alleviate much of the computational difficulties associated with the study of control-structure interactions.

IX. Concluding Remarks

During the development of the present partitioned procedure for conducting control-structure interaction analysis, several theoretical and computational issues have been raised. First, the availability of a second-order observer model is required to exploit the symmetry and sparsity for efficiency purposes. However, second-order observers pose restrictions on the design of observer-based controllers. Second, for time-variant systems, the solution of a Riccati equation to determine the control force with appropriate terminal conditions and the solution of another Riccati equation for the observer equation with given initial conditions can pose bottlenecks in real-time control-structure interaction simulations. (Note that these two solutions are required when using optimal control theory to obtain the control and observer gain matrices.) Parallel computing can potentially remove the aforementioned bottlenecks of the time-variant problem. Finally, we have not considered the intrinsic dynamics of both the actuator and the sensor. However, since the present procedure uses a differential form for the actuator and sensor equations, intrinsic dynamics could be incorporated into the present computational procedure without too much difficulty. Further experiments and inclusions of more realistic control-structure interaction models are necessary before the present simulation procedure can become a production-simulation tool. This is being carried out at present.

Appendix: First-Order Observer Solution

The first-order observer can be numerically integrated as described as follows. Implicit midpoint formulas are used in the time discretization. All variables are defined in Eq. (1).

The first-order observer takes the form

$$\dot{\hat{x}} = A\hat{x} + Ef + Bu + L(z - H\hat{x}) \quad (A1)$$

Combining like terms, we derive

$$\dot{\hat{x}} = A_0\hat{x} + Ef + Lz \quad (A2)$$

where

$$A_0 = \begin{bmatrix} -L_1H_d & I - L_1H_v \\ -M^{-1}(K + BF_1) - L_2H_d & -M^{-1}(D + BF_2) - L_2H_v \end{bmatrix}$$

Time discretization yields

$$(I - \delta A_0)\hat{x}^{n+1/2} = \delta(Ef^{n+1/2} + Lz^{n+1/2}) + \hat{x}^n \quad (A3)$$

$$\hat{x}^{n+1} = 2\hat{x}^{n+1/2} - \hat{x}^n \quad (A4)$$

$$u^{n+1/2} = -F\hat{x}^{n+1/2} \quad (A5)$$

Equation (15) would be replaced by Eq. (A3) in the integration procedure. Then, Eq. (A5) would be used to solve Eq. (16a) and there would be no need to solve for γ . The computational efficiency of the first-order observer of Eq. (A3) is discussed in Sec. VII of the paper.

Acknowledgments

It is a pleasure to acknowledge the support of the first author by the Air Force Office of Scientific Research (AFOSR) through Grant F49620-87-C-0074 and of the second author through a NASA employee fellowship. We thank Anthony K. Amos of AFOSR and Jer-Nan Juang of NASA Langley Research Center for their encouragement during the course of this study.

References

- ¹Likins, P. W., "Dynamics and Control of Flexible Space Vehicles," NASA TR-32-1329, 1970.
- ²Balas, M. J., "Active Control of Flexible Systems," *Journal of Optimization Theory and Applications*, Vol. 25, July 1978, pp. 415-436.
- ³Roberson, R. E., "Two Decades of Spacecraft Attitude Control," *Journal of Guidance, Control, and Dynamics*, Vol. 2, No. 1, 1979, pp. 3-8.
- ⁴Skelton, R., "Theory and Applications in Optimal Control in Aero-space Systems," AGARD Publ. 251, edited by P. Kant, Brussels, Belgium, 1982.
- ⁵Meirovitch, L., and Silverberg, L. M., "Globally Optimal Control of Self-Adjoint Distributed Systems," *Optimal Control and Applications*, Vol. 4, No. 4, 1983, pp. 365-386.
- ⁶Horta, L. G., Juang, J. N., and Junkins, J. L., "A Sequential Linear Optimization Approach for Controller Design," AIAA Paper 85-1971, 1985.
- ⁷Kalman, R. E., "On the General Theory of Control Systems," *Proceedings of the First International Congress on Automatic Control*, Butterworths, London, 1961, pp. 481-491.
- ⁸Kalman, R. E., and Bucy, R. S., "New Results in Linear Filtering and Prediction Theory," *ASME Journal of Basic Engineering*, Vol. 83, 1961, pp. 95-108.
- ⁹Kwakernaak, H., and Sivan, R., *Linear Optimal Control Systems*, Wiley-Interscience, New York, 1974.
- ¹⁰Arnold, W. F., and Laub, A. J., "Generalized Eigenproblem Algorithms and Software for Algebraic Riccati Equations," *Proceedings of the IEEE*, Vol. 72, No. 12, 1984, pp. 1746-1754.
- ¹¹Park, K. C., "Computational Issues in Control-Structure Interaction Analysis," *Large Space Structures: Dynamics and Control*, edited by S. N. Atluri, Springer-Verlag, Berlin, 1988, pp. 115-131.
- ¹²Park, K. C., and Bevin, W. K., "Partitioned Procedures for Control-Structure Interaction Analysis," *Computational Mechanics '88*, edited by S. N. Atluri and G. Yagawa, Vol. 2, Springer-Verlag, Berlin, 1988, pp. 64.iii.1-4.
- ¹³Park, K. C., Felippa, C. A., and DeRuntz, J. A., "Stabilization of Staggered Solution Procedures for Fluid-Structure Interaction Analysis," *Computational Methods for Fluid-Structure Interaction Problems*, edited by T. Belytschko and T. L. Geers, AMD Vol. 26, American Society of Mechanical Engineers, New York, 1977, pp. 95-124.
- ¹⁴Park, K. C., and Felippa, C. A., "Partitioned Analysis of Coupled Systems," *Computational Methods for Transient Analysis*, edited by T. Belytschko and T. J. R. Hughes, Elsevier, Amsterdam, 1983, pp. 157-219.
- ¹⁵Zienkiewicz, O. C., "Coupled Problems and Their Numerical Solution," *Numerical Methods in Coupled Systems*, edited by R. W. Lewis, P. Bettess, and E. Hinton, Wiley, New York, 1984, pp. 35-38.
- ¹⁶Park, K. C., and Chiou, J. C., "Stabilization of Computational Procedures for Constrained Dynamical Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 11, No. 4, 1988, pp. 365-370.
- ¹⁷Hughes, P. C., and Skelton, R. E., "Controllability and Observability of Linear Matrix Second-Order Systems," *Journal of Applied Mechanics*, Vol. 47, 1980, pp. 415-420.
- ¹⁸Hashemipour, H. R., and Laub, A. J., "Kalman Filtering for Second-Order Models," *Journal of Guidance, Control, and Dynamics*, Vol. 11, No. 2, 1988, pp. 181-185.
- ¹⁹Meirovitch, L., and Oz, H., "Modal-Space Control of Distributed Gyroscopic Systems," *Journal of Guidance and Control*, Vol. 5, No. 1, 1982, pp. 140-150.
- ²⁰Yousuff, A., and Skelton, R. E., "Controller Reduction by Component Cost Analysis," *IEEE Transactions on Automatic Control*, Vol. AC-24, 1984, pp. 520-530.
- ²¹Hale, A. L., Lisowski, R. J., and Dahl, W. E., "Optimal Simultaneous Structural and Control Design of Maneuvering Flexible Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 8, No. 1, 1985, pp. 86-93.
- ²²Hafka, R. T., et al., "Sensitivity of Optimized Control Systems to Minor Structural Modifications," 26th Structures, Structural Dynamics, and Materials Conference, Orlando, FL, April 1985.
- ²³Junkins, J. L., and Rew, D. W., "Unified Optimization of Structures and Controllers," *Large Space Structures: Dynamics and Controls*, edited by S. N. Atluri and A. K. Amos, Springer-Verlag, Berlin, 1988, pp. 323-353.
- ²⁴Khot, N. S., et al., "Optimal Structural Modifications to Enhance the Optimal Active Vibration Control of Large Flexible Structures," 26th Structures, Structural Dynamics, and Materials Conference, Orlando, FL, April 1985.
- ²⁵Belvin, W. K., and Park, K. C., "Structural Tailoring and Feedback Control Synthesis: An Interdisciplinary Approach," AIAA Paper 88-2206, May 1988, also *Journal of Guidance, Control, and Dynamics* (to be published).
- ²⁶Belvin, W. K., and Park, K. C., "On the State Estimation of Structures with Second Order Observers," AIAA Paper 89-1241, April 1989.
- ²⁷Luenberger, D. G., "Observing the State of a Linear System," *IEEE Transactions on Military Electronics*, Vol. 8, 1964, pp. 74-80.
- ²⁸Belvin, W. K., "Simulation and Interdisciplinary Design Methodology for Control-Structure Interaction Systems," Ph.D. Thesis, Center for Space Structures and Controls, Univ. of Colorado, Boulder, CO, Rept. CU-CSSC-89-10, July 1989.
- ²⁹Dahlquist, G., "A Special Stability Problem for Linear Multi-step Methods," *BIT*, Vol. 3, 1963, pp. 27-43.
- ³⁰Hughes, T. J. R., and Belytschko, T., "A Précis of Developments in Computational Methods for Transient Analysis," *Journal of Applied Mechanics*, Vol. 50, 1985, pp. 1033-1041.
- ³¹Park, K. C., "Transient Analysis Methods in Computational Dynamics," *Finite Elements: Theory and Applications*, edited by D. L. Dwyer, M. Y. Hussaini, and R. G. Veigt, Springer-Verlag, New York, 1988, pp. 240-267.
- ³²Park, K. C., "Partitioned Analysis Procedures for Coupled-Field Problems: Stability Analysis," *Journal of Applied Mechanics*, Vol. 47, 1980, pp. 370-378.
- ³³Gantmacher, F. R., *The Theory of Matrices*, Vol. 2, Chelsea, New York, 1959, pp. 190-196.

PARALLEL COMPUTATIONS AND CONTROL OF ADAPTIVE STRUCTURES

K. C. Park and Kenneth Alvin

Department of Aerospace Engineering Sciences and
Center for Space Structures and Controls
University of Colorado, Campus Box 429
Boulder, Colorado 80309

and

W. K. Belvin
Spacecraft Dynamics Branch
NASA/Langley Research Center
Hampton, VA 23665

ABSTRACT

The equations of motion for structures with adaptive elements for vibration control are presented for parallel computations to be used as a software package for real-time control of flexible space structures. A brief introduction of the state-of-the-art parallel computational capability is also presented. Time marching and the necessary arithmetic operations. An example is offered for the simulation of control-structure interaction on a parallel computer and the impact of the approach presented herein for applications in other disciplines than aerospace industry is assessed.

1. Introduction

Active suppression of structural vibrations or active control of flexible structures has made considerable progress in recent years. As a result, it is now possible to actively suppress vibrations in mechanical systems emanating from machine foundations, in robotic manufacturing arms, truss-space structures and automobile suspension systems. A common characteristic to these applications of active control theory has been its discrete actuators and discrete sensors, ranging from proof mass actuators and gyro

shambers to strain gages and accelerometers. Because most available discrete actuators are inertia force-oriented devices, actuation often triggers coupling between the actuator dynamics and structural transients. A practical consequence of such coupling is a limitation of achievable final residual vibration level if both the actuator and structure possess insufficient passive damping level. It is noted that structures made of high stiffness composite materials have very low intrinsic damping, hence limiting the achievable residual vibration level for space maneuvering and space disturbance rejection purposes. This has been a motivating factor for the development of distributed actuators and sensors which are often embedded as an integral part of the structure so that control force can be effectively maintained by strain actuation, thus alleviating the undesirable actuator dynamics associated with inertia-force actuation.

Various activities that are being pursued by many investigators on the subject of adaptive structures may be categorized into three major thrusts: device developments, control laws synthesis and experimental demonstrations, and hardware/software implementation. The device developments effort has been the objective of many material scientists [1-3]. As the applications needs increase it is expected that functionally more reliable electrostrictive and magnetostrictive elements will be available for use in active control/strain damping with improved product quality.

The study of control laws synthesis and demonstration employing adaptive elements has been one of the predominant activities in recent years. As scientists accumulate experience in the characterization of the coupling between the structure and the adaptive element, the applications will then be expanded from the current beam-like structures to the truss long beams, plates and shells. In order to effectively utilize as many adaptive elements as necessary for actively controlling the vibration of such large-scale structures in real-time operations, it will be imperative that the software/hardware components in the real-time control loop must be able to process data fast enough so that control commands and the measurements can be carried without saturating and/or jamming the control system.

With the advent of new technology in distributed actuators and sensors [4-9], it appears that a combination of decentralized/distributed and hierarchical control strategies can be a viable alternative to conventional centralized control strategies. The real-time computer control of such systems as well as design of such control systems through iterating on simulations and hardware realizations thus will require the processing of a vast amount of data from and to the distributed actuators and sensors. A significant part of such data processing for the decentralized actuators and sensors is planned to be self-managed, viz., there will be embedded microprocessors for each actuator and sensor pair or for each group of them. However, the necessary links between the decentralized control systems and the global control system as well as the necessary global control strategy will still require computational power far in excess of presently available real-time data processing capability. In addition, if one contemplates the performance of neural-network control or adaptive control for onboard real-time control of large-scale space structures, the computational need will dramatically increase beyond the current capability. As a case in point, even for the control of 20-bay truss beam vibrations by

three proof mass actuators and six sensors, NASA/Langley is relying on CRAY-XMP for adequate real-time data processing requirements.

The objective of this paper is thus to present a computational framework by which one can bring the two emerging new technologies together, namely, the distributed actuators and sensors and the parallel computing capability, toward the real-time control of vibrations in large structural systems such as space stations, space cranes and in-space construction facilities. We will then discuss the potential for applying such a space technology to mitigate and/or minimize the earthquake damage of ground structures such as high-rise buildings, bridges and lifeline equipment.

2. Models for Structures with Embedded Actuators and Sensors

The coupling between the structural behavior and an adaptive electrostrictive element, whether it is embedded or surface-mounted, is primarily due to the following constitutive relation [3,10-12]:

$$\begin{Bmatrix} \epsilon \\ \sigma \end{Bmatrix} = \begin{bmatrix} \epsilon & g \\ -g^T & c \end{bmatrix} \begin{Bmatrix} v \\ \epsilon \end{Bmatrix} \quad (1)$$

where ϵ and v are the electrical displacement (charges/unit area) and the electric field (volt/unit area), σ and ϵ are the stress and strain, and ϵ , g and c are the constitutive coefficient matrices, respectively. For magnetostrictive elements, one needs to replace ϵ and v by the magnetic field (H) and the magnetic induction (B), respectively, and the subsequent derivations will hold without any loss of generality.

The coupled equations of motion for the structure and the adaptive elements can proceed by augmenting the standard procedure for the structure with the electric transient equations plus the appropriate modification of the structural equilibrium equations that reflect the coupled constitutive equations (1). The resulting coupled structural-piezoelectric equations of motion take the following form [13-15]:

$$\left\{ \begin{array}{ll} \text{Structure:} & a) \quad M\ddot{q} + D\dot{q} + (K_s + K_a)q = f + S a \\ & q(0) = q_0, \quad \dot{q}(0) = \dot{q}_0 \\ \text{Sensor Output:} & b) \quad y = H_p q + H_s \dot{q} + H_a a \\ \text{Actuator:} & c) \quad \dot{a} + \Theta a = B_a u - \bar{S}^T \begin{Bmatrix} q \\ \dot{q} \end{Bmatrix} \\ \text{Controller:} & d) \quad \dot{u} + G u = L y \end{array} \right. \quad (2)$$

$$a = \begin{Bmatrix} \epsilon \\ v \end{Bmatrix}, \quad u = \begin{Bmatrix} I_0 \\ V_0 \end{Bmatrix}$$

where

In the preceding equations, M is the mass matrix, D is the damping matrix, K_s is the stiffness matrix due to structural strain-displacement relations and K_a is the stiffness matrix due to the strain actuation. $f(t)$ is the applied force. S is the actuator projection matrix. H_p , H_r and H_a are the sensor calibration gain matrices, Θ is actuator dynamic characteristics, B_a is the gain matrix that translates the applied current/charge and voltage into the corresponding strain and strain rate where $\dot{S}$ is the transducer conversion gain q is the generalized displacement vector and the superscript dot denotes time differentiation, and u is the control law that consists of the applied current (or charge), I_0 , and voltage across the electrostrictive devices, V_0 , G is the electric circuit characteristics, and L is the optimum direct feedback gain matrix. The case of dynamic compensations can be augmented to (2) by introducing an observer. But in subsequent discussions we limit ourselves to direct feedback cases only.

It is noted that the control laws, unlike conventional control-structure interaction systems, are not directly fed back into the structural equations. Instead, the controller is simply a regulator controlling the electric charge, the voltage or the current. These regulated electric quantities are then fed into the piezoelectric sensors and actuators. Hence, it is the piezoelectric actuation that triggers feedback into the structures.

3. Parallel Computations for the Dynamics of Adaptive Structures

The earliest recorded computational results in mechanics were the parabolic trajectory calculations of a falling body by Galileo [16]. Since then, most scientific computations have been carried out by anthropomorphic algorithms, viz., step-by-step binary and/or decimal arithmetics. To set the stage properly for the present objective, parallel computations of the dynamic response of structures with distributed adaptive elements, we recall a passage by Kepler to John Napier, the inventor of a logarithmic table:

Newton was essentially dependent upon the results of Kepler's calculations, and these calculations might not have been completed but for the aid of that logarithms afforded. Without the logarithms, ... the development of modern science might have been very different [17].

In terms of the present day data processing requirement, Napier's logarithmic table in 1614 contained about 100 kilobytes, which was perhaps the most important computational aid to Kepler and Newton. Three and one-half centuries later we are witnessing gigabytes of tables being stored and retrieved at our disposal [18]. But these tables complement the weakness of the human mind and computational speed. long term memory and human arithmetic speed. In addition, for problems requiring a sequential nature of computations, i.e., ballistic trajectories which deal only with the position and velocity of a single shell or quasi static equilibrium equations of a building structure, the computing activities do not interact with "time" and the computing efficiency affects only the manpower efficiency for completing the computational task.

There are many important scientific and engineering endeavors whose computations must be fast enough for real-time delivery of the computed results. A classical example was Richardson's lattice model for weather prediction by numerical process in 1922 [19]. The motivation for adopting such a lattice concept was due to the fact that the equation state at each lattice node takes on a different value set in time and an efficient way of interchanging and transmitting the nodal values at each time step was mandatory if the computations were to be carried out in real-time to predict the weather. Indeed, this was the dawn of the parallel computing era, even though the basic idea had to wait for its validity for 60 years. Today, many controls engineering activities have been implemented by using computers so that their intended functions can be monitored and controlled in real-time. These include chemical processing, autopiloting and vibration control of simple structures. It is important to note that the computational framework employed for such applications is based on sequential architecture. Hence, we believe that future improvements that can deal with large parameter models and large parameter controls must adopt a parallel computational framework. One such area is the dynamics and control of large structures with distributed/embedded adaptive elements.

In order to carry out the necessary parallel computations, there are three distinct steps that must be addressed: discretizing the structure into appropriate partitions, mapping the physical partitions onto the processors, and step advancing of the equation states. These will be discussed below.

3.1 Partitioning and Mapping of Adaptive Structures

Ideally, if the sensor and actuator leads fall on the discrete nodes, no spatial interpolation would be necessary. However, such a situation is either difficult to realize or may prohibit the use of spatially convolving sensors [20] that are known to filter certain harmonic signals for minimizing phase lag in the feedback loop. Hence, we will assume that the sensor and actuator characteristics can be interpolated to the discrete nodes; in this way the partition boundaries can be chosen arbitrarily regardless of the physical locations of the sensor and actuator leads. In addition, this approach can lead to a natural embedding of the sensor and actuator characteristics into the finite element or boundary integral structural models. Once the partitioning is accomplished, the next step is to map the discrete partitions for adaptive elements onto the corresponding multiprocessors.

Consider an adaptive structure that has been modeled as a set of discrete elements as shown in Fig. 1. In a sequential computing environment, in order to advance the necessary computations for the present states, the arithmetic operations are carried out step-by-step for each node at a time. Hence, each nodal-state computations is performed in a manner similar to one courier delivering and picking up all the mails throughout the entire routes. In a parallel computational environment, in contrast, there can be as many couriers as necessary who comb through the routes concurrently in order to pick up and deliver all the mail at once. One of the most popular concepts in executing such tasks is the hypercube architecture (see Fig. 2) whose every node

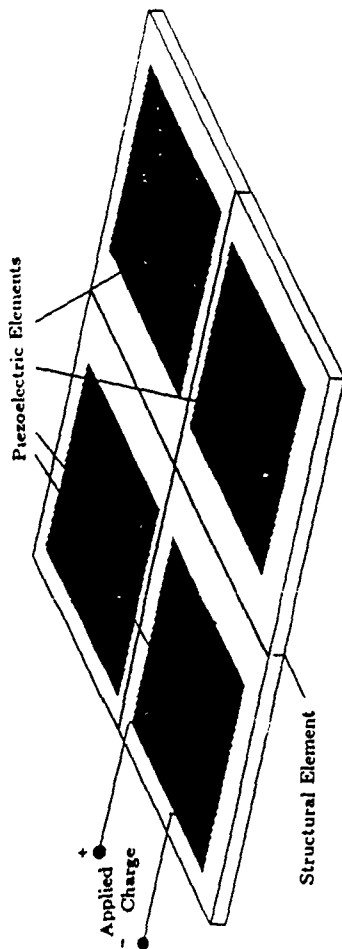
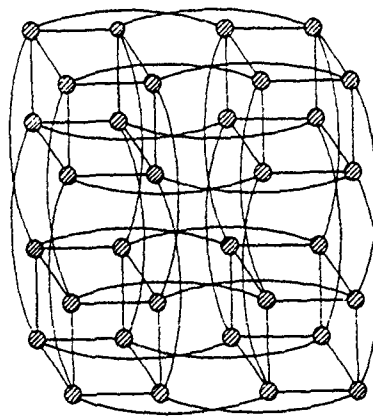


Fig. 1 Discrete Model of Adaptive Structures

Fig. 2 Hypercube Interconnection Network of a 32-Processor
(each node represents a processor)

is associated with a processor. Thus, to process the necessary computations for an adaptive structure with 19 partitions, one can assign the 19 adaptive elements to 19 processors as shown in Fig. 3. The procedure for assigning the physical domain (elements) to the parallel processors with minimal interprocessor communications is called mapping.

Of several techniques available for the processor mapping of the computational domains [22], we will adopt a heuristic mapper developed by Farhat [23] since it can accommodate both the synchronous and asynchronous cases with robust and acceptable complexities. An application of this mapping technique for modeling a bulkhead substructure for massively parallel computing is shown in Fig. 4. A similar mapping can be used for parallel computations of adaptive structures.

3.2 Parallel Data Structure and Algorithms

We will assume that each processor is assigned to carry out all the necessary computations for at least one set of a sensor, an actuator, and a controller or a group of them. Therefore, the word *partition* does not necessarily imply a finite element; it can be a substructure, an element or even a sublayer within the composite layer that includes a sensor or an actuator. In carrying out the step-advancing in time, one may invoke an implicit or explicit direct time integration algorithm. When an implicit algorithm is employed, one needs to communicate not only the state variable vectors but also the associated matrices, i.e., the stiffness matrix, among the processors. Although we will show our results using implicit algorithms, we will, for illustrative purposes, restrict ourselves to an explicit direct time integration algorithm as it is intrinsically parallel and the data structure aspects can be explained more succinctly via an explicit algorithm. It should, however, be mentioned that the choice of the solution algorithm can greatly influence the design and implementation of the necessary mapping and data structure.

Consider the explicit integration of the equations of motion for the structure (2a) as recalled here:

$$M\ddot{q} + f_{int} = f + f_{cont} \quad (3)$$

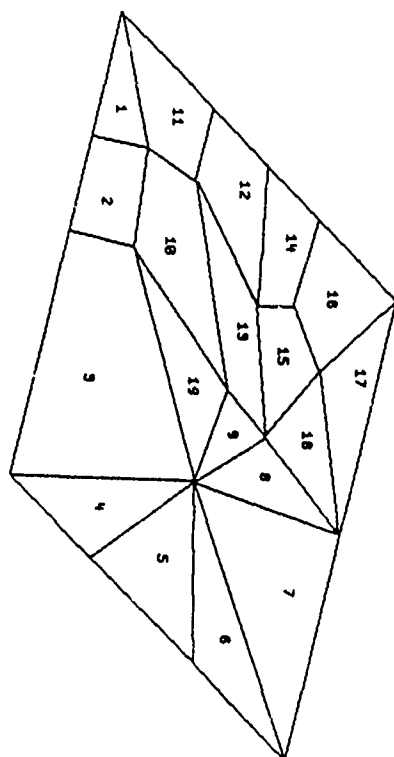
where f_{int} and f_{cont} are the internal and applied control forces, respectively, given by

$$f_{int} = D\dot{q} + (K_s + K_a)q$$

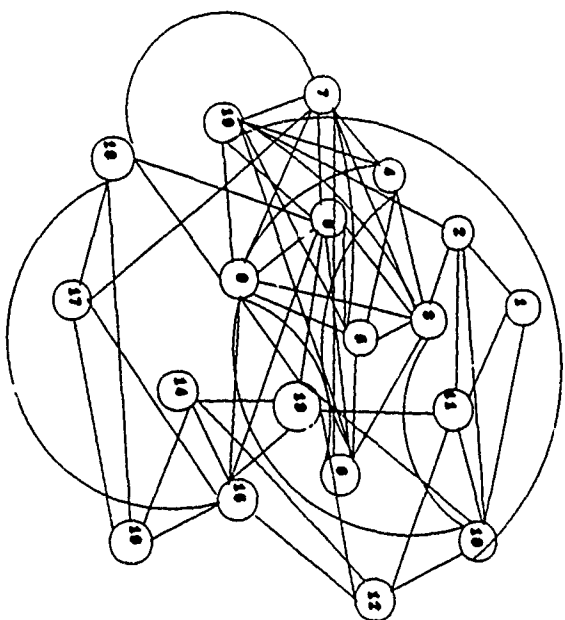
$$f_{cont} = Sa$$

The use of the central difference algorithm to integrate (3) leads to the following difference equations in time

$$\begin{cases} q^{n+\frac{1}{2}} = q^{n-\frac{1}{2}} + hM^{-1}(f^n + f_{cont}^n - f_{int}^n) \\ q^{n+1} = q^n + h\dot{q}^{n+\frac{1}{2}} \end{cases} \quad (4)$$



(Physical Domain)



(Processor Configuration)

Fig. 3 Physical Domain and Its Mapping Onto
Hypercube Processors

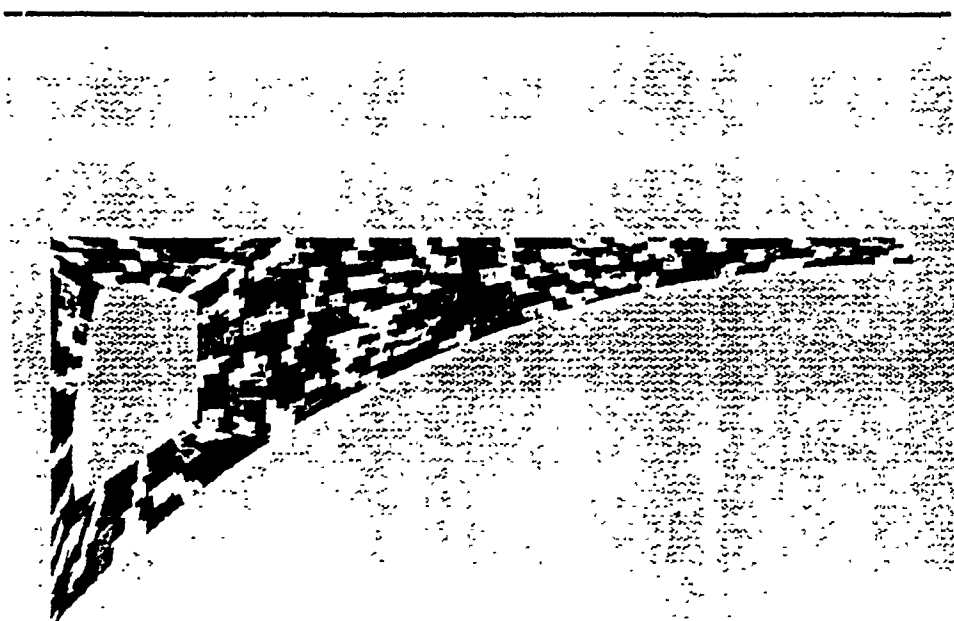


Fig. 4 Decomposition of the Structure with
"Finite Element Chips"

where the superscript n is the discrete time station and h is the timestep increment.

It is observed that all the necessary computations to advance the integration marching can be performed in terms of vectorial quantities. As the mass matrix can be made diagonal, this can be computed in a host computer and broadcast to the processors at the beginning of the computations. However, the computations of the internal force f_{int} require further considerations. If one employs the finite element method, the internal force for the case of no damping can be constructed as

$$f_{int} = \sum_{l=1}^N \int_V B^T \sigma dV \quad (5)$$

where N is the total number of elements, B is the strain-displacement relation matrix, σ is the stress expressed in a vector form, V is the volume of the structural domain. If we interpret $\int_V B^T \sigma dV$ as elemental or substructural computations, then $\sum_{l=1}^N$ can be considered a Boolean operator that sums up all the elemental contributions for the internal force vector. It is this property that can be exploited in parallel computations. This can be implemented as follows.

Before we begin the time marching computations, we load all the elemental data to each processor's memory, which include the element type (beam, plate, shell, spatial integration rule, etc.), intrinsic data (density, Young's modulus, damping coefficient, ..., etc.), geometry, boundary conditions, applied force, and parameters needed for time integration.

Notice that, in computing σ , the displacement q at adjacent elements must also be routed accordingly. Suppose p is a processor to which the element e_p is assigned. Therefore, the processor p must have information regarding from which processors it must receive q , to which ones it must send its information, and how particular degrees of freedom have to be assembled. For efficient computations, this information needs to be loaded onto the processors, thus avoiding the message passing from the front end to the processors at each time marching step. To implement this, each nodal data set for the processor p are given an identifier that carry which elemental data sets in other processors are connected to this node. In addition, for each entry in this data set, a location pointer is provided so that the incoming data from the neighboring processors can be allocated to the corresponding degrees of freedom in the processor p . More detailed discussions on parallel data structures may be found in [25].

As an example, consider the element e_p mapped onto the processor p as shown in Fig. 5. The internal nodes 1 and 2 need data transfer only from processors R and T . On the other hand, nodes 3 and 4 do need information from all the neighboring processors. It is this need that requires an efficient and reliable mapping, interprocessor communication algorithms and streamlined data arrangement as described above.

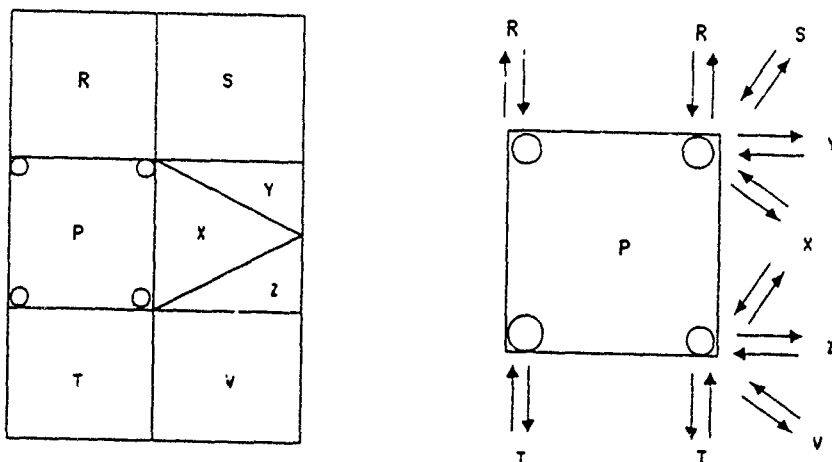


Fig. 5 Partitioning and Communication Requirement

4. Implementation and Illustrative Example

The mapping, partitioning and data structures above discussed have been implemented based on a shared-memory concurrent machine (Alliant FX/8) by modifying the software framework developed for finite element computations [26] and the control-structure interaction simulation and design software developed in [27, 28]. At present the following specialized systems of equations are implemented:

$$\left\{ \begin{array}{ll} \text{Structure:} & a) \quad M\ddot{q} + D\dot{q} + Kq = f + Bu + Gw \\ & q(0) = q_0, \quad \dot{q}(0) = \dot{q}_0 \\ \text{Sensor Output: } b) & z = Hx + m \\ \text{Estimator:} & c) \quad \dot{\hat{x}} = A\hat{x} + Ef + \bar{B}u + L(z - H\hat{x}) \\ & \hat{x}(0) = 0 \\ \text{Control Force: } d) & u = -F\hat{x} \end{array} \right. \quad (6)$$

where

$$x = \begin{Bmatrix} q \\ \dot{q} \end{Bmatrix}, \quad \hat{x} = \begin{Bmatrix} \hat{q} \\ \dot{\hat{q}} \end{Bmatrix}$$

and

$$H = [H_d \quad H_v], \quad L = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix}, \quad F = [F_1 \quad F_2]$$

It is noted that in the above implemented equations, we have merged the actuator and the control law equations into one by neglecting the actuator and control law dynamics. Instead, we have introduced an estimator equation as we do not have all the measurements needed for complete feedback. In the above equations, B and $\bar{B}$ represent the input influence matrix for actuator locations whereas G and $\bar{G}$ represent the disturbance locations. The vector q is the generalized displacement, w is a disturbance vector and the vector m is measurement noise. In Eq. (6b), z is the measured sensor output. The matrix H_d is the matrix of displacement sensor locations and H_v is the matrix of velocity sensor locations. The state estimator in Eq. (6c) may or may not be model based. The superscript $\sim$ and $\cdot$ denote the estimated states and time differentiation respectively. The input command, u , is a function of the state estimator variables, $\hat{q}$ and $\dot{\hat{q}}$, and F_1 and F_2 are control gains. The observer is governed by A , the state matrix representing the plant dynamics, and L , the filter gain matrix.

The software thus implemented was used to test its applicability to solve the control-structure interaction design of a model Earth Pointing Satellite (EPS), shown in Fig. 6, which is a derivative of a geostationary platform proposed for the study of Earth Observation Sciences. Two flexible antennas are attached to a truss bus. Typical missions involve pointing one antenna to earth, while tracking or scanning with

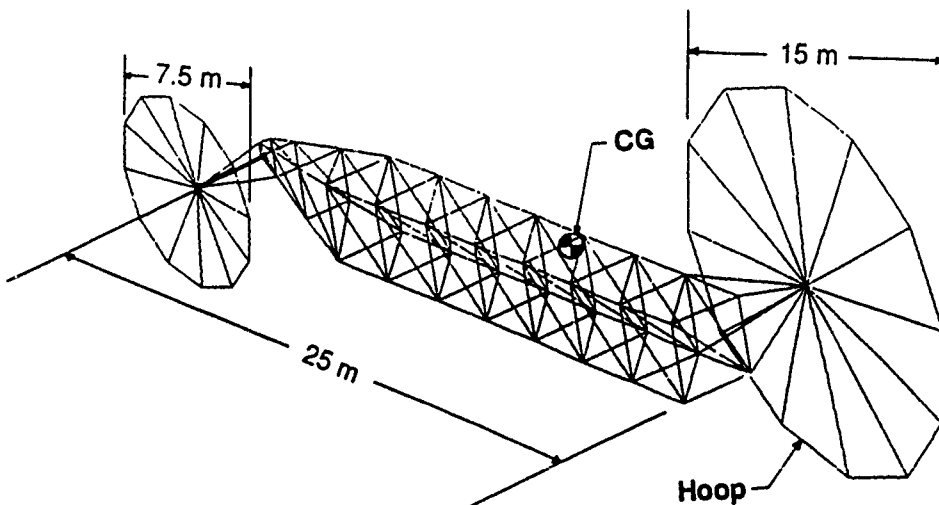


Fig. 6 Earth Pointing Satellite Structure

the other antenna. The EPS structure possesses the common feature of a number of proposed spacecraft, namely, several quite flexible appendages are attached to a relatively rigid bus structure. This type of structure is difficult to control since the attitude control system is usually located on the bus. The attitude control system location may render many of the flexible appendage modes uncontrollable. The following section describes an integrated structure and control design for the EPS system which includes additional actuators for flexible body vibration suppression.

There are six basic components that make up the EPS structure: the truss, a 15 m antenna, a 7.5 m antenna, the 15 m antenna to truss attachment, the 7.5 m antenna to truss attachment and the center of gravity attitude control system support structure. The truss has a three meter square cross section. It consists of 51 mm diameter by 1.59 mm wall thickness graphite epoxy tubes. The modulus of elasticity and mass density of the tubes are $275.9 \times 10^9 \text{ N/M}^2$ and 3250.0 Kg/M^3 , respectively. A total of 135 beam elements form the truss.

The 15 m antenna is a simple design which exhibits dynamic structural deformations which are characteristic of large space antennas. The antenna has 12 ribs which radiate from the center. The ends of the ribs are connected by hoop elements as shown in Fig. 6. Graphite tube diameters and thicknesses are the same as those used for the truss for the nominal design. However, nonstructural mass which simulates the antenna reflector is included. The nonstructural mass is assumed to be twice the weight of the antenna members. The 7.5 m antenna has a geometrically similar design to the 15 m design. Again, for the nominal design, the tube diameters are the same as used on the truss. Nonstructural mass is also assumed, for the 7.5 m antenna, to be twice the weight of the antenna members.

The antenna to truss connections, for both antennas, are made by four beam elements with the same nominal material and tube properties as the truss members. The attitude control system is supported at the center of gravity (CG) of the nominal design by eight beam elements. The attitude control system support is also constructed from the same tubes as used in the truss.

Three torque actuators are located at the CG each with an assumed mass of 50 Kg. This yields a total vehicle mass of 1027.95 Kg. The mass is divided into 548.32 Kg of structural mass, 150 Kg of attitude control mass, and 329.63 Kg of nonstructural mass on the antennas. The finite element model representing the EPS consists of 570 degrees of freedom. Table 1 lists the first 20 frequencies for the nominal EPS structure. Torsion and/or bending of both the 15 m and 7.5 m antennas dominate the flexible body motion in modes 7 through 20. For example, the first flexible mode, mode 7, is torsion of the 15 m antenna. Modes with repeated frequencies (e.g. mode pairs 11-12, 15-16, and 17-18) involve bending of the antennas and occur in pairs due to the symmetry of the antennas.

Table 1. EPS Vibration Frequencies (Hz.)

Mode No.	Frequency
(1 - 6)	0.000
(7)	0.242
(8)	0.406
(9)	0.565
(10)	0.656
(11,12)	0.888
(13)	1.438
(14)	1.536
(15,16)	1.776
(17,18)	3.026
(19)	3.513
(20)	3.531

A small disturbance force was applied to the nominal EPS system in the form of a reboost maneuver. The force acted at the center of gravity in the Y-axis direction for 0.1 seconds at a 10 N force level and from 0.1 to 0.2 seconds the force level was -10 N. The disturbance was removed after 0.2 seconds. Figure 7 shows the open-loop angular response about the X-axis of the 15 m antenna. A small amount of passive damping was assumed ($D = 0.0002 \text{ K}$). The vibrational response produced more than 4.5μ radians of RMS pointing error due to this small reboost disturbance. Although many modes participate in the flexible body response, this particular reboost maneuver strongly excites modes near 4 Hz. The following paragraphs present an integrated control and structure design which seeks to lower the vibrational response of the EPS subject to some additional constraints. Figure 8 shows the closed-loop angular response about the X-axis of the 15 m antenna after design optimization. The pointing error is significantly reduced from that of the open-loop system shown.

5. Future Work and Discussions

The example problem analyzed in the previous section used a set of lumped actuators and localized sensors instead of distributed adaptive actuators and spatially integrated sensors. While such a model at best capture the adaptive elements used by Anderson et al. [29], Matsunaga [30], and Takahara [31], it can not simulate on a large scale the distributed usage of piezoelectric actuators and sensors proposed by de Luis [32], Rogers et al. [33], and Burk and Hubbard [34]. Our immediate future work will concentrate on the implementation of distributed adaptive elements and assess their practical applicability beyond the currently reported beam-like structural components. In this regard, we are exploring an adaptation of neural-network concepts [35] in the modeling and parallel computations of controlled structures with adaptive elements. Specifically, the limits of the applicability of distributed parameter modeling and control theory and discrete structures with discrete actuators and sensors, and their cross-over

performance must be investigated. Design, modeling, simulation and testing criteria from such studies will provide greater insight into the eventual adoptions of adaptive structures as viable choice for future space systems design alternatives.

The real-time simulation procedures presented herein may be applicable to the vibration control of lifeline equipment, and secondarily in minimizing the damage of buildings during earthquakes. In this applications, the sensor measurements used herein can be directly applicable to the vibration and earthquake-causing forces on the structures. An idea that may prove to be crucial in this case is the use of earthquake-generated natural force as vibration minimization actuators. In other words, instead of trying to mitigate the earthquake-generating forces, exploit the natural forces instantly to activate certain vibration minimizing devices! Research along this line may in the end lead to the design of actuators attachable to the columns and floors, if properly triggered during earthquakes, can minimize damages based on the natural forces.

Acknowledgements

It is pleasure to acknowledge the support from NASA/Langley Research Center under Grant NAG-1-1021 and by Air Force Office of Scientific Research under Grant F49620-87-C-0074. We thank Dr. Spencer Wu of AFOSR and Dr. Ernst Armstrong of NASA/Langley Research Center for their encouragement during the course of this study. We thank Prof. Charbel Farhat for his assistance in data structures and implementation aspects employed in the present study.

References

1. Jaffe, B., Cook, W. and Jaffe, H., *Piezoelectric Ceramics*, Academic Press, London and New York, 1971.
2. Zelenka, J., *Piezoelectric Resonators and Their Applications*, Elsevier Science Publishing Co., Inc., New York, 1986.
3. Cross, L.E., Piezoelectric and electrostrictive sensors and actuators for adaptive structures and smart materials. In *Adaptive Structures*, ed. B.K. Wada, ASME, AD-Vol. 15, New York, 1989, pp. 9-17.
4. Forward, R.L., Electronic damping of vibrations in optical structures. *Journal of Applied Optics*, 1979, 18 (5), 690-697.
5. Crawley, E.F. and de Luis, J., Use of piezoelectric actuators as elements of intelligent structures. *AIAA Journal*, 1987, 25, (10), 1373-1385.
6. Bailey, T. and Hubbard, J.E., Distributed piezoelectric polymer active vibration control of a cantilever beam. *Journal of Guidance, Control, and Dynamics*, 1985, 8, (5).

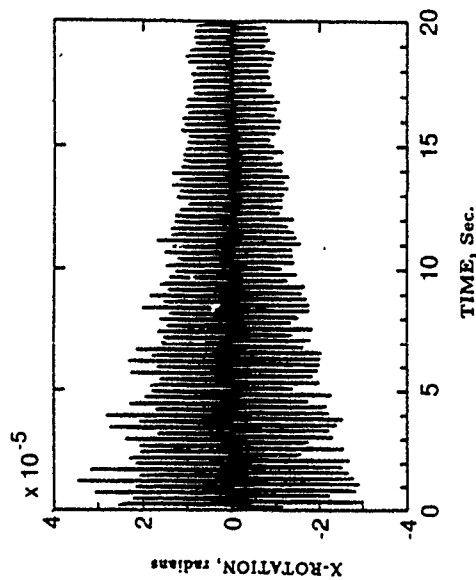


Fig. 7 Open-Loop Response of EPS Structure

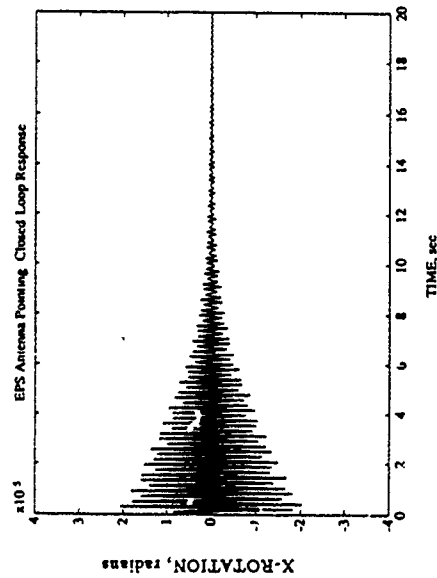


Fig. 8 Closed-Loop Response of Structure EPS

7. Hanagud, S., Obal, M.W. and Calise, A.J., Optimal vibration control by the use of piezoceramic sensors and actuators, AIAA Paper No. 87-0959, presented at the 28th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics and Materials Conference, Monterey, CA, April 1987, pp. 987-997.
8. Lee, C.K., Chiang, W.W., and O'Sullivan, T.C., Piezoelectric modal sensors and actuators achieving critical damping on a cantilevered plate. Proc. the 30th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics and Materials Conf., AIAA, Washington D.C., 1989, pp. 2018-2026.
9. Baz, A. and Poh, S., Performance of an active control system with piezoelectric actuators. *Journal of Sound and Vibration*, 1988, 126, (2), 327-343.
10. Newnham, R.E., Skinner, D.P. and Cross, L.E., Connectivity and piezoelectric-pyroelectric composites. *Mat. Res. Bull.*, 1978, 13, 525.
11. Lee, C.K., Piezoelectric laminates for torsional and bending modal control: theory and experiment. Ph.D. Thesis, Cornell University, Ithaca NY, 1987.
12. de Luis, J., Crawley, E.F. and Hall, S.R., Design and implementation of optimal controllers for intelligent structures using infinite order structural models. Report No. 3-89, Space Systems Laboratory, M.I.T., Cambridge, MA, 1989.
13. Tzou, H.S. and Tseng, C.I., Distributed piezoelectric sensor/actuator design for dynamic measurement/control of distributed parameter systems: a finite element approach. *Journal of Sound and Vibration* 1990, 137, (1).
14. Nailon, M., Coursant, R.H. and Besnier, F., Analysis of piezoelectric structures by a finite element method. *ACTA Electronica*, 1983, 25, (4), 341-362.
15. Hagood, N. and von Flotow, A., Modelling of piezoelectric actuator dynamics for active structural control. AIAA Paper No. 90-1087, Proceedings of the 31st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Long Beach, CA, April 1990, pp. 2242-2256.
16. Galileo, G., *Two New Sciences*, Dover Publication, New York, 1954, pp. 284-288.
17. Napier, M., *Memoirs of John Napier of Merchiston*, William Blackwood, Edinburgh, 1834, p. 501.
18. Anonymous, *Mathematica*, Wolfram Research, Inc., 1989.
19. Richardson, L. F., *Weather Prediction by Numerical Process*, Dover, New York, 1922, p. 219.
20. Miller, D., Collins, S. and Peltzman, S., Development of spatially convolving sensors for structural control applications. AIAA Paper No. 90-1127, Proceedings of the 31st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Long Beach, CA, April 1990, pp. 2283-2297.

21. Noor, A.K., Parallel processing in finite element structural analysis. In *Parallel Computations and Their Impact on Mechanics*, ed. A.K. Noor, American Society of Mechanical Engineers, New York, 1987, pp. 253-277.
22. Bokhari, S.H., On the mapping problem. *IEEE Transactions on Computers*, 1981, C-30, (3), 267-274.
23. Farhat, C., "On the mapping of massively parallel processors onto finite element graphs," *Computers & Structures*, Vol 32, No. 2, 347-354 (1989).
24. Farhat, C., Felippa, C.A. and Park, K.C., "Implementation Aspects of Concurrent Finite Element Computations," in *Parallel Computations and Their Impact on Mechanics*, American Society of Mechanical Engineers, New York, 1987, pp. 310-316.
25. Farhat, C., Sobh, N. and Park, K.C., "Transient Finite Element Computations on 65,536 Processors: The Connection Machine," Report No. CU-CSSC-89-01, Center for Space Structures and Controls, University of Colorado, February 1989, to appear in *International Journal on Numerical Methods in Engineering*, 1990.
26. Farhat, C., A simple and efficient automatic finite element decomposer. *Computers & Structures*, 1988, 28.
27. Park, K. C. and Belvin, W. K., "A Partitioned Solution Procedure for Control-Structure Interaction Simulations," to appear in *J. Guidance, Control and Dynamics*, 1990.
28. Belvin, W. K. and Park, K. C., "Computer Implementation of Analysis and Optimization Procedures for Control-Structure Interaction Problems," Proc. the 1990 AIAA Dynamics Specialist Conference, Paper No. AIAA-90-1194, Long Beach, Calif., 5-6 April 1990.
29. Anderson, E., Moore, D., Fanson, J. and Ealey, M., Development of an active member using piezoelectric and electrostrictive actuation for control of precision structures. AIAA Paper No. 90-1085, Proc. of the 31st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Long Beach, CA, April 1990, pp. 2221-2233.
30. Matsunaga, S. Miura, K. and Natori, M., A construction concept of large space structures using intelligent/adaptive structures. AIAA Paper No. 90-1128, Proceedings of the 31st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Long Beach, CA, April 1990, pp. 2298-2305.
31. Takahara, K., Kuwano, Shigesara, M. Katoh, T., Motohashi, S. and Natori, M., Piezo linear actuators for adaptive truss structures. In *Adaptive Structures*, ed. B.K. Wada, ASME, 1989, pp. 83-88.

32. de Luis, J. and Crawley, E., Experimental results of active control on a prototype intelligent structure. AIAA Paper No. 90-1163, Proceedings of the 31st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Long Beach, CA, April 1990, pp. 2340-2350.
33. Rogers, C. and Ramaseshan, A., Investigation of embedded actuators using generalized laminated plate theory. AIAA Paper No. 90-1168, Proceedings of the 31st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Long Beach, CA, April 1990.
34. Burke, S. and Hubbard, J.E., Active vibration control of a simply-supported beam using a spatially distributed actuator. *IEEE Control Systems Magazine*, August 1987, 7, (6), 25-30.
35. Arbib, M.A., *Brains, Machines, and Mathematics*, 2nd Edition, Springer-Verlag, 1987.

Structural Tailoring and Feedback Control Synthesis: An Interdisciplinary Approach

W. Keith Belvin*

NASA Langley Research Center, Hampton, Virginia 23665

and

K. C. Park†

University of Colorado, Boulder, Colorado 80309

Structural tailoring provides an attractive method to optimize the performance of actively controlled space structures. However, the simultaneous optimization of control gains and structural properties often becomes prohibitively expensive for large systems, and physical insight is often lost in the resulting control law. This paper presents a method for optimization of the closed-loop structural system using only structural tailoring. Optimal linear quadratic regulator (LQR) control theory is used with weighting matrices chosen based on physical considerations. The LQR control law depends only on two scalar gains and the structural properties. Hence, the closed-loop performance can be expressed in terms of the structural parameters. Results are given for a beam and a truss beam to show the simplicity of the method and the importance of structural tailoring to increase dynamic performance and to reduce control effort.

Nomenclature

A	= state matrix
a_j	= real part of the j th circular frequency
B	= control influence matrix
b_j	= imaginary part of the j th circular frequency
C	= damping matrix
D	= actuator influence matrix
E	= sensor influence matrix
E	= Young's modulus
f	= disturbance force vector
G	= equal to one-half of the critical damping matrix
G	= shear modulus
I	= identity matrix
i	= equal to $\sqrt{-1}$
J	= scalar performance/cost index
K	= stiffness matrix
l	= beam length
M	= mass matrix
P	= Riccati matrix for optimal control
Q	= state weighting matrix
q	= structural displacement vector
$\dot{q}$	= structural velocity vector
R	= control force weighting matrix
T	= eigenvector matrix
t	= beam thickness
U	= triple matrix product, Eq. (20)
u	= control force vector
V	= proportional control gain matrix
W	= rate control gain matrix
x	= state vector
$\dot{x}$	= temporal derivative of state vector
y	= system output
z	= modal displacement vector

$\dot{z}$	= modal velocity vector
α	= scalar weighting of strain energy
β	= scalar weighting of kinetic energy
γ	= modal force participation coefficient
ω_j	= j th circular frequency
Λ	= diagonal eigenvalue matrix
η	= scalar proportional control gain
ζ	= scalar rate control gain
ρ	= mass density

I. Introduction

APPLICATIONS of optimal designs in aerospace structural engineering are becoming increasingly important due to the high cost of spacecraft, more stringent performance requirements, and increasing emphasis on structural safety for space deployment. These considerations are particularly important in the design of space structures that employ active feedback control. A number of recent studies have considered the simultaneous optimization of both structure and control system, as can be found in Refs. 1-10. These works have demonstrated that tailoring of the structural mass and stiffness can improve performance and reduce the cost of active control. However, conventional control/structure optimization usually involves a large number of control gains which become part of the overall system design variables. For large problems, the number of control and structural design variables can make simultaneous optimization prohibitively expensive.

A recent approach to simultaneous control structure optimization, found for example in Refs. 8-10, has involved the use of optimal control theory with weighting matrices chosen to be functions of the structural mass and stiffness matrices. This method yields a control law that depends only on structural parameters and several scalar gains. Although such an approach does not guarantee the "best" control law, it does provide the advantage of reducing the number of design variables as compared to the conventional control/structure optimization procedure. Unfortunately, even when the control law is expressed in terms of the structural parameters, simultaneous optimization of controls and structures often reduces to a set of nonlinear constrained equations, which are difficult to interpret in terms of the underlying physics.

Presented as Paper 88-2206 at the AIAA/ASME/ASCE/AHS 29th Structures, Structural Dynamics, and Materials Conference, Williamsburg, VA, April 18-20, 1988; received July 27, 1988; revision received Dec. 12, 1988. Copyright © 1988 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

*Structural Dynamics Division. Member AIAA.

†Professor of Aerospace Engineering. Member AIAA.

For successful interdisciplinary control structure optimization, physical insight into the effect of structural tailoring on the closed-loop system is needed. To this end, we propose using optimal control theory with the weighting matrices of Refs. 8-10 in conjunction with modal-space control.¹¹ A close-to-optimum solution for the control law that yields substantial physical insight into the proper structural tailoring objective is presented. The paper shows that weighting matrices that minimize system energy in the performance cost functional result in control gains that exploit the intrinsic characteristics of the structural stiffness and mass matrices. Hence, optimization of the closed-loop system can be performed using only structural tailoring. A structural tailoring objective is derived that is simple enough to be used in the initial design of actively controlled structures. Results for a simple beam and a more complicated truss structure are used to show the simplicity of the proposed structural tailoring procedure as well as the importance of tailoring structures to increase system performance and to decrease control effort.

II. Physical Choices for Optimal Control Weighting Matrices

Optimal linear quadratic (LQ) design techniques are frequently used to synthesize control laws for multi-input, multi-output systems. The synthesized control law, u , is nonlinearly related to matrices that weight the cost of performance and control. The following paragraphs describe a choice of weighting matrices based on physical considerations that were also used in Refs. 8-10. First, the governing equations are described, then, the energy of the closed-loop system is used to choose the weighting matrices.

A. Governing Equations for Linear Quadratic Regulator Design

For the present discussion, we consider an unforced structural system with full state feedback control. The objective of the control law, known as a regulator, is to suppress vibrations which may have been induced by rigid body slewing or external disturbances. The equations of motion for this system may be written in discrete form as

$$M\dot{q} + C\dot{q} + Kq = Du, \quad q(0) = q_0, \quad \dot{q}(0) = \dot{q}_0 \quad (1)$$

Equation (1) may also be written in first-order, state variable form as

$$\dot{x} = Ax + Bu \quad y = Ex \quad (2)$$

where

$$A = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ M^{-1}D \end{bmatrix}, \quad x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

For the regulator design, we wish to minimize the response of the closed-loop system and the control effort simultaneously. This may be expressed using the following quadratic cost functional

$$J = \int_0^{\infty} \frac{1}{2} (x^T Q x + u^T R u) d\tau \quad (3)$$

where Q and R are weighting matrices which are based on the relative importance of each state and control force, respectively. The only restrictions on Q and R are that Q must be at least semidefinite, and R must be positive definite.

To minimize the preceding functional, the optimal control force can be derived as given in basic texts, such as in Ref. 12, to be

$$u = -R^{-1}B^T P x \quad (4)$$

where P is the only positive definite solution to the steady-

state matrix Riccati equation

$$0 = Q + PA + A^T P - PBR^{-1}B^T P \quad (5)$$

Thus, it is obvious that the control gain $(-RB^T P)$ is nonlinearly related to the weighting matrices Q and R . Finding the "best" choice of Q and R can be one of the more difficult steps in the feedback control design process. Simultaneous optimization of structural properties as well as elements of the Q and R matrices becomes prohibitively expensive for large problems. In addition, it frequently leads to a total loss of physical insight into the resulting control gains. However, by choosing the weighting matrices a priori to yield consistent physical quantities in the cost index of Eq. (3) (e.g., energy, forces, displacements), one can arrive at a control law that preserves physical insight.

B. Minimization of Energy

The objective of many optimization problems involves minimizing the system energy. For structure/control problems, if the stiffness matrix is positive definite (i.e., there are no rigid body modes), a logical choice of weighting matrices to minimize the energy of the structure and the quasistatic "work" of the controller is

$$Q = \begin{bmatrix} \alpha K & 0 \\ 0 & \beta M \end{bmatrix} \quad R = [D^T K^{-1} D] \quad (6)$$

The coefficients α and β weight the structure's strain and kinetic energy. The work of the controller is implicitly weighted by α and β [$\alpha, \beta \geq 0$ and $(\alpha + \beta) > 0$]. Minimization of the controller work is relevant where both limited force and stroke are available as with inertial mass actuators. For cases where the stiffness matrix is not positive definite, rigid body motion could be removed by a reduction technique and the controller designed for flexible modes of vibration only. Equation (6) differs slightly from that given in Refs. 8-10 in that only two scalar weights are used.

Solution of the matrix Riccati equation with the weighting matrices of Eq. (6) yields a control law which depends only on the scalar weights α and β and the structural mass and stiffness matrices. As will be shown in the next section, a closed-form solution for the optimal control law is available when the actuator influence matrix, D , is full rank.

III. Closed-Form Solution of the Riccati Equation

A closed-form solution of the Riccati equation can be easily derived for the preceding weighting matrices when the number of actuators is equal to the number of model degrees of freedom (D is full rank). Meirovitch¹¹ has exploited the case of D being full rank to develop modal-space control when the number of actuators is equal to the number of controlled modes. This paper reviews the case where D is full rank because it so clearly illustrates the physical aspects of the control law when the weighting matrices of Eq. (6) are used. First the solution is discussed in spatial variables and then in modal form.

A. Spatial Form of the Energy-Based Control Law

If we restrict our attention to the case where $C = 0$ (i.e., no structural damping) substituting Eqs. (2) and (6) into Eq. (5) yields

$$P = \begin{bmatrix} 2\zeta G & \eta M \\ \eta M & \zeta MK^{-1}G \end{bmatrix} \quad (7)$$

where

$$\eta = \sqrt{1 + \alpha} - 1$$

$$\zeta = \sqrt{2\eta + \beta}$$

$$G = M^{-1/2} [M^{-1/2} K M^{-1/2}]^{-1/2} M^{1/2}$$

Equation (7), when substituted into Eq. (4), yields the optimal control law vector to be

$$\mathbf{D}\mathbf{u} = -\eta\mathbf{K}\mathbf{q} - \zeta\mathbf{G}\dot{\mathbf{q}} \quad (8)$$

Theoretically, with $\mathbf{D}$ a full-rank matrix, the closed-loop eigenvalues and eigenvectors can be placed anywhere. Hence it is interesting to note where the optimal control law places the eigenvalues and eigenvectors when minimization of energy is used as a cost index.

The first term of Eq. (8) yields a force directly proportional to the internal stiffness of the structure. Note that, as α is increased, we increase the speed of response of the structure by essentially multiplying the eigenvalues of the structure by $(1 + \eta)$. The second term in Eq. (8) produces equal damping in all modes of the structure. Critical damping for a structure is given by $2G$; thus, the damping ratio for each mode of the closed-loop system is

$$\xi = \sqrt{(\eta + \beta)/[2(1 + \eta)]} \quad (9)$$

The pole locations are found by multiplying each of the open-loop poles ($\pm i\omega_j$) by a scalar, such that the closed-loop poles ($a_j \pm ib_j$) are given by

$$a_j \pm ib_j = [\omega_j\sqrt{1 + \eta}] [-\xi \pm i\sqrt{1 - \xi^2}] \quad (10)$$

Thus choosing weighting matrices of the form given by Eq. (6) moves the poles in a manner that yields equal damping ratios for all modes. The closed-loop eigenvectors are unchanged from the open-loop vectors when $\mathbf{D}$ is full rank. Rarely will the number of actuators be equal to the number of spatial degrees of freedom; hence, the next section examines a reduced-order modal model.

B. Modal Form of the Energy-Based Control Law

Equation (2) may be transformed from a physical (spatial) basis to a modal basis by the eigenvector matrix $\mathbf{T}$, which is orthonormal with respect to $\mathbf{M}$. Applying the transformation $\mathbf{q} = \mathbf{T}\mathbf{z}$ one can derive the first order modal equations to be

$$\dot{\mathbf{z}} = \bar{\mathbf{A}}\mathbf{z} + \bar{\mathbf{B}}\mathbf{u} \quad (11)$$

where

$$\bar{\mathbf{A}} = \begin{bmatrix} 0 & \mathbf{I} \\ -\Lambda & -\mathbf{T}^T\mathbf{C}\mathbf{T} \end{bmatrix}, \quad \bar{\mathbf{B}} = \begin{bmatrix} 0 \\ \mathbf{D} \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} \dot{\mathbf{z}} \\ \mathbf{z} \end{bmatrix}$$

in which Λ is the diagonal eigenvalue matrix and $\bar{\mathbf{D}} = \mathbf{T}^T\mathbf{D}$. The matrix $\bar{\mathbf{D}}$ is a square matrix if the number of actuators is equal to the number of modes used in the transformation.

The quadratic functional to be minimized may also be written in modal coordinates as

$$\mathbf{J} = \int_{t_0}^{t_1} \frac{1}{2} (\dot{\mathbf{z}}^T \bar{\mathbf{Q}} \dot{\mathbf{z}} + \mathbf{u}^T \mathbf{R} \mathbf{u}) d\tau \quad (12)$$

where

$$\bar{\mathbf{Q}} = \mathbf{T}^T \mathbf{Q} \mathbf{T} = \begin{bmatrix} \alpha\Lambda & 0 \\ 0 & \beta\mathbf{I} \end{bmatrix}$$

and $\mathbf{R}$ is written in modal form as $[\bar{\mathbf{D}}^T \Lambda^{-1} \bar{\mathbf{D}}]$.

Under the restriction that $\mathbf{C} = 0$ and the number of actuators is equal to the number of modes in the modal model, a closed-form solution for the modal Riccati equation

$$0 = \bar{\mathbf{Q}} + \bar{\mathbf{P}}\bar{\mathbf{A}} + \bar{\mathbf{A}}^T\bar{\mathbf{P}} - \bar{\mathbf{P}}\bar{\mathbf{B}}\mathbf{R}^{-1}\bar{\mathbf{B}}^T\bar{\mathbf{P}} \quad (13)$$

is

$$\bar{\mathbf{P}} = \begin{bmatrix} 2\zeta\Lambda^{1/2} & \eta\mathbf{I} \\ \eta\mathbf{I} & \zeta\Lambda^{-1/2} \end{bmatrix} \quad (14)$$

Equation (14) yields the optimal modal control law vector to be

$$\bar{\mathbf{D}}\mathbf{u} = -\eta\Lambda\mathbf{z} - \zeta\Lambda^{1/2}\dot{\mathbf{z}}$$

or in spatial coordinates the control law vector is

$$\mathbf{D}\mathbf{u} = -\eta\mathbf{D}\bar{\mathbf{D}}^{-1}\Lambda\mathbf{T}^T\mathbf{M}\mathbf{q} - \zeta\mathbf{D}\bar{\mathbf{D}}^{-1}\Lambda^{1/2}\mathbf{T}^T\mathbf{M}\dot{\mathbf{q}} \quad (15)$$

Note that Eq. (15) reduces to Eq. (8) when no modes are truncated (i.e., $\mathbf{T}$ is a square matrix, $\mathbf{D}\bar{\mathbf{D}}^{-1} = \mathbf{T}^{-T} = \mathbf{M}\mathbf{T}$).

By choosing the weighting matrices to yield consistent physical units in the quadratic functional, the control depends only on two scalar gains and the structural properties. Even for the more general case when the number of actuators is not equal to the number of degrees of freedom in the discrete model (i.e., when no closed-form solution is generally available), the control will still be dependant only on two scalar gains and the structural properties. Such a control law permits more physical insight into the closed-loop system and, as will be discussed in the following sections, is amenable to optimization by structural tailoring.

IV. Structural Tailoring to Optimize Closed-Loop Performance

To minimize the functional of Eq. (3) with respect to some set of design variables, the first step involves evaluation of the quadratic cost function which becomes the minimization objective in the optimization. Since a closed-form solution for the Riccati equation exists, the cost function may be expressed in terms of the initial state, that is, the state when the control system is activated.

$$\mathbf{J} = \int_{t_0}^{t_1} \left(\frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u} \right) d\tau = \frac{1}{2} \mathbf{x}_0^T \mathbf{P} \mathbf{x}_0 \quad (16)$$

Thus, to evaluate the cost function, we must estimate the initial state of the structure. Fortunately, for open-loop structures, we can estimate the initial state assuming the structure is excited by external disturbances. The following sections describe a quasistatic measure of the structural response due to unknown disturbances. This approximate measure of system response is then used to compute the magnitude of the cost function given by Eq. (16).

A. Approximation of Closed-Loop Response due to Unknown Disturbance

To minimize Eq. (16), the peak magnitude of the displacement vector $\mathbf{q}$ and velocity vector $\dot{\mathbf{q}}$ have been approximated based on the open-loop modal response due to an external step load. Given an undamped structural system subjected to an external force

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{f}$$

the peak response can be computed by a summation of the linear modes of vibration. For a step load, the peak displacement and velocity can be easily found to be proportional to

$$\mathbf{q} \propto \mathbf{T}\Lambda^{-1}\mathbf{T}^T\mathbf{f}, \quad \dot{\mathbf{q}} \propto \mathbf{T}\Lambda^{-1/2}\mathbf{T}^T\mathbf{f} \quad (17)$$

To estimate the effects of $\mathbf{f}$ in this study, we expand $\mathbf{f}$ into an orthonormal basis of the form

$$\mathbf{f} = \mathbf{M}^{1/2}\mathbf{T}\boldsymbol{\gamma} \quad (18)$$

where $\boldsymbol{\gamma}$ is a vector of modal-force participation coefficients. The modal-force participation coefficients are dependant on the disturbance; however, if no disturbance information is known, equal participation of all modes can be assumed. For the study herein, $\gamma_j = 1$ for all modes.

Substituting Eq. (18) into (17) and writing in state vector form we derive

$$\dot{x} = \begin{Bmatrix} \dot{q} \\ \ddot{q} \end{Bmatrix} \propto \begin{Bmatrix} T\Lambda^{-1}U\gamma \\ T\Lambda^{-2}U\gamma \end{Bmatrix} \quad (19)$$

where

$$U = T^T M^{-1} T \quad (20)$$

Equation (19), albeit approximate, gives a measure of the peak magnitude of the displacement and velocity response of the structural system. Expressing f in the orthonormal basis of Eq. (18) is approximate to the extent that f can be represented by a truncated set of eigenvectors. Expansion of f in the eigenvector basis makes interpretation of the underlying physics much more transparent.

B. Approximate Cost Function

Equation (16) can now be explicitly stated in terms of structural parameters, two scalar gains and the assumed modal-force coefficients γ . Substituting Eqs. (19) and (14) into (16) gives the following cost function

$$J \propto \left(\eta + \frac{3}{2} \zeta \right) \gamma^T U^T \Lambda^{-3/2} U \gamma \quad (21)$$

Thus, the optimal tailoring for the case of an actuator for each controlled mode is that which will minimize $U^T \Lambda^{-3/2} U$, which is inversely proportional to the frequency cubed. These results are discussed more fully in the next section with the aid of numerical examples.

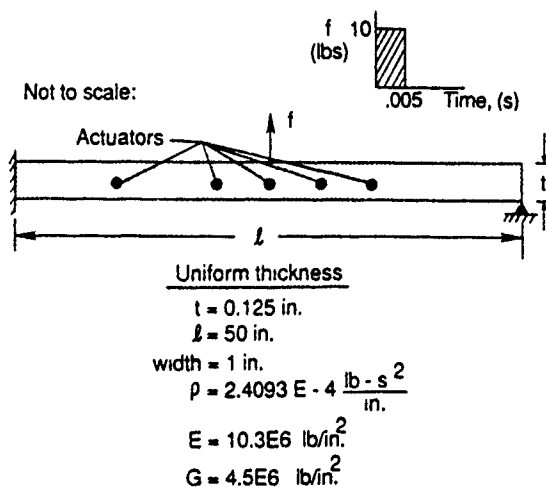


Fig. 1 Uniform clamped-pinned beam.

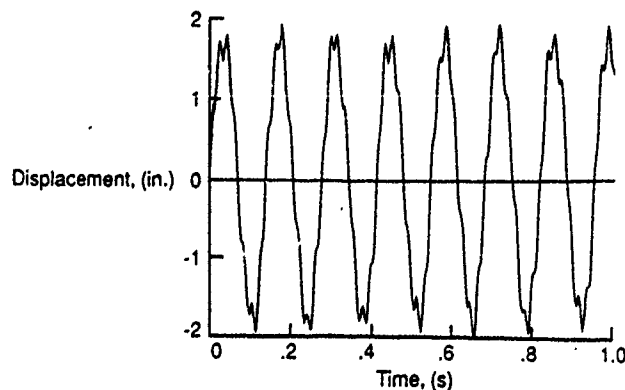


Fig. 2 Open-loop response of uniform beam.

V. Results and Discussion

The previous section described an approximate cost function, which minimizes the control energy necessary to achieve vibration suppression. The objective herein is to minimize Eq. (21) by structural tailoring. This section presents results for a clamped-pinned beam and a cantilevered truss beam. Both open- and closed-loop simulations are presented to demonstrate the effect of structural tailoring. The optimization procedure is discussed first and followed by the numerical examples.

A. Structural Tailoring Optimization Procedure

Minimization of Eq. (21) has been performed by tailoring appropriate structural design variables, such as member thickness, to maximize the weighted frequency cubed of selected modes. The contribution of each mode to the objective function is $(U_j \gamma_j / \omega_j^{3/2})^2$. (Note U_j is calculated from Eq. (20) with T consisting of only the j th eigenvector.) This weighting readily shows that the low-frequency modes contribute most to the cost index. Hence, usually only the low-frequency modes need to be considered in the tailoring process.

Constrained optimization was performed using the ADS¹³ system of optimization routines. Eigenvalue sensitivities were computed by finite differences. Since the tailoring procedure is based on the open-loop eigenvalues and eigenvectors, only a finite-element modeling algorithm to compute mass and stiffness matrices, a real eigenvalue algorithm for eigenproblem analysis, and an optimization algorithm for constrained function minimization are required. The following results have also used an implicit integration algorithm and a modal control law synthesis algorithm for transient simulation of the open- and closed-loop systems. All simulations have been computed in spatial coordinates.

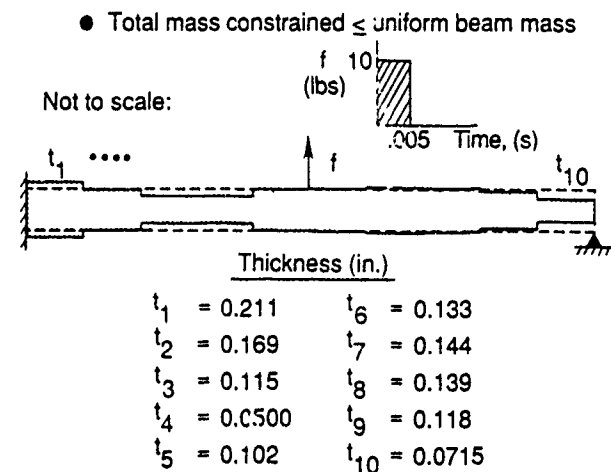


Fig. 3 Tailored clamped-pinned beam.

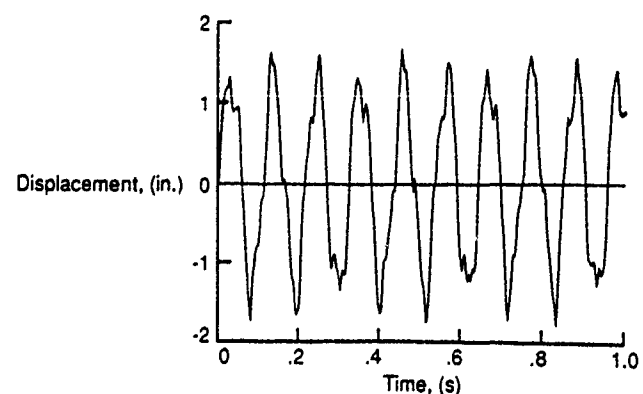


Fig. 4 Open-loop response of tailored beam.

B. Clamped-Pinned Beam Results

The beam shown in Fig. 1 has been studied to determine the effects of structural tailoring on the dynamic performance and feedback control of the beam. Ten Timoshenko beam elements were used in the discrete model. The beam is subjected to a step load at the center of 10 lb for 5 ms. Figure 2 shows the undamped transient response of the uniform beam to this loading condition.

The thickness of each element was tailored to minimize the cost index of Eq. (21). Only the first three modes were considered as the higher modes contributed little to the cost function. An active constraint in the minimization procedure was that the total mass of the tailored beam was to be less than or equal to the mass of the uniform beam. Figure 3 shows the tailored beam and lists the thickness of each element. The undamped transient response of the tailored beam, depicted in Fig. 4, shows a significant reduction in vibration amplitude.

To determine the effect of structural tailoring on the work done by the actuators, the control law given by Eq. (15) was used. Five modes were used to compute the optimal control law for the five actuators located as shown in Fig. 1. For simulation purposes, a perfect estimation of the state has been assumed. The control is turned on at 0.1 s and must reduce the vibration amplitude to less than 0.10 in. within 0.5 s after active control begins. For the examples herein, only rate feedback was used, i.e., $\alpha = 0$.

Figure 5 shows the closed-loop transient response of the uniform and tailored beams. Both beams satisfy the preceding performance requirement. Figure 6 shows a reduction of 37% in total control work for the tailored beam. Thus, the work of the actuators can be significantly reduced for some actively controlled structures by simply tailoring the structure to minimize a function inversely proportional to the vibration frequency cubed. Although such tailoring may be advantageous to other control laws, the physical based control law presented

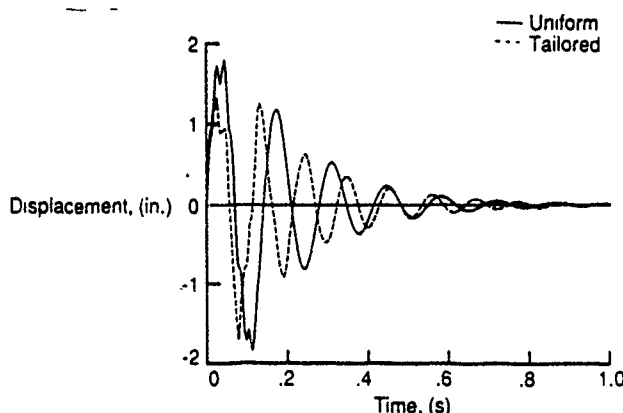


Fig. 5 Closed-loop response of uniform and tailored beams.

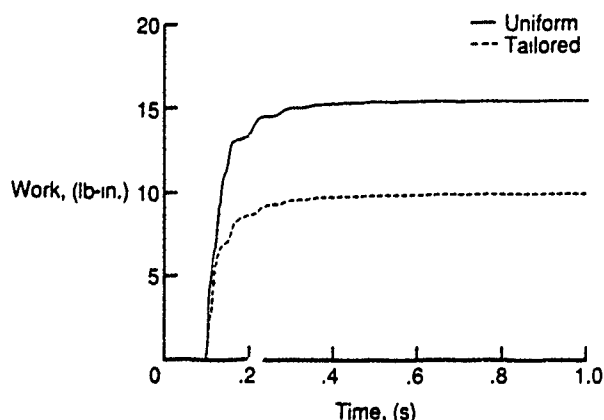


Fig. 6 Controller work of uniform and tailored beams.

herein maximizes the effect of tailoring to reduce control work.

C. Cantilever Truss-Beam Results

The truss beam shown in Fig. 7 has been used to demonstrate that even large-order systems can be easily tailored to reduce control cost using the method proposed in this study. The three longeron, single-laced truss was modeled by finite elements with one Timoshenko beam element from joint-to-joint. The model had 165 nodes and 990 degrees of freedom. All members are tubular with the inside diameter set equal to 75% of the outside diameter. Three design variables, the outside diameters of the batten, and diagonal and longeron elements are used to tailor the structure. In addition to the mass constraint mentioned previously, the first pinned-pinned frequency of each element was constrained to be more than 100 times the first global bending frequency.

Table 1 lists the truss-beam properties as well as the nominal and tailored tube diameters. The nominal outside tube diameters were chosen such that the first pinned-pinned beam frequency of each element was 100 Hz. The beams are subjected to a step load at two points as shown in Fig. 7 of 10 lb magnitude for a duration of 100 ms. Figure 8 shows the undamped response of the uniform and tailored beam. Again, the tailored beam shows a substantial decrease in vibration amplitude.

A seven-mode control law was used to study the control cost for each truss beam necessary to reduce the vibration amplitude to 0.025 in. within 10 s after active control began. Six

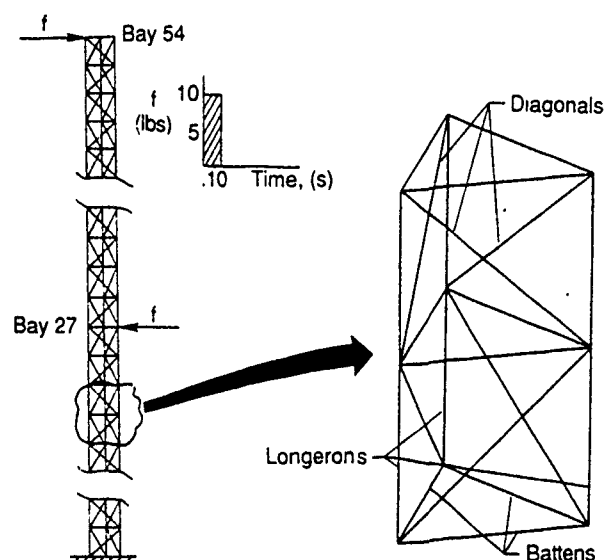


Fig. 7 Cantilevered truss beam.

Table 1 Truss-beam properties

Outside tube diameter (in.)	Nominal	Tailored
longeron	0.789	1.717
diagonal	1.707	1.284
batten	0.918	0.640

Material properties

$$\rho = 1.5285 \times 10^{-4} \text{ lb} \cdot \text{s}^2/\text{in.}^4$$

$$E = 40.0 \times 10^6 \text{ lb/in.}^2$$

$$G = 2.4 \times 10^6 \text{ lb/in.}^2$$

Element lengths

$$\text{longeron} = 44.25 \text{ in.}$$

$$\text{diagonal} = 65.09 \text{ in.}$$

$$\text{batten} = 47.73 \text{ in.}$$

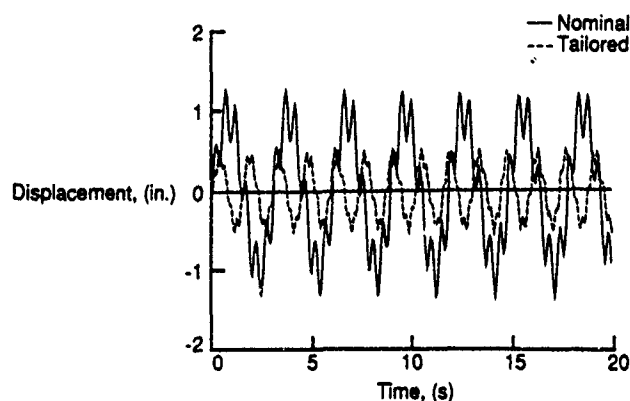


Fig. 8 Open-loop response of nominal and tailored truss beams.

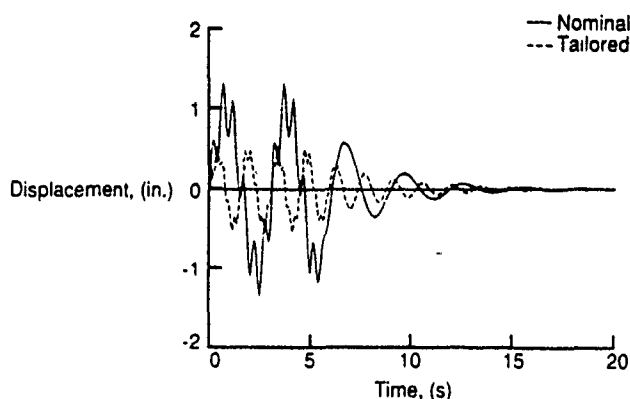


Fig. 9 Closed-loop response of nominal and tailored truss beams.

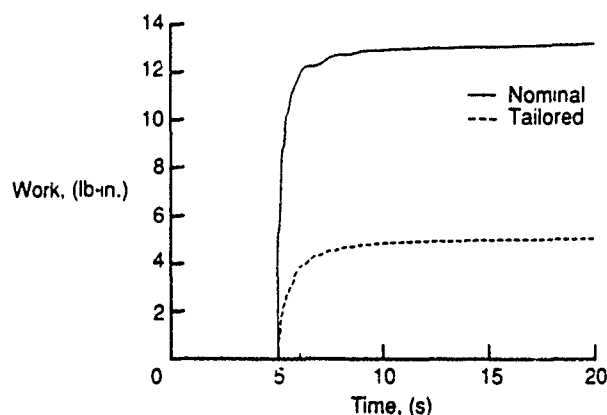


Fig. 10 Controller work of nominal and tailored truss beams.

actuators were located in both bending planes at bays 27, 53, and 54. One additional actuator was located in the plane of the external force at bay 28. Active control was initiated 5 s after the load was applied. Figure 9 shows that both truss beams meet the performance requirement; however, the tailored beam requires 56% less total actuator work as shown in Fig. 10. Thus, structural tailoring can produce substantial reductions in controller energy. Moreover, the structural procedure presented in this study is simple enough to be applied to large systems.

VI. Summary

Structural tailoring of actively controlled structures has significant advantages over pure feedback control. However,

considerable effort is required to incorporate physical insight as part of the synthesis process in order to take full advantage of combined structural tailoring and active feedback control. Results herein have shown that a set of physically based weighting matrices in the LQR performance criterion leads to a control law that has direct physical meaning. This physical understanding of the control law leads to the development of the proper structural tailoring objective. For minimization of controller energy, the tailoring objective is inversely proportional to the open-loop vibration frequencies cubed. This objective is simple enough to be applied early in the structure/control design process.

Numerical results for a beam subjected to an external disturbance have been presented. Structural tailoring of the beam thickness reduced the control energy by 37% from that of the uniform beam. In addition, numerical results for a more realistic space truss beam were presented. Structural tailoring of the truss beam reduced the actuator work by 56%.

Future studies are needed to address more general control laws where the number of actuators differs from the number of model degrees of freedom. Such control laws will result even for modal-space control when actuator dynamics are considered. In this case, the control laws will produce closed-loop eigenvectors that differ from the open-loop eigenvectors.

Acknowledgment

It is a pleasure to acknowledge support of the second author by the Air Force Office of Scientific Research for the work reported herein under Grant F49620-87-C-0074. He wishes to thank Dr. Anthony K. Amos for his interest and encouragement during the course of the present work.

References

- ¹Onoda, J., and Haftka, R. T., "An Approach to Structure/Control Simultaneous Optimization for Large Flexible Spacecraft," *AIAA Journal*, Vol. 25, No. 8, 1987, pp. 1133-1138.
- ²Haftka, R. T., Martinovic, Z. N., and Hallauer, W. L., Jr., "Enhanced Vibration Controllability by Minor Structural Modifications," *AIAA Journal*, Vol. 23, No. 8, 1985, pp. 1260-1266.
- ³Hale, A. L., Lisowski, R. J., and Dahl, W. E., "Optimal Simultaneous Structural and Control Design of Maneuvering Flexible Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 8, No. 10, 1985, pp. 86-93.
- ⁴Junkins, J. L., and Rew, D. W., "Unified Optimization of Structures and Controllers," *Large Space Structures: Dynamics and Controls*, Springer-Verlag, New York (to be published).
- ⁵Hanks, B. R., and Skelton, R. E., "Designing Structures for Reduced Response by Modern Control Theory," *AIAA Paper 83-0815*, May 1983.
- ⁶Khot, N. S., Venkayya, V. B., Oz, R. V., Grandhi, R. V., and Eastep, F. E., "Optimal Structural Design with Control Gain Norm Constraint," *AIAA Paper 87-0019*, Jan. 1987.
- ⁷Khot, N. S., Grandhi, R. V., and Venkayya, V. B., "Structural and Control Optimization of Space Structures," *Proceedings of the AIAA/ASME/ASCE/AHS 28th Structures, Structural Dynamics, and Materials Conference*, AIAA, New York, 1987, pp. 850-860; also *AIAA Paper 87-0939*.
- ⁸Venkayya, V. B., and Tischler, V. A., "Frequency Control and its Effect on the Dynamic Response of Flexible Structures," *AIAA Journal*, Vol. 23, No. 11, 1985, pp. 1768-1774.
- ⁹Miller, D. F., Venkayya, V. B., and Tischler, V. A., "Integration of Structures and Controls—Some Computational Issues," *Proceedings of the 24th Conference on Decision and Control*, AIAA, New York, 1985, pp. 924-931.
- ¹⁰Becus, G. A., Lui, C. Y., Venkayya, V. B., and Tischler, V. A., "Simultaneous Structural and Control Optimization via Linear Quadratic Regulator Eigenstructure Assignment," *NASA CP-2488*, Oct. 1987, pp. 225-232.
- ¹¹Meirovitch, L., and Oz, H., "Modal-Space Control of Distributed Gyroscopic Systems," *Journal of Guidance and Control*, Vol. 5, No. 1, 1982, pp. 140-150.
- ¹²Kwakernaak, H., and Sivan, R., *Linear Optimal Control Systems*, Wiley-Interscience, New York, 1972.
- ¹³Vanderplaats, G. N., "ADS-A Fortran Program For Automated Design Synthesis—Version 1.10," *NASA CR 177985*, Sept. 1985.

Computer Implementation of Analysis and Optimization Procedures for Control-Structure Interaction Problems

W. Keith Belvin¹
Spacecraft Dynamics Branch
NASA/Langley Research Center
Hampton, VA. 23665

and

K. C. Park²
Center for Space Structures and Controls
University of Colorado, Campus Box 429
Boulder, Colorado 80309

Abstract

Implementation aspects of control-structure interaction analysis and optimization by the staggered use of single-discipline analysis modules are discussed. The single-discipline modules include structural analysis, controller synthesis and optimization. The software modularity is maintained by employing a partitioned control-structure interaction analysis procedure, thus avoiding the need for embedding the single-discipline modules into a monolithic program. A software testbed has been constructed as a stand-alone analysis and optimization program and tested for its versatility and software modularity by applying it to the dynamic analysis and preliminary design of a prototype Earth Pointing Satellite. Experience with the in-core testbed program so far demonstrates that the testbed is efficient, preserves software modularity, and enables the analyst to choose a different set of algorithms, control strategies and design parameters via user software interfaces. Thus, the present software architecture is recommended for adoption by control-structure interaction analysts as a preliminary analysis and design tool.

Introduction

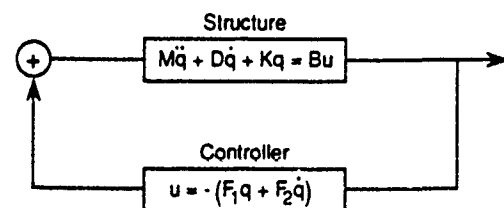
Computer simulations have become an essential part of design activities, especially for the interdisciplinary design tasks such as controller-structure interaction dynamics. Traditionally, computer simulations of design activities that involve only a specific discipline, i.e., aerodynamic drag and lift, structural design, thermal problems, etc., have been carried out with a tightly packaged computer program. As an example, a typical structural analysis computer program has, within a single package, several related capabilities such as linear stress, vibration, buckling and perhaps nonlinear analyses.

As the design complexities increase to include several disciplines such as coupled structural-thermal design, aerodynamics-structural coupling and control-structure in-

teraction (CSI) design, the general tendency has been to embed the required interaction phenomena into an existing single-discipline analysis and design program. As a result, most designs which consider both the controller and structure as design variables, see Fig. 1, have been performed using an ad hoc collection of discipline specific software modules. Such software tools were originally developed for the solution of single-discipline problems (e.g. control law synthesis, finite element structural modeling, etc.). The use of these tools has required specialized interfaces to be developed, e.g. Refs. 1-2.

Integration of such single-discipline analysis codes by means of a common data base manager provides for immediate usage of existing software. However, the program is usually hardwired to a few design methods thereby losing versatility. Moreover, the high cost associated with repeated cold starts of structural analysis or controller synthesis packages discourages parameter studies. Most importantly, the use of ad hoc single-discipline programs may distort the physics associated with the interaction phenomenon because reduced order modeling is usually employed.

The present paper presents an architecture for the analysis and optimization of structure-controller problems that alleviates some of the computational problems associated with the embedded implementation of structures into a controls package or vice-versa. The objectives of



In addition to the controller gains F_1 , F_2 , the structural parameters M , D , K , and B can be changed to meet performance/robustness requirements.

¹ Structural Dynamics Division, NASA Langley Research Center. Member AIAA.

² Professor of Aerospace Engineering, University of Colorado. Member AIAA.

Figure 1. Integrated Controller and Structure Design

the architecture are to employ existing discipline specific analysis and design software modules in a loosely coupled manner, to exploit sparse matrix utilities that have proved so essential for efficient implementation of many engineering analysis activities, to effect efficient data transfer between structural analysis processors and control synthesis algorithms, and to maintain versatility in the optimization methodology. The present version of the architecture is implemented as an in-core program so that all required structure, control and optimization analyses are performed within one executable program. The in-core data transfer between the control, structure, and optimization modules facilitates user interfaces for new implementations of solution algorithms and control strategies. A distinctive feature of the control-structure interaction transient response analysis capability is the adoption of the partitioned solution approach and second-order observers as described in Refs. 3-5.

The basic structure and controller equations are reviewed with emphasis on computer implementation. Conventional CSI simulation is shown to suffer computational difficulties that can be substantially reduced by the proposed software architecture. A prototype version of the architecture and its capabilities for design is described. In addition, an example problem is presented to demonstrate the interdisciplinary controls and structures design of an Earth Pointing Satellite. Summary remarks and future directions for controlled structure design are also given.

Implementation Considerations

A typical control-structure interaction system can be represented as shown in Fig. 2. In addition to the controlled structure model, an observer (or state estimator) is used to estimate the full structural state from the measured output z . The spatially discrete equations of motion for control-structure interaction systems may be described by

$$\left\{ \begin{array}{ll} \text{Structure:} & a) \quad M\ddot{q} + D\dot{q} + Kq = f + Bu + Gw \\ & q(0) = q_0, \quad \dot{q}(0) = \dot{q}_0 \\ \text{Sensor Output:} & b) \quad z = Hx + v \\ \text{Estimator:} & c) \quad \dot{\hat{x}} = A\hat{x} + E\dot{f} + \hat{B}u + L(z - H\hat{x}) \\ & \hat{x}(0) = 0 \\ \text{Control Force:} & d) \quad u = -F\hat{x} \end{array} \right. \quad (1)$$

where

$$x = \begin{Bmatrix} q \\ \dot{q} \end{Bmatrix}, \quad \hat{x} = \begin{Bmatrix} \hat{q} \\ \hat{\dot{q}} \end{Bmatrix}$$

and

$$H = [H_d \ H_v], \quad L = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix}, \quad F = [F_1 \ F_2]$$

The structure model consists of the mass matrix M , the damping matrix D , and the stiffness matrix K . The state independent applied forces to the structure are given by f . B and $\hat{B}$ represent the input influence matrix for actuator

locations whereas G and $\hat{G}$ represent the disturbance locations. The vector q is the generalized displacement, w is a disturbance vector and the vector v is measurement noise. In Eq. (1b), z is the measured sensor output. The matrix H_d is the matrix of displacement sensor locations and H_v is the matrix of velocity sensor locations. The vector v is measurement noise. The state estimator in Eq. (1c) may or may not be model based. The superscript $\sim$ and $\cdot$ denote the estimated states and time differentiation respectively. The input command, u , is a function of the state estimator variables, $\hat{q}$ and $\hat{\dot{q}}$, and F_1 and F_2 are control gains. The observer is governed by A , the state matrix representing the plant dynamics, and L , the filter gain matrix.

There exist a number of theoretical and computational issues associated with the design of control-structure interaction systems described by Eq. (1). Modeling of the dynamic plant and model reduction for control are still unresolved issues. Synthesis of control force gains and state estimators for multi-input and multi-output MIMO systems are still subject to experimental validation. Moreover, stability and performance evaluation of the CSI system design via computer simulation have become a necessity for MIMO design. But perhaps the most demanding of the above issues, from a computational viewpoint, is the integrated design that is possible by simultaneously tuning the structure and the controller to achieve an optimized design (Fig. 1). These issues are discussed in relation to the need for a new software architecture for CSI design in the remainder of this section.

Modeling Issues

Development of the dynamic model of the plant or structure has been the subject of much research. Although finite element based models dominate structural analysis, their role in the study of control-structure interaction analysis should be considered from an implementation viewpoint. To this end, the following paragraphs describe prevailing practices in control system design. For controlled structures, it is shown that the finite element based structure models lead naturally to finite dimensional CSI models.

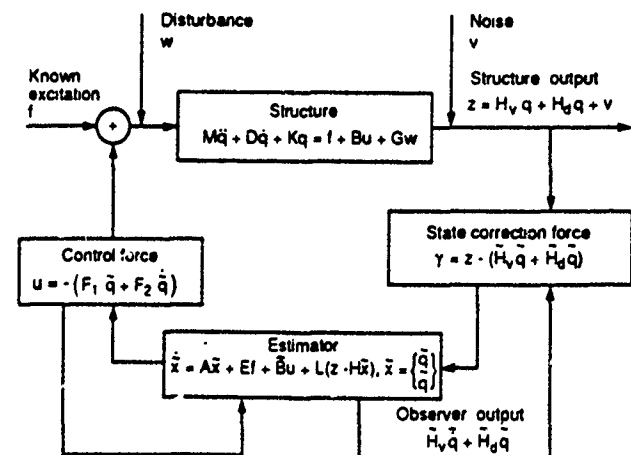


Figure 2. Control Structure Interaction System

The need for designing controllers for coupled MIMO systems prompted the development of modern control theory. In modern control theory, the state-space differential equations are used directly to design the controller. Not only does modern control theory produce a more computationally attractive controller design approach, but it also is advantageous in that internal couplings in the system are not obscured by the classical input-output model. Thus, modern control theory is quite amenable to structural controller design.

Within modern control theory, there exist two distinct modeling methods: distributed parameter control theory and discrete parameter control theory. In distributed parameter control, an infinite dimensional state space is assumed. Partial differential equations and integral equations are typically employed in distributed parameter control. In discrete parameter control, a finite dimensional state space is assumed to be represented by ordinary differential equations. Distributed parameter control theory can be difficult to implement on structural systems because distributed parameter models are difficult to derive except for the simplest of structural geometries such as beams and plates. Also, the computational implementation of distributed parameter control theory is still in a developing state, whereas, with discrete parameter control theory, numerous advances in equation solution techniques have been made.

Discrete parameter modeling within the framework of modern control theory leads naturally to the use of finite element based structural models. Many of the existing ordinary differential equation solution techniques for controls and structures may be used if certain properties of the coupled equations are preserved. Moreover, the force vectors on the right hand side of Eq. (1a) can consist of body forces, surface forces and external point forces. The ability to handle such a wide variety of forces allows the displacement based finite element equations to be employed in the study of fluid-structure, thermal-structure, control-structure and many other interaction problems. For the study of control-structure interaction, discrete points on the structure may be chosen for control action,

$$Bu = B(-F_1 q - F_2 \dot{q}) \quad (2)$$

where the input influence matrix B determines the degrees-of-freedom directly affected by control actions, and u is a vector of actuator control commands. For most practical control-structure interaction systems, the number of control forces is much less than the number of system degrees of freedom. This fact has been exploited in the present simulation algorithms.

Equation (1a) represents a controlled structure which possesses the same attractive computational properties as open-loop structural simulation does provided the state dependent control interaction force is maintained on the right hand side of the second-order equations. As shown in Refs. 3-5, the conventional simulation approach which converts Eq. (1) to a first-order state space representation

$$\dot{x} = A_s x \quad (3)$$

where,

$$A_s = \begin{bmatrix} 0 & I \\ -M^{-1}(K + BF_1) & -M^{-1}(D + BF_2) \end{bmatrix}, x = \begin{Bmatrix} q \\ \dot{q} \end{Bmatrix}$$

needlessly destroys the computational features of the matrices M , D , and K . The 'curse of dimensionality' is aggravated by the first-order representation in Eq. (3) where in addition to doubling the number of equations, the sparsity and symmetry of the coefficient matrices is lost. To circumvent the dimensionality problems with finite element based models, reduced order models for the structure are typically constructed using a truncated modal based realization.

Consider the generalized coordinates p where

$$q = Tp; \quad \dim(p) \ll \dim(q) \quad (4)$$

and T is a matrix consisting of a reduced set of basis vectors. Transforming (1a) with (4) yields

$$T^T M T \ddot{p} + T^T D T \dot{p} + T^T K T p = T^T B u \quad (5)$$

In general, the coefficient matrices of Eq. (5) may not be sparse, however, if T is chosen to be a reduced set of eigenvectors from the eigenvector matrix Φ for the real-symmetric eigenvalue problem

$$(K - M\lambda)\Phi = 0$$

then, the transformation yields

$$\ddot{p} + T^T D T \dot{p} + \Delta p = T^T B u \quad (6)$$

For the special case when $T^T D T$ is diagonal, the transformation not only reduces the model size, but, simultaneously maximizes the coefficient matrix sparsity. If l of the total N eigenvectors are chosen, Eq. (6) is reduced from N to l equations in second-order form or to $2l$ equations in the first-order form

$$\dot{x}_p = \begin{bmatrix} 0 & I \\ -\Delta & -T^T B F_1 T \end{bmatrix} x_p, x_p = \begin{Bmatrix} p \\ \dot{p} \end{Bmatrix} \quad (7)$$

For many applications, the reduced order model in Eqs. (6) or (7) is sufficient for synthesis and simulation studies. However, for structures characterized by high modal density in the bandwidth of interest, even the first-order form becomes computationally intensive. Hence, the second-order form of the CSI finite dimensional models is preferred for control-structure interaction studies. The second-order form also simplifies the implementation aspects by treating the control and structure interaction terms as external disturbances. Similar arguments can be made for the estimator model in Eq. (1c). The implementation of second-order estimator models in Refs. 3-5 shows significant computational advantages by treating the interaction terms as external disturbances.

From the above discussion, it is seen that the finite element analysis module and the control synthesis modules have equal roles in the study of CSI. Maintaining the second-order form of the differential equations for the structure model (and the estimator model if possible) simplifies the implementation of CSI software by partitioning the structure and control computations. In addition, the level of model reduction prior to simulation may be reduced since it becomes unnecessary to embed a first order state model of the structure into a controls simulation package. The implementation of CSI design software should make finite element structural analysis and control synthesis available within the same program.

Integrated Design Issues

In addition to the modeling issues above, the present software architecture has been driven by the distinct differences between CSI analysis and design. Design issues involve not only parameter studies of the controller but also parameter studies of structural variables. Thus, the need exists to perform both structural and controller calculations in a repetitive manner.

To date, most designs which consider both controls and structure design variables have been performed using discipline specific software modules. The use of these tools has requires a database to transfer data from one module to the next as shown in Fig. 3. The architecture of Fig. 3 works well for the analysis of CSI systems, provided model reduction issues are overcome, since the structure model remains constant. The data base architecture also works well for classical control design where the plant remains fixed and the controller parameters are optimized. However, for integrated controller/structure design, the structure model changes repeatedly. In this case the data base would need to be rewritten for every structural parameter change. The implementation of CSI software indicated in Fig. 3 greatly hinders true interdisciplinary design.

To alleviate some of the computational problems associated with integrated design of structures and controllers, an in-core architecture is proposed as shown in Fig. 4. This architecture permits all data to be shared between the control, structure and optimization software. By having ready access to the data from all disciplines in one executable program, repeated calculations pose no difficulties. In fact, the optimizer can govern the flow of calculations without the need to rewrite a data base when the problem objectives or constraints change.

Many of the structure and controller results calculated for the initial set of design variables remain constant during the design process. The proposed software architecture does not require cold starts of the structure or control software modules. Instead, control and structure calculations from the previous set of design variables are continuously available through the in-core architecture. The availability of results from previous iterations of the optimizer greatly reduces the computations required during integrated design parameter studies.

Other advantages exist in using the proposed architecture versus the conventional data base approach. First, the computational speed can be improved using in-core data transfer (i.e. common blocks instead of data bases). Coupling the input/output time savings with algorithms that exploit matrix sparsity and the second-order form of structures equations enable moderate size problems to be solved routinely. Second, the new architecture requires engineers and scientists from both controls and structures disciplines to work more closely. Since they both use the same software tool, a conducive software environment exists for exploring interdisciplinary design problems. Finally, the in-core architecture permits much more flexibility to implement new concepts into software.

Although the available memory (virtual memory) of new computers has grown dramatically in recent years, some very large problems must still be solved out-of-core. Data-base type design codes will continue to be needed to handle very large problems for the foreseeable future. The proposed architecture is targeted for research studies of design methodology for small to moderate size problems. However, as multiple instruction multiple data (MIMD) computers become readily available to the research and design community, the in-core architecture will soon be amenable to quite large problems.

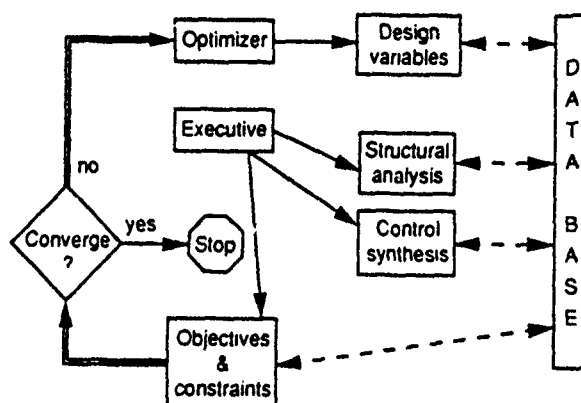


Figure 3. Conventional Architecture for Controlled Structure Design

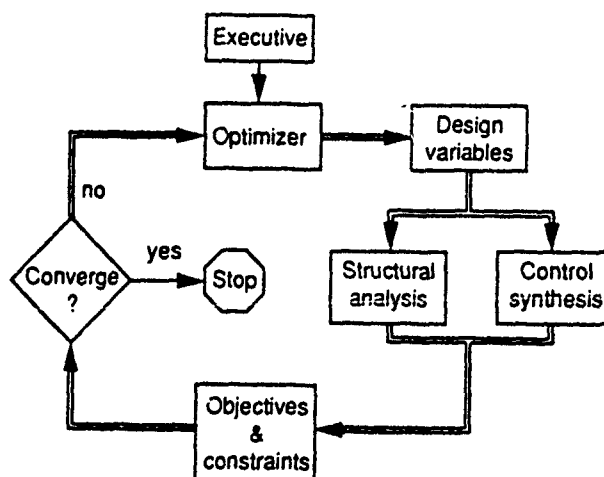


Figure 4. New Architecture for Controlled Structure Design

Prototype CSI Design Software System

A prototype code called Controlled Structure Simulation Software (CS^3) has been developed using public domain software to implement the in-core architecture shown in Fig. 4. The key feature sought in choosing the software is the availability of source code which could be modified to permit in-core data transfer among the different programs. There exist many other possible choices for the optimizer, structural analysis and control synthesis than the ones presented herein.

The following sections describe key features of each module used in CS^3 . It is noted that the executive program is simply one that tests input data to determine whether analysis or design is to be performed. If design is to be performed, the program flow is governed by the optimization module. Otherwise, the appropriate controls or structural analysis is performed. User modules can be easily added to CS^3 which make the code quite useful as a software testbed.

Optimization Module

The optimization path uses the Automated Design Synthesis (ADS) system of subroutines (Ref. 6). The ADS system allows for constrained function minimization by the method of feasible directions, various penalty function methods as well as other techniques. The wide variety of methods allows the user to tailor the optimizer to the problem at hand. Since there exist many good optimizers in the literature, the choice of ADS has been made based on availability of the source code rather than technical considerations.

A number of solution strategies may be chosen within the ADS system to find the minimum of a constrained function both by nonlinear programming and sequential linear programming techniques. The advantage of nonlinear programming minimization methods is the simplicity of the optimization procedure from the user's viewpoint. However, for large numbers of design variables, optimization by nonlinear programming can become prohibitively expensive. An alternative method for optimization of functions, is sequential linear programming as proposed in Refs. 7-10. Sequential linear programming is quite robust to large numbers of design variables, however, some pre-processing (linearization) of the optimization problem is required on the user's part. Since simplicity for the user is a key consideration in choosing the optimizer, nonlinear programming methods are frequently employed whereby the user needs only to supply routines for evaluation of the objective function and constraints and perhaps their gradients. The objective function and constraints are usually dependent on results obtained from the structural and/or control processors.

One of the most important (and expensive) aspects of optimization is the calculation of objective function and constraint gradients with respect to the design variables. For simultaneous structure and controller design, the objective function and constraints may involve calculations of eigenvalues and eigenvectors, closed-loop frequencies and

damping, total weight of the structure and actuators, robustness measures, and/or transient response quantities. There is considerable on-going research into gradient calculation of dynamic systems as found for example in Refs. 11-16. The calculation of gradient information for use by ADS may be either analytic, semi-analytic, or by finite differences. Although some analytic or semi-analytic gradient information may be available, the need for finite-difference calculation of gradients usually arises for integrated control and structure design. If all gradient information is to be computed by finite differences, the ADS program provides an automatic utility for this purpose.

Structural Modeling and Analysis Module

Structural finite element modeling, real-symmetric eigenvalue analysis, and transient response calculations are performed with code called Linear Analysis of Sparse Structures (LASS). LASS was developed to permit finite dimensional models of spacecraft to be easily developed and manipulated. Although a number of linear finite element programs for modeling of structures exist (e.g. NASTRAN¹⁷ and EAL¹⁸), the computational overhead of these large codes and the lack of source code to implement the in-core data architecture led to development of LASS.

LASS uses a sparse matrix utility with a skyline data structure for efficient manipulation of matrix vector equations. The skyline data structure stores fewer matrix elements than a banded storage scheme yet the data structure is still preserved after matrix factorization. Linear beam elements (both Euler and Timoshenko) are used to assemble mass and stiffness matrices for structures of arbitrary geometry. Assembly of the structural mass and stiffness matrices is readily performed by storage of the transformed element matrices into global arrays. A utility for scaling structural properties is included to permit optimization with structural design variables such as the radius of tubular beams.

Three analysis paths exist in LASS; static analysis, transient response analysis and eigenvalue/eigenvector analysis. Static analysis capabilities are limited to the displacement response determination of the structure subject to an applied load. Any combination of forces and moments are permitted.

Transient response capabilities include open-loop and closed-loop response simulation due to time-variant disturbance forces. The algorithms presented in Refs. 4 and 5 are implemented for integration of the ordinary differential equations. The sparse matrix utility and the implicit integration technique permit direct integration of the controlled structure without model reduction. Thus, transient response calculations can be performed without an intermediate eigen-solution to obtain a reduced set of basis vectors. However, the capability does exist in LASS to use a reduced order modal model for transient response simulation. Note that software modularity is a key feature of the proposed testbed architecture, inclusion of new structural models, new control laws or new integration algorithms can be easily performed. The modular style of the transient response utility makes CS^3 a useful software testbed.

Solution of the real-symmetric eigenvalue problem is performed by a sub-space iteration technique. The Jacobi method is used to find all the eigenvalues of the subspace. The eigen-solution algorithms also use the skyline data structure. Eigenvalue convergence tolerances can be controlled by the user. In addition, during optimization, the previous eigenvectors can be used as the initial subspace vectors to increase the convergence rate. Model reduction is virtually inevitable for controller synthesis. Thus, a subset of the smallest eigenvalues and associated eigenvectors are usually computed for model based controller design. The control design capability of CS^2 is discussed next.

Control Synthesis and Analysis Module

Control synthesis is performed using the Optimal Regulator Algorithms for the Control of Linear Systems (ORACLS)¹⁹ library of linear algebra subroutines. ORACLS is a collection of software for linear-quadratic-gaussian (LQG) control law design. Typical linear algebra problems such as matrix operations, solution of Liapunov and Riccati equations, eigenvalue/vector computations and calculation of the matrix exponential can be performed using ORACLS.

The primary driver in choosing the ORACLS routines was the availability of source code. Other control synthesis packages such as MATRIX²⁰ could be used to perform similar matrix algebra operations, however, the ease with which data transfer can be achieved with the in-core architecture to and from ORACLS make it ideally suited for use in the CS^2 testbed. One of the goals in developing the CS^2 testbed is to provide a conducive environment for both structural and control engineers to work and thereby promote interdisciplinary research. The combination of ADS, LASS and ORACLS seems to meet this goal.

Illustrative Example: Geostationary Platform

The solution procedures and design methodology presented in Refs. 3-5 and 21 have been implemented in the software testbed described in Section 3. In this Section, the design of a geostationary platform, is presented to demonstrate the versatility of the proposed software architecture. All of the numerical results were produced using the CS^2 code. Calculations have been performed on the Alliant/FX MIMD computer, the SUN 3/60 workstation and the μ -Vax workstation. On each computer, calculations have been performed in double precision using the Fortran language.

A realistic example of spacecraft being studied for active vibration control is the Earth Pointing Satellite (EPS), shown in Fig. 5, which is a derivative of a geostationary platform proposed for the study of Earth Observation Sciences²². Two flexible antennas are attached to a truss bus. Typical missions involve pointing one antenna to earth, while tracking or scanning with the other antenna. The EPS structure possesses the common feature of a number of proposed spacecraft, namely, several quite flexible appendages are attached to a relatively rigid bus

structure. This type of structure is difficult to control since the attitude control system is usually located on the bus. The attitude control system location may render many of the flexible appendage modes uncontrollable. The following section describes an integrated structure and control design for the EPS system which includes additional actuators for flexible body vibration suppression.

EPS Model Description

There are six basic components that make up the EPS structure: the truss, a 15 m antenna, a 7.5 m antenna, the 15 m antenna to truss attachment, the 7.5 m antenna to truss attachment and the center of gravity attitude control system support structure. The truss has a three meter square cross section. It consists of 51 mm diameter by 1.59 mm wall thickness graphite epoxy tubes. The modulus of elasticity and mass density of the tubes are 275.9×10^9 N/M² and 3250.0 Kg/M³, respectively. A total of 135 beam elements form the truss.

The 15 m antenna is a simple design which exhibits dynamic structural deformations which are characteristic of large space antennas. The antenna has 12 ribs which radiate from the center. The ends of the ribs are connected by hoop elements as shown in Fig. 5. Graphite tube diameters and thicknesses are the same as those used for the truss for the nominal design. However, nonstructural mass which simulates the antenna reflector is included. The non-structural mass is assumed to be twice the weight of the antenna members. The 7.5 m antenna has a geometrically similar design to the 15 m design. Again, for the nominal design, the tube diameters are the same as used on the truss. Nonstructural mass is also assumed, for the 7.5 m antenna, to be twice the weight of the antenna members.

The antenna to truss connections, for both antennas, are made by four beam elements with the same nominal material and tube properties as the truss members. The attitude control system is supported at the center of gravity (CG) of the nominal design by eight beam elements. The attitude control system support is also constructed from the same tubes as used in the truss.

Three torque actuators are located at the CG each with an assumed mass of 50 Kg. This yields a total vehicle mass of 1027.95 Kg. The mass is divided into 548.32 Kg of structural mass, 150 Kg of attitude control mass, and

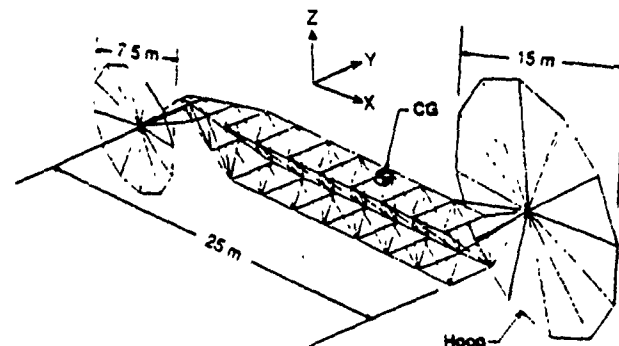


Figure 5. Earth Pointing Satellite Structure

329.63 Kg of nonstructural mass on the antennas. The finite element model representing the EPS consists of 570 degrees of freedom.

Table 1 lists the first 20 frequencies for the nominal EPS structure. Torsion and/or bending of both the 15 m and 7.5 m antennas dominate the flexible body motion in modes 7 through 20. For example, the first flexible mode, mode 7, is torsion of the 15 m antenna. Modes with repeated frequencies (e.g. mode pairs 11-12, 15-16, and 17-18) involve bending of the antennas and occur in pairs due to the symmetry of the antennas.

Table 1. EPS Vibration Frequencies (Hz.)

Mode No.	Frequency
(1 - 6)	0.000
(7)	0.242
(8)	0.406
(9)	0.565
(10)	0.656
(11,12)	0.888
(13)	1.438
(14)	1.536
(15,16)	1.776
(17,18)	3.026
(19)	3.513
(20)	3.531

EPS Integrated Design Formulation

A small disturbance force was applied to the nominal EPS system in the form of a reboost maneuver. The force acted at the center of gravity in the Y-axis direction for 0.1 seconds at a 10 N force level and from 0.1 to 0.2 seconds the force level was -10 N. The disturbance was removed after 0.2 seconds. Figure 6 shows the angular response about the X-axis of the 15 m antenna. A small amount of passive damping was assumed ($D = 0.0002$ K). The vibrational response produced more than 4.5μ radians of RMS pointing error due to this small reboost disturbance. Although many modes participate in the flexible body response, this particular reboost maneuver strongly excites modes near 4 Hz. The following paragraphs present an in-

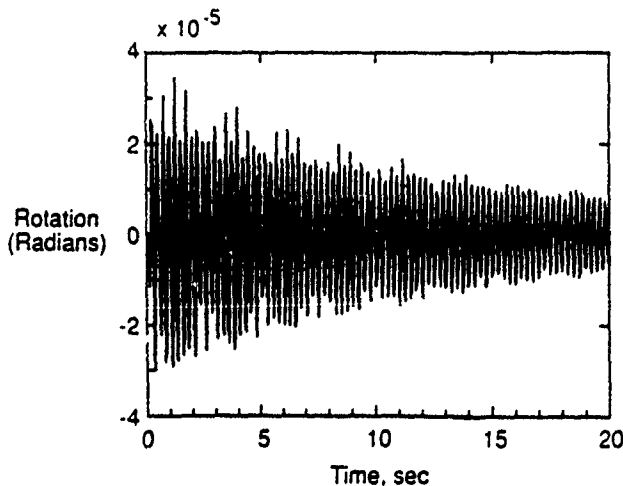


Figure 6. EPS Antenna Pointing: Open-Loop Response

tegrated control and structure design which seeks to lower the vibrational response of the EPS subject to some additional constraints.

In the optimization based design, 20 design variables have been used. Eight design variables are used for the structure, 6 design variables for angular rate control at the CG and 6 design variables for additional flexible body actuators. The structural design variables consisted of the outside tube diameters with the inside diameter set equal to 75 percent of the outer diameter. A lower bound of 0.025 m was set for the minimum allowable value for the structure design variables. The truss was divided into four nearly equal sections, the first section being nearest the 15 m antenna and the fourth section nearest the 7.5 m antenna. The structural design variables d_i are

$$\begin{cases} d_1 = \text{diameter of truss members, section 1} \\ d_2 = \text{diameter of truss members, section 2} \\ d_3 = \text{diameter of truss members, section 3} \\ d_4 = \text{diameter of truss members, section 4} \\ d_5 = \text{diameter of 15 M antenna members} \\ d_6 = \text{diameter of 7.5 M antenna members} \\ d_7 = \text{diameter of antenna to truss connections} \\ d_8 = \text{diameter of members between the CG and truss} \end{cases}$$

The control design variables directly specify the feedback gain matrices. The attitude controller located at the center of gravity of the nominal design was used for damping augmentation with a control law of the form

$$u_{cg} = -F_{cg} \dot{q}_{cg} \quad (8)$$

where u_{cg} is the control torque vector at the CG location

$$q_{cg} = \begin{Bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{Bmatrix}_{cg}$$

and F_{cg} is a (3X3) positive definite matrix. A Cholesky form of the matrices is employed, such that

$$F_{cg} = \bar{F}_{cg} \bar{F}_{cg}^T \quad (9)$$

where

$$\bar{F}_{cg} = \begin{bmatrix} d_9 & 0 & 0 \\ d_{10} & d_{12} & 0 \\ d_{11} & d_{13} & d_{14} \end{bmatrix}$$

To design a flexible body control system for vibration suppression on the EPS system, actuators were added to both the 15 m and the 7.5 m antenna hoops. On each hoop, groups of 3 torque actuators were located at 90 degree increments around the hoop. Thus, a total of 24 actuators were added. These actuators were assumed to have collocated feedback of the angular rate. The control law at point p on the 15 M antenna was

$$u_p^{15} = -F^{15} \dot{q}_p^{15} \quad (10)$$

where

$$F^{15} = \begin{bmatrix} d_{15} & 0 & 0 \\ 0 & d_{16} & 0 \\ 0 & 0 & d_{17} \end{bmatrix}$$

Similarly, the control law at point p on the 7.5 M antenna was

$$u_p^{7.5} = -F^{7.5} \dot{q}_p^{7.5} \quad (11)$$

where

$$\mathbf{F}^{7,8} = \begin{bmatrix} d_{18} & 0 & 0 \\ 0 & d_{19} & 0 \\ 0 & 0 & d_{20} \end{bmatrix}$$

The objective (J) for the integrated design was to minimize the RMS control force required over a 10 second time period

$$J = \frac{1}{10} \int_0^{10} [u_{e1}^T u_{e1} + \sum_{p=1}^4 (u_p^{16} T u_p^{16} + u_p^{7,8} T u_p^{7,8})]^{1/2} d\tau \quad (12)$$

Two constraints were imposed. The first constraint (g_1) was that the total mass of the structure and actuators was to be less than 1000 kg. The second constraint (g_2) was that the RMS pointing error, as measured by the equation below, was to be less than 0.5μ radians.

$$\begin{cases} g_1 = (\text{total mass}/1000) - 1 \leq 0 \\ g_2 = (RMS_e/0.5E-6) - 1 \leq 0 \end{cases} \quad (13)$$

where

$$RMS_e = \frac{1}{5} \int_0^{10} [q_1^{16} T q_1^{16} + q_1^{7,8} T q_1^{7,8}]^{1/2} d\tau$$

The remaining undefined quantity needed to perform the integrated structure and controller design is the mass of the actuators. A linear relationship between the feedback gain matrices and the actuator mass has been employed. First, the infinity norm of the gain matrices, $\mathbf{F}_{e1}$, $\mathbf{F}^{16}$, $\mathbf{F}^{7,8}$, are multiplied by a worst case attitude rate ($\dot{\theta}_{max}$). Subsequently, the actuator mass is based on an empirical relation between actuator torque and mass per unit torque. The actuator mass equation assumed in this study is summarized as follows:

$$m_a = \gamma(3|\mathbf{F}_{e1}|_{\infty} + 12|\mathbf{F}^{16}|_{\infty} + 12|\mathbf{F}^{7,8}|_{\infty})\dot{\theta}_{max} \quad (14)$$

where the mass per unit torque = $\gamma = 1 \text{ kg}/(\text{N}\cdot\text{m})$ and $\dot{\theta}_{max} = 1.7453 \text{ rad/s}$.

EPS Integrated Design Results

The flow chart shown in Fig. 7 shows the integrated design path using CS³. The dominant computational burden for this optimization problem is associated with transient response analysis of the closed-loop equations to compute the RMS control force and the RMS pointing error. Since gradient information was computed using finite difference equations, 20 function evaluations were required for each iteration of the optimizer. The EPS integrated design demonstrates the need for efficient transient response simulation of CSI systems.

Figure 8 shows the closed-loop angular response about the X-axis of the 15 m antenna after design optimization. The pointing error is significantly reduced from that of the open-loop system shown. Table 2 shows the nominal design (open-loop), the initial design used to begin the optimization, and the final optimization results. This design required 20 iterations of the optimizer. The RMS control torque was reduced from an initial value of 14.3 N-m to 2.68 N-m.

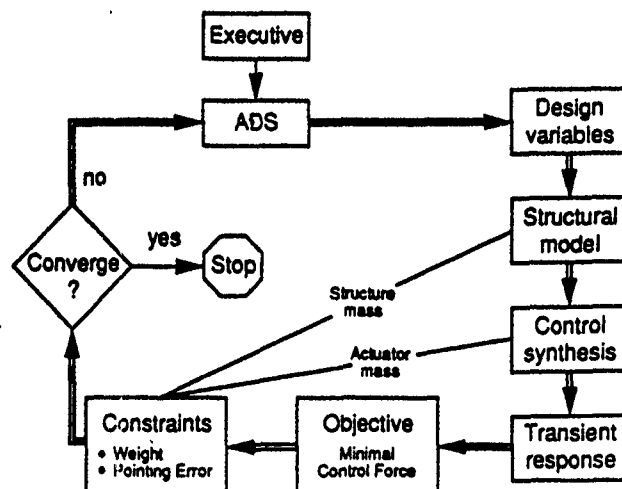


Figure 7. EPS Design Using Controlled Structure Simulation Software

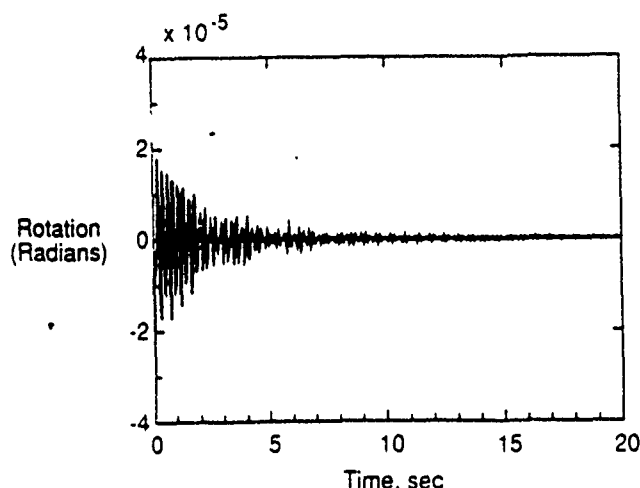


Figure 8. EPS Antenna Pointing: Closed-Loop Response

Design Variable	Nominal	Initial	Final
Member Diameters			
1	0.051	0.051	0.031
2	0.051	0.051	0.045
3	0.051	0.051	0.054
4	0.051	0.051	0.087
5	0.051	0.051	0.056
6	0.051	0.051	0.049
7	0.051	0.051	0.050
8	0.051	0.051	0.025
Attitude Control Gains			
9	0.0	31.6	2.96
10	0.0	31.6	0.204
11	0.0	31.6	0.661
12	0.0	31.6	5.12
13	0.0	31.6	6.61
14	0.0	31.6	6.29
15 m Antenna Gains			
15	0.0	70.	5.73
16	0.0	70.	0.86
17	0.0	70.	4.66
7.5 m Antenna Gains			
18	0.0	70.	9.38
19	0.0	70.	23.
20	0.0	70.	6.71

Examination of the closed-loop response in Fig. 8 indicates that modes near 4 Hz. still dominate the vibrational behavior. However, the optimized design reduced the initial vibration amplitude by a factor of two as compared to the open-loop response. The optimized structural design reduced the "transfer function" amplitude between the CG location and the antenna measurement locations. Reduction of the "transfer function" amplitude was accomplished by changing the mode shapes of the structural system. An alternative design approach is presented next.

EPS Integrated Design Results with Initially Tailored Structure

One difficulty with optimization based designs is the selection of an initial structural design to start the optimizer. If the initial design is too far from the optimum, then the optimizer may require a large number of iterations before convergence, or the optimizer may converge to a local minima far from the optimal solution. Thus, design methodology is sought to choose good initial designs to start the optimizer.

The structural tailoring strategy presented in Ref. 21 was developed to obtain good structural designs for CSI systems using a particular form of the control law. The control law cost function involved minimizing a quadratic measure of controller energy and vibrational response. The structural tailoring strategy to minimize the cost function involved minimizing a quantity inversely related to the open-loop vibration frequencies of the structure.

The EPS system was redesigned using structural tailoring to select an initial set of structure design variables. Table 3 gives the vibration frequencies of the tailored EPS design. The first flexible mode of the tailored design still involves torsion of the 15 m antenna, however, the frequency has been increased by 300 percent. This large frequency increase occurred by stiffening the antenna support truss represented by design variable number 7. The 15 m antenna bending modes also increased in frequency for the tailored design.

Table 3. Nominal and Tailored EPS
Vibration Frequencies (Hz.)

Mode No.	Nominal	Tailored
(1 - 6)	0.000	0.000
(7)	0.242	0.726
(8)	0.406	0.800
(9)	0.565	1.197
(10)	0.656	1.229
(11)	0.888	1.229
(12)	0.888	1.277

Table 4 shows the initial tailored design used to start the optimizer. The nominal (open-loop) design and the final optimization results are also given in Table 4. It is noted the final structural design variables changed little from the initial tailored design. The closed-loop response of the EPS system for the design of Table 4 is quite similar in magnitude to the response given in Fig. 8.

Table 4 EPS Optimisation Results
with Structural Tailoring

Design Variable	Nominal	Tailored	Final
<i>Member Diameters</i>			
1	0.051	0.025	0.025
2	0.051	0.025	0.027
3	0.051	0.025	0.028
4	0.051	0.025	0.025
5	0.051	0.070	0.069
6	0.051	0.031	0.028
7	0.051	0.216	0.211
8	0.051	0.025	0.025
<i>Attitude Control Gains</i>			
9	0.0	31.6	28.43
10	0.0	31.6	29.44
11	0.0	31.6	29.50
12	0.0	31.6	28.68
13	0.0	31.6	26.28
14	0.0	31.6	8.21
<i>15 m Antenna Gains</i>			
15	0.0	70.	187.
16	0.0	70.	0.0
17	0.0	70.	39.0
<i>7.5 m Antenna Gains</i>			
18	0.0	70.	98.7
19	0.0	70.	191.
20	0.0	70.	48.3

By applying the structural tailoring methodology of Ref. 21 to compute an initial design for the optimizer a 70 percent reduction in computational cost was realized. This is shown graphically in Fig. 9 by the number of iterations the optimization required. When starting with the nominal structure 20 iterations were needed for convergence, whereas by starting with the tailored design, only 6 iterations were needed. Note that this resulted in 280 fewer objective function and constraint calculations because gradient information was computed using finite differences. These results indicate that design methodology which select good initial designs for the optimizer in conjunction with the modular closed-loop simulation procedures adopted for the present study yield computationally efficient tools for interdisciplinary design of CSI systems.

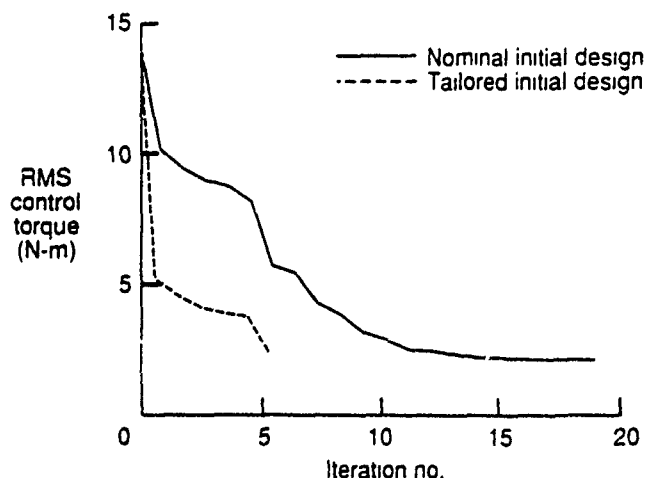


Figure 9. EPS Control Force Optimization History

Concluding Remarks

A software testbed has been implemented as a stand-alone analysis and optimization program for the preliminary design of control-structure interaction systems. By employing a partitioned control-structure interaction analysis procedure, versatility and modularity of the software has been maintained. Implementation of the testbed has been performed using an in-core data base approach which permits the designer great freedom in developing user interface programs. The shared data architecture encourages interdisciplinary design by both control and structural experts.

Numerical results for a realistic geostationary platform have been presented which demonstrate some of the capabilities of the testbed for integrated control and structure design. It has been shown that the integrated design of controlled structure systems differs substantially in implementation than the individual analysis of the single-disciplines. The user interfaces in the software testbed should permit development and verification of CSI design methodology, improved algorithms and more physical insight into unified design.

The CSI design testbed presented herein addresses the linear time invariant class of controlled structures. Future activities must involve multi-body dynamics and time variant controller design. Nevertheless, the present architecture should be invoked where possible to facilitate integrated structure and controller design. In addition, the present work must be extended to permit more realistic simulation of experimental testbeds by including actuator and sensor dynamics, filters, discrete time sampling and real time computations. The extension of CS² for simulation of experimental testbeds is currently underway.

Acknowledgements

The contribution of the second author to the present work reported herein was supported by Air Force Office of Scientific Research under Grant F49620-87-C-0074. We thank Dr. Anthony K. Amos for his encouragement during the course of this study.

References

- Cooper, P. A., Young, J. W. and Sutter, T. R., "Multidisciplinary Analysis of Actively Controlled Large Flexible Spacecraft," First NASA/DOD CSI Technology Conference, NASA CP-2447, 1986.
- Baker, M. et. al. "Space Station Multidisciplinary Analysis Capability - IDEAS²," Proceedings of the 27th Structures, Structural Dynamics and Materials Conference, AIAA 86-0954, San Antonio, TX, May 19-21, 1986.
- Belvin, W. K. and Park, K. C., "On the State Estimation of Structures with Second-Order Observers," Proceedings of the 30th Structures, Structural Dynamics and Materials Conference, AIAA Paper No. 89-1241, April 3-5, 1989.
- Park, K. C. and Belvin, W. K., "Stability and Implementation of Partitioned CSI Solution Procedures," Proceedings of the 30th Structures, Structural Dynamics and Materials Conference, AIAA Paper No. 89-1238, April 3-5, 1989.
- Belvin, W. K., *Simulation and Interdisciplinary Design Methodology for Control-Structure Interaction Systems*, PhD Dissertation, Department of Aerospace Engineering Sciences and Center for Space Structures and Controls, University of Colorado, Report No. CU-CSSC-89-10, July 1989, Boulder, Colorado.
- Vanderplaats, G. N., "ADS- A Fortran Program For Automated Design Synthesis - Version 1.10", NASA CR 177985, Sept. 1985.
- Horta, L. G., Juang, J. N. and Junkins, J. L., "A Sequential Linear Optimization Approach for Controller Design," AIAA Paper 85-1971, 1985.
- Lim, K., "A Unified Approach To Structure And Controller Design Optimization", Ph.D. Dissertation, Virginia Polytechnic Institute and State University, August 1986.
- Kelley, J. E., "The Cutting Plane Method for Solving Convex Programs," *Journal of SIAM*, 1960, pp. 703-712.
- Moses, F., "Optimum Structural Design Using Linear Programming", *Proceedings of the ASCE*, Vol. 90, ST6, 1964, pp. 89-104.
- Greene, W. H. and Haftka, R. T., "Computational Aspects of Sensitivity Calculations in Transient Structural Analysis," NASA TM 100589, April 1988.
- Haftka, R. T. and Kamat, M. P., *Elements of Structural Optimization*, Martinus Nijhoff Publishers, 1985.
- Fox, R. L. and Kapoor, M. P., "Structural Optimization in the Dynamics Regime: A Computational Approach," *AIAA Journal*, Vol. 8, No. 10, Oct. 1970, pp. 1798-1804.
- Mills-Curran, W. C. and Schmit, L. A., "Structural Optimization with Dynamic Behavior Constraints," *AIAA Journal*, Vol. 23, No. 1, Jan. 1985, pp. 132-138.
- Haug, E. J. and Arora, J. S., "Design Sensitivity Analysis of Elastic Mechanical Systems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 15, 1978, pp. 35-62.
- Haug, E. J., "Design Sensitivity Analysis of Dynamic Systems," NATO ASI Series, Vol. F27, *Computer Aided Optimal Design: Structural and Mechanical Systems*, Edited by C. A. Mota Soares, Springer-Verlag, 1987, pp. 705-755.
- NASTRAN Users Manual*, (Level 17.5), NASA SP-222(05), 1978.
- Whetstone, W. D., *EISI-EAL Engineering Analysis Language Reference Manual*, EISI-EAL System Level 2091, Vols. 1 and 2, Engineering Information Systems Inc., San Jose, CA, July, 1983.
- Armstrong, E. S., "ORACLS- A System For Linear-Quadratic-Gaussian Control Law Design," NASA Technical Paper 1106, 1978.
- MATRIX/W: Engineering Analysis, Control Design, System Identification and Signal Processing, Integrated Systems Inc., Santa Clara, CA., 1989.
- Belvin, W. K. and Park, K. C., "Structural Tailoring and Feedback Control Synthesis: An Interdisciplinary Approach," Proceedings of the 29th Structures, Structural Dynamics and Materials Conference, AIAA Paper No. 88-2206, May, 1988, also to appear in the *Journal of Guidance, Control and Dynamics*, 1990.
- Ramler, J. and Durrett, R., "NASA'S Geostationary Communications Platform Program," Proceedings of the AIAA 10th Communications Satellite Systems Conference, AIAA 84-0702, Orlando, FL, March 4, 1984.

**Second-Order Discrete Kalman Filtering Equations
for
Control-Structure Interaction Simulations⁺**

K. C. Park¹ and K. F. Alvin²

Center for Space Structures and Controls
University of Colorado, Campus Box 429
Boulder, Colorado 80309

and

W. Keith Belvin³

Spacecraft Dynamics Branch
NASA Langley Research Center
Hampton, Virginia 23665

Abstract

A general form for the first-order representation of the continuous, second-order linear structural dynamics equations is introduced in order to derive a corresponding form of first-order continuous Kalman filtering equations. Time integration of the resulting first-order Kalman filtering equations is carried out via a set of linear multistep integration formulas. It is shown that a judicious combined selection of computational paths and the undetermined matrices introduced in the general form of the first-order linear structural systems leads to a class of second-order discrete Kalman filtering equations involving only symmetric, sparse $N \times N$ solution matrices. The present integration procedure thus overcomes the difficulty in resolving the difference between the time derivative of the estimated displacement vector ($\frac{d}{dt}\hat{x}$) and the estimated velocity vector ($\dot{\hat{x}}$) that are encountered when one attempts first to eliminate ($\dot{\hat{x}}$) in order to form an equivalent set of second-order filtering equations in terms of ($\frac{d}{dt}\hat{x}$). A partitioned solution procedure is then employed to exploit matrix symmetry and sparsity of the original second-order structural systems, thus realizing substantial computational simplicity heretofore thought difficult to achieve.

⁺ An earlier version of the present paper without numerical experiments was presented at the AIAA Guidance and Control Conference, Portland, Ore., 20-22 August 1990, Paper No. AIAA 90-3387.

¹ Professor of Aerospace Engineering, University of Colorado. Associate Fellow of AIAA.

² Graduate Research Assistant ³ Structural Dynamics Division, NASA Langley Research Center. Member AIAA.

Introduction

Current practice in the design, modeling and analysis of flexible large space structures is by and large based on the finite element method and the associated software. The resulting discrete equations of motion for structures, both in terms of physical coordinates and of modal coordinates, are expressed in a second-order form. As a result, the structural engineering community has been investing a considerable amount of research and development resources to develop computer-oriented discrete modeling tools, analysis methods and interface capabilities with design synthesis procedures; all of these exploiting the characteristics of second-order models.

On the other hand, modern linear control theory has its roots firmly in a first-order form of the governing differential equations, e.g., (Kwakernaak and Sivan, 1972). Thus, several investigators have addressed the issues of interfacing second-order structural systems and control theory based on the first-order form (Hughes and Skelton, 1980; Arnold and Laub, 1984; Bender and Laub, 1985; Oshman, Inman and Laub, 1987; Belvin and Park, 1989, 1990). As a result of these studies, it has become straightforward for one to synthesize non-observer based control laws within the framework of a first-order control theory and then to recast the resulting control laws in terms of the second-order structural systems.

Unfortunately, controllers based on a first-order observer are difficult to express in a pure second-order form because the first-order observer implicitly incorporates an additional filter equation (Belvin and Park, 1989). However a recent work (Juang and Maghami, 1990) has enabled the first-order observer gain matrices to be synthesized using only second-order equations. To complement the second-order gain synthesis, the objective of the present paper is to develop a second-order based simulation procedure for first-order observers. The particular class of first-order observers chosen for study are the Kalman Filter based state estimators as applied to second-order structural systems. The procedure permits simulation of first-order observers with nearly the same solution procedure used for treating the structural dynamics equation. Hence, the reduced size of system matrices and the computational techniques that are tailored to sparse second-order structural systems may be employed. As will be shown, the procedure hinges on discrete time integration formulas to effectively reduce the continuous time Kalman Filter to a set of second-order difference equations.

The paper first reviews of the conventional first-order representation of the continuous second-order structural equations of motion. An examination of the corresponding first-order Kalman filtering equations indicates that, due to the difference in the derivative of the estimated displacement ($\frac{d}{dt}\hat{x}$) and the estimated velocity ($\hat{\dot{x}}$), transformation of the first-order observer into an equivalent second-order observer requires the time derivative of measurement data, a process not recommended for practical implementation.

Next, a transformation via a generalized momentum is introduced to recast the structural equations of motion in a general first-order setting. It is shown that discrete time numerical integration followed by reduction of the resulting difference equations circumvents the need for the time derivative of measurements to solve Kalman filtering equations in a second-order framework. Hence, the Kalman filter equations can be solved using a second-order solution software package.

Subsequently, computer implementation aspects of the present second-order observer are presented. Several computational paths are discussed in the context of discrete and continuous time simulation. For continuous time simulation, an equation augmentation is introduced to exploit the symmetry and sparsity of the attendant matrices by maintaining state dependant control and observer terms on the right-hand-side (RHS) of the filter equations. In addition, the computational efficiency of the present second order observer as compared to the first order observer is presented.

Continuous Formulation of Observers for Structural Systems

Linear, second-order discrete structural models can be expressed as

$$M\ddot{x} + D\dot{x} + Kx = Bu + Gw, \quad x(0) = x_0, \quad \dot{x}(0) = \dot{x}_0 \quad (1)$$

$$u = -Z_1x - Z_2\dot{x}$$

with the associated measurements

$$z = H_1x + H_2\dot{x} + \nu \quad (2)$$

where M, D, K are the mass damping and stiffness matrices of size $(N \times N)$; x is the structural displacement vector, $(N \times 1)$; u is the active control force $(m \times 1)$; B is a constant force distribution matrix $(N \times m)$; z is a set of measurements $(r \times 1)$; H_1 and H_2 are the measurement distribution matrices $(r \times N)$; Z_1 and Z_2 are the control feedback gain matrices $(m \times N)$; w and ν are zero-mean, white Gaussian processes with their respective covariances Q and R ; and the superscript dot designates time differentiation. In the present study, we will restrict ourselves to the case wherein Q and R are uncorrelated with each other and the initial conditions x_0 and $\dot{x}_0$ are also themselves jointly Gaussian with known means and covariances.

The conventional representation of (1) in a first-order form is facilitated by

$$\begin{cases} x_1 = x \\ x_2 = \dot{x} = \dot{x}_1 \\ M\dot{x}_2 = M\ddot{x} = Bu + Gw - Dx_2 - Kx_1 \end{cases} \quad (3)$$

which, when cast in a first-order form, can be expressed as

$$\begin{cases} E\dot{q} = Fq + \bar{B}u + \bar{G}w, & q = \begin{pmatrix} x_1 & x_2 \end{pmatrix}^T \\ z = Hq + v \end{cases} \quad (4)$$

where

$$E = \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix}, \quad F = \begin{bmatrix} 0 & I \\ -K & -D \end{bmatrix},$$

$$\bar{B} = \begin{Bmatrix} 0 \\ B \end{Bmatrix}, \quad \bar{G} = \begin{Bmatrix} 0 \\ G \end{Bmatrix} \quad (5)$$

It is well-known that the Kalman filtering equations (Kalman, 1961; Kalman and Bucy, 1963) for (4) can be shown to be (Arnold and Laub, 1984):

$$E\dot{\hat{q}} = F\hat{q} + \bar{B}u + EPH^T R^{-1} \bar{z} \quad (6)$$

where

$$\bar{z} = z - H\hat{q}, \quad P = \begin{bmatrix} U & S^T \\ S & L \end{bmatrix}, \quad \hat{q} = \begin{Bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{Bmatrix} = \begin{Bmatrix} \hat{x} \\ \dot{\hat{x}} \end{Bmatrix} \quad (7)$$

in which U and L are positive definite matrices and the matrix P is determined by the Riccati equation (Kwakernaak and Sivan, 1972; Arnold and Laub, 1984)

$$E\dot{P}E^T = FPE^T + EPF^T - EPH^T R^{-1} HPE^T + \bar{G}\bar{Q}\bar{G}^T \quad (8)$$

The inherent difficulty of reducing the first-order Kalman filtering equations given by (6) to second order form can be appreciated if one attempts to write (6) in a form introduced in (3):

$$\begin{cases} a) & \hat{x}_1 = \hat{x} \\ b) & \hat{x}_2 = \dot{\hat{x}} = \dot{\hat{x}}_1 - L_1 \bar{z} \\ c) & M\dot{\hat{x}}_2 = -D\hat{x}_2 - K\hat{x}_1 + B\hat{u} + ML_2 \bar{z} \end{cases} \quad (9)$$

where

$$L_1 = (H_1 U + H_2 S)^T R^{-1}, \quad L_2 = (H_1 S^T + H_2 L)^T R^{-1}$$

Note from (9b) that $\hat{x}_2 \neq \dot{\hat{x}}_1$. In other words, the time derivative of the estimated displacement ($\dot{\hat{x}}$) is not the same as the estimated velocity ($\hat{x}_2$); hence, $\hat{x}_1$ and $\hat{x}_2$ must be treated as two independent variables, an important observation somehow overlooked in Hashemipour and Laub (1988).

Of course, although not practical, one can eliminate $\hat{x}_2$ from (9). Assuming $\dot{\hat{x}}_1$ and $\hat{x}_2$ are differentiable, differentiate (9b) and multiply both sides by M to obtain

$$M\ddot{\hat{x}}_1 = M\dot{\hat{x}}_2 + ML_1 \dot{\bar{z}} \quad (10)$$

Substituting $M\dot{\hat{x}}_2$ from (9c) and $\hat{x}_2$ from (9b) in (10) yields

$$M\ddot{\hat{x}}_1 = -D(\dot{\hat{x}}_1 - L_1\bar{z}) - K\hat{x}_1 + Bu + ML_2\bar{z} + ML_1\dot{\bar{z}} \quad (11)$$

which, upon rearrangements, becomes

$$M\ddot{\hat{x}}_1 + D\dot{\hat{x}}_1 + K\hat{x}_1 = Bu + ML_2\bar{z} + ML_1\dot{\bar{z}} + DL_1\bar{z} \quad (12)$$

There are two difficulties with the above second-order observer. First, the numerical solution of (12) involves the computation of $\ddot{\hat{x}}_1$ when rate measurements are made. The accuracy of this computation is in general very susceptible to errors caused in numerical differentiation of $\dot{\hat{x}}_1$. Second, and most important, the numerical evaluation of $\dot{\bar{z}}$ that is required in (12) assumes that the derivative of measurement information is available which should be avoided in practice. We now present a computational procedure that circumvents the need for computing measurement derivatives and that enables one to construct observers based on the second-order models.

Second-Order Transformation of Continuous Kalman Filtering Equations

This section presents a transformation of the continuous time first-order Kalman filter to a discrete time set of second-order difference equations for digital implementation. The procedure avoids the need for measurement derivative information. In addition, the sparsity and symmetry of the original mass, damping and stiffness matrices can be maintained. Prior to describing the numerical integration procedure, a transformation based on generalized momenta is presented which is later used to improve computational efficiency of the equation solution.

Generalized Momenta

Instead of the conventional transformation (3) of the second-order structural system (1) into a first-order form, let us consider the following generalized momenta (Jensen, 1974; Felippa and Park, 1978):

$$\begin{cases} a) & x_1 = x \\ b) & x_2 = AM\dot{x}_1 + Cx_1 \end{cases} \quad (13)$$

where A and C are constant matrices to be determined. Time differentiation of (13b) yields

$$\dot{x}_2 = AM\ddot{x}_1 + C\dot{x}_1 \quad (14)$$

Substituting (1) via (13a) into (14), one obtains

$$\dot{x}_2 = A(Bu + Gw) - (AD - C)\dot{x}_1 - AKx_1 \quad (15)$$

Finally, pairing of (13b) and (15) gives the following first-order form:

$$\begin{bmatrix} AM & 0 \\ AD - C & I \end{bmatrix} \begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} + \begin{bmatrix} C & -I \\ AK & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{bmatrix} 0 \\ A(Bu + Gw) \end{bmatrix} \quad (16)$$

The associated Kalman filtering equation can be shown to be of the following form:

$$\begin{bmatrix} AM & 0 \\ AD - C & I \end{bmatrix} \begin{Bmatrix} \dot{\hat{x}}_1 \\ \dot{\hat{x}}_2 \end{Bmatrix} + \begin{bmatrix} C & -I \\ AK & 0 \end{bmatrix} \begin{Bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ ABu \end{Bmatrix} + \begin{bmatrix} AM & 0 \\ AD - C & I \end{bmatrix} \begin{bmatrix} \bar{L}_1 \\ \bar{L}_2 \end{bmatrix} \bar{z} \quad (17)$$

where

$$\bar{L}_1 = (\bar{H}_1 U + \bar{H}_2 S)^T R^{-1}, \quad \bar{L}_2 = (\bar{H}_1 S^T + \bar{H}_2 L)^T R^{-1}$$

and $\bar{H}_1$ and $\bar{H}_2$ correspond to a modified form of measurements expressed as

$$z = H_1 x + H_2 \dot{x} = \bar{H}_1 x_1 + \bar{H}_2 x_2 \quad (18)$$

where

$$\bar{H}_1 = H_1 - H_2 M^{-1} A^{-1} C, \quad \bar{H}_2 = H_2 M^{-1} A^{-1}$$

Clearly, as in the conventional first-order form (9), $\hat{x}_1$ and $\hat{x}_2$ in (17) are now two independent variables. Specifically, the case of $A = M^{-1}$ and $C = 0$ corresponds to (3) with $x_2 = \dot{x}_1$. However, as we shall see below, the Kalman filtering equations based on the generalized momenta (13) offer several computational advantages over (3).

Numerical Integration

At this juncture it is noted that in the previous section one first performs the elimination of $\hat{x}_1$ in order to obtain a second-order observer, then performs the numerical solution of the resulting second-order observer. This approach has the disadvantage of having to deal with the time derivative of measurement data. To avoid this, we will first integrate numerically the associated Kalman filtering equation (17).

The direct time integration formula we propose to employ is a mid-point version of the trapezoidal rule:

$$\begin{cases} a) & \begin{Bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{Bmatrix}^{n+1/2} = \begin{Bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{Bmatrix}^n + \delta \begin{Bmatrix} \dot{\hat{x}}_1 \\ \dot{\hat{x}}_2 \end{Bmatrix}^{n+1/2} \\ b) & \begin{Bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{Bmatrix}^{n+1} = 2 \begin{Bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{Bmatrix}^{n+1/2} - \begin{Bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{Bmatrix}^n \end{cases} \quad (19)$$

where the superscript n denotes the discrete time interval $t^n = nh$, h is the time increment and $\delta = h/2$.

Time discretization of (17) by (19a) at the $n + 1/2$ time step yields

$$\begin{aligned} & \begin{bmatrix} AM & 0 \\ AD - C & I \end{bmatrix} \begin{Bmatrix} \hat{x}_1^{n+1/2} - \hat{x}_1^n \\ \hat{x}_2^{n+1/2} - \hat{x}_2^n \end{Bmatrix} + \delta \begin{bmatrix} C & -I \\ AK & 0 \end{bmatrix} \begin{Bmatrix} \hat{x}_1^{n+1/2} \\ \hat{x}_2^{n+1/2} \end{Bmatrix} \\ &= \delta \begin{bmatrix} AM & 0 \\ AD - C & I \end{bmatrix} \begin{bmatrix} \bar{L}_1 \\ \bar{L}_2 \end{bmatrix} \bar{z}^{n+1/2} + \delta \begin{Bmatrix} 0 \\ ABu^{n+1/2} \end{Bmatrix} \end{aligned} \quad (20)$$

The above difference equations require the solution of matrix equations of $2N$ variables, namely, in terms of the two variables $\hat{x}_2^{n+1/2}$ and $\hat{x}_1^{n+1/2}$, each with a size of N . To reduce the above coupled equations of order $2N$ into the corresponding ones of order N , we proceed in the following way by exploiting the nature of parametric matrices of A and C as introduced in (13). To this end, we write out (20) as two coupled difference equations as follows:

$$\begin{aligned} & AM(\hat{x}_1^{n+1/2} - \hat{x}_1^n) + \delta(C\hat{x}_1^{n+1/2} - \hat{x}_2^{n+1/2}) \\ &= \delta AM\bar{L}_1 \bar{z}^{n+1/2} \end{aligned} \quad (21)$$

$$\begin{aligned} & (AD - C)(\hat{x}_1^{n+1/2} - \hat{x}_1^n) + (\hat{x}_2^{n+1/2} - \hat{x}_2^n) + \delta AK\hat{x}_1^{n+1/2} \\ &= \delta(AD - C)\bar{L}_1 \bar{z}^{n+1/2} + \delta\bar{L}_2 \bar{z}^{n+1/2} + \delta ABu^{n+1/2} \end{aligned} \quad (22)$$

Multiplying (22) by δ and adding the resulting equation to (21) yields

$$\begin{aligned} & A(M + \delta D + \delta^2 K)\hat{x}_1^{n+1/2} = (AM + \delta(AD - C))\hat{x}_1^n + \delta\hat{x}_2^n \\ & + \{\delta AM\bar{L}_1 + \delta^2(AD - C)\bar{L}_1 + \delta^2\bar{L}_2\}\bar{z}^{n+1/2} + \delta^2 ABu^{n+1/2} \end{aligned} \quad (23)$$

Of several possible choices for matrices A and B , we will examine

$$\begin{cases} a) & A = I, \quad C = D \\ b) & A = M^{-1}, \quad C = 0 \end{cases} \quad (24)$$

The choice of (24a) reduces (23) to:

$$(M + \delta D + \delta^2 K)\hat{x}_1^{n+1/2} = M\hat{x}_1^n + \delta\hat{x}_2^n + \delta^2 Bu^{n+1/2} + \delta\{M\bar{L}_1 + \delta\bar{L}_2\}\bar{z}^{n+1/2} \quad (25)$$

so that once $\hat{x}_1^{n+1/2}$ is computed, $\hat{x}_2^{n+1/2}$ is obtained from (22) rewritten as

$$\hat{x}_2^{n+1/2} = \hat{x}_2^n + \delta\hat{g}^n - \delta K\hat{x}_1^{n+1/2} \quad (26)$$

where

$$\hat{g}^n = Bu^{n+1/2} + \bar{L}_2\bar{z}^{n+1/2} \quad (27)$$

which is already computed in order to construct the right-hand side of (25). Hence, $K\hat{x}_1^{n+1/2}$ is the only additional computation needed to obtain $\hat{x}_2^{n+1/2}$. It is noted that neither any numerical differentiation nor matrix inversion is required in computing $\hat{x}_2^{n+1/2}$. This has been achieved through the introduction of the general transformation (13) and the particular choice of the parameter matrices given by (24a).

On the other hand, if one chooses the conventional representation (24b), the solution of $\hat{x}_1^{n+1/2}$ is obtained from (23)

$$(M + \delta D + \delta^2 K)\hat{x}_1^{n+1/2} = (M + \delta D)\hat{x}_1^n + \delta M\hat{x}_2^n + \delta\{(M + \delta D)\bar{L}_1 + \delta M\bar{L}_2\}\bar{z}^{n+1/2} + \delta^2 Bu^{n+1/2} \quad (28)$$

Once $\hat{x}_1^{n+1/2}$ is obtained, $\hat{x}_2^{n+1/2}$ can be computed either by

$$\hat{x}_2^{n+1/2} = (\hat{x}_1^{n+1/2} - \hat{x}_1^n)/\delta - \bar{L}_1\bar{z}^{n+1/2} \quad (29)$$

which is not accurate due to the numerical differentiation to obtain $\hat{x}_1^{n+1/2}$, or by (22)

$$\hat{x}_2^{n+1/2} = \hat{x}_2^n + \delta\hat{g}^n - \delta M^{-1}K\hat{x}_1^{n+1/2} - M^{-1}D(\hat{x}_1^{n+1/2} - \hat{x}_1^n) + \delta M^{-1}D\bar{L}_1\bar{z}^{n+1/2} \quad (30)$$

which involves two additional matrix-vector multiplications, when $D \neq 0$, as compared with the choice of $A = I$ and $C = D$. Thus (24a) is the preferred representation in a first-order form of the second-order structural dynamics equations (1) and is used in the remainder of this work.

Decoupling Of Difference Equations

We have seen in the previous section, instead of solving the first-order Kalman filtering equations of $2n$ variables for the structural dynamics systems (1), the solution of the implicit time-discrete observer equation (25) of n variables can potentially offer a substantial computational saving by exploiting the reduced size and sparsity of M, D and K . This assumes that $\bar{z}^{n+1/2}$ and $u^{n+1/2}$ are available, which is not the case since at the n^{th} time step

$$u^{n+1/2} = -\bar{Z}_1 \hat{x}_1^{n+1/2} - \bar{Z}_2 \hat{x}_2^{n+1/2} \quad (31)$$

$$\bar{z}^{n+1/2} = z^{n+1/2} - \bar{H}_1 \hat{x}_1^{n+1/2} - \bar{H}_2 \hat{x}_2^{n+1/2} \quad (32)$$

requires both $\hat{x}_1^{n+1/2}$ and $\hat{x}_2^{n+1/2}$ even if $z^{n+1/2}$ is assumed to be known from measurements or by solution of (1). Note in (32), the control gain matrices are transformed by

$$\bar{Z}_1 = Z_1 - Z_2 M^{-1} A^{-1} C, \quad \bar{Z}_2 = Z_2 M^{-1} A^{-1}$$

There are two distinct approaches to uncouple (25) and (26) as described in the following sections.

Discrete Time Update

Equations (31) and (32) can be approximated using

$$\bar{z}^{n+1/2} \simeq z^n - \bar{H}_1 \hat{x}_1^n - \bar{H}_2 \hat{x}_2^n \quad (33)$$

$$u^{n+1/2} \simeq -\bar{Z}_1 \hat{x}_1^n - \bar{Z}_2 \hat{x}_2^n \quad (34)$$

This approximation leads to a discrete time update of the control force and state correction terms which is analogous to that which exists in experiments where a finite bandwidth of measurement updates occurs. For discrete time approximation, the step size $h = t^{n+1} - t^n$ should be chosen to match the time required to acquire, process and output a control update.

Discrete time simulation is quite simple to implement as the control force and state corrections are treated with no approximation on the right-hand-side (RHS) of (25) and (26). Should continuous time simulation be required, a different approach is necessary.

Continuous Time Update

To simulate the system given in (25) and (26) in continuous time, strictly speaking, one must rearrange (25) and (26) so that the terms involving $\hat{x}_1^{n+1/2}$ and $\hat{x}_2^{n+1/2}$ are augmented

to the left-hand-side (LHS) of the equations. However, this augmentation into the solution matrix $(M + \delta D + \delta^2 K)$ would destroy the computational advantages of the matrix sparsity and symmetry. Thus, a partitioned solution procedure has been developed for continuous time simulation as described in (Park and Belvin, 1991). The procedure, briefly outlined herein, maintains the control force and state correction on the RHS of the equations as follows.

First, $\hat{x}_1^{n+1/2}$ and $\hat{x}_2^{n+1/2}$ are predicted by

$$\hat{x}_{1p}^{n+1/2} = \hat{x}_1^n, \quad \hat{x}_{2p}^{n+1/2} = \hat{x}_2^n \quad (35)$$

However, instead of direct substitution of the above predicted quantity to obtain $u_p^{n+1/2}$ and $\bar{z}_p^{n+1/2}$ based on (31) and (32), equation augmentations are introduced to improve the accuracy of $u_p^{n+1/2}$ and $\bar{z}_p^{n+1/2}$. Of several augmentation procedures that are applicable to construct discrete filters for the computations of $u^{n+1/2}$ and $\bar{z}^{n+1/2}$, we substitute (26) into (31) and (32) to obtain

$$\begin{cases} u^{n+1/2} = -\bar{Z}_1 \hat{x}_1^{n+1/2} - \bar{Z}_2 (\hat{x}_2^n - \delta K \hat{x}_1^{n+1/2} + \\ \delta B u^{n+1/2} + \delta \bar{L}_2 \bar{z}^{n+1/2}) \\ \bar{z}^{n+1/2} = z^{n+1/2} - \bar{H}_1 \hat{x}_1^{n+1/2} - \\ \bar{H}_2 (\hat{x}_2^n - \delta K \hat{x}_1^{n+1/2} + \delta B u^{n+1/2} + \delta \bar{L}_2 \bar{z}^{n+1/2}) \end{cases} \quad (36)$$

Rearranging the above coupled equations, one obtains

$$\begin{bmatrix} (I + \delta \bar{Z}_2 B) & \delta \bar{Z}_2 \bar{L}_2 \\ \delta \bar{H}_2 B & (I + \delta \bar{H}_2 \bar{L}_2) \end{bmatrix} \begin{Bmatrix} u^{n+1/2} \\ \bar{z}^{n+1/2} \end{Bmatrix} = \begin{Bmatrix} -\bar{Z}_2 \hat{x}_2^n - (\bar{Z}_1 - \delta \bar{Z}_2 K) \hat{x}_1^{n+1/2} \\ z^{n+1/2} - \bar{H}_2 \hat{x}_2^n - (\bar{H}_1 - \delta \bar{H}_2 K) \hat{x}_1^{n+1/2} \end{Bmatrix} \quad (37)$$

which corresponds to a first order filter to reduce the errors in computing $\hat{x}_2 = M\hat{x} + D\hat{x}$. A second-order discrete filter for computing u and $\bar{z}$ can be obtained by differentiating u and $\bar{z}$ to obtain

$$\begin{cases} \dot{u} = -\bar{Z}_1 \dot{\hat{x}}_1 - \bar{Z}_2 \dot{\hat{x}}_2 \\ \dot{\bar{z}} = \dot{z} - \bar{H}_1 \dot{\hat{x}}_1 - \bar{H}_2 \dot{\hat{x}}_2 \end{cases} \quad (38)$$

and then substituting $\dot{\hat{x}}_1$ and $\dot{\hat{x}}_2$ from (17). Subsequently, (19) is applied to integrate the equations for u and $\bar{z}$ which yields

$$\begin{bmatrix} I + \delta \bar{Z}_2 B + \delta^2 \bar{Z}_1 M^{-1} B & \delta(\bar{Z}_2 \bar{L}_2 + \bar{Z}_1 \bar{L}_1 + \delta \bar{Z}_1 M^{-1} \bar{L}_2) \\ \delta(\bar{H}_2 B + \delta \bar{H}_1 M^{-1} B) & I + \delta \bar{H}_1 (\bar{L}_1 + \delta M^{-1} \bar{L}_2) + \delta \bar{H}_2 \bar{L}_2 \end{bmatrix} \begin{Bmatrix} u^{n+1/2} \\ \bar{z}^{n+1/2} \end{Bmatrix} = \begin{Bmatrix} u^n \\ \bar{z}^n \end{Bmatrix} - \delta \begin{Bmatrix} \bar{Z}_1 M^{-1} (\hat{x}_2^n - \delta K \hat{x}_1^{n+1/2} - D \hat{x}_1^{n+1/2}) + \bar{Z}_2 K \hat{x}_1^{n+1/2} \\ \bar{H}_1 M^{-1} (\hat{x}_2^n - \delta K \hat{x}_1^{n+1/2} - D \hat{x}_1^{n+1/2}) + \bar{H}_2 K \hat{x}_1^{n+1/2} \end{Bmatrix} + \begin{Bmatrix} 0 \\ z^{n+1/2} - z^n \end{Bmatrix} \quad (39)$$

The net effects of this augmentation are to filter out the errors committed in estimating both $\hat{x}_1$ and $\hat{x}_2$. Solution of (39) for $u^{n+1/2}$ and $\bar{z}^{n+1/2}$ permits (25) and (26) to be solved in continuous time for $\hat{x}_1^{n+1/2}$ and $\hat{x}_2^{n+1/2}$. Subsequently, (29b) is used for $\hat{x}_1^{n+1}$ and $\hat{x}_2^{n+1}$.

The preceding augmentation (39) leads to an accurate estimate of the control force and observer error correction at the $(n+1/2)$ time step. Although (39) involves the solution of an additional algebraic equation, the equation size is relatively small (size = number of actuators (m) plus the number of measurements (r)). Thus, (39) is an efficient method for continuous time simulation of the Kalman filter equations provided the size of (39) is significantly lower than the first order form of (4). The next section discusses the relative efficiency of the present method and the conventional first order solution. More details on the equation augmentation procedure (39) may be found in Park and Belvin (1991).

Finally, it is noted that by following a similar time discretization procedure adopted for computing $\hat{x}_1^{n+1/2}$ and $\hat{x}_2^{n+1/2}$, the structural dynamics equation (1) can be solved by

$$\begin{cases} (M + \delta D + \delta^2 K)x_1^{n+1/2} = Mx_1^n + \delta x_2^n + \delta^2 Bu^{n+1/2} \\ x_2^{n+1/2} = x_2^n + \delta Bu^{n+1/2} - \delta Kx_1^{n+1/2} \end{cases} \quad (40)$$

Thus, numerical solutions of the structural dynamics equation (1) and the observer equation (20) can be carried out within the second-order solution context, thus realizing substantial computational simplicity compared with the solution of first-order systems of equations (4) and the corresponding first-order observer equations (6).

It is emphasized that the solutions of both the structural displacement x and the reconstructed displacement $\hat{x}$ employ the same solution matrix, $(M + \delta D + \delta^2 K)$. The computational stability of the present procedure can be examined as investigated in Park (1980) and Park and Felippa (1983, 1984). The result, when applied to the present case, can be stated as

$$\delta^2 \lambda_{\max} \leq 1 \quad (41)$$

where $\lambda_{\max}$ is the maximum eigenvalue of

$$(\lambda^2 I + \lambda \bar{Z}_2 B + \bar{Z}_1 M^{-1} B)y = 0 \quad (42)$$

Experience has shown that $|\lambda_{\max}|$ is several orders of magnitude smaller than $\mu_{\max}$ of the structural dynamics eigenvalue problem:

$$\mu M y = K y \quad (43)$$

Considering that a typical explicit algorithm has its stability limit $\mu_{\max} \cdot h \leq 2$, the maximum step size allowed by (42) is in fact several orders of magnitude larger than allowed by any explicit algorithm.

Computational Efficiency

Solution of the Kalman filtering equations in second-order form is prompted by the potential gain in computational efficiency due to the beneficial nature of matrix sparsity and symmetry in the solution matrix of the second-order observer equations. There is an overhead to be paid for the present second-order procedure, that is, the additional computations introduced to minimize the control force and observer error terms on the right-hand-side of the resulting discrete equations. The following paragraphs show the second-order solution is most advantageous for observer models with sparse coefficient matrices M , D and K .

Solution of the first order Kalman filter equation (6) or the second-order form (25-26, 39) may be performed using a time discretization as given by (19). For linear time invariant (LTI) systems, the solution matrix is decomposed once and subsequently upper and lower triangular system solutions are performed to compute the observer state at each time step. Thus, the computations required at each time step result from calculation of the RHS and subsequent triangular system solutions. For the results that follow, the number of floating point operations per second (flops) are estimated for LTI systems of order $O(N)$. In addition, it is assumed that the mass, damping and stiffness matrices (M , D and K) are symmetric and banded with bandwidth αN , where $0 \leq \alpha \leq (0.5 - \frac{1}{2N})$.

The first-order Kalman filter equation (6) requires $(4N^2 + 2Nr + O(N))$ flops at each time step. The discrete time second-order Kalman filter solution (25-26, 33-34) require $(8\alpha^2 N^2 + 2\alpha N^2 + 3Nm + 4Nr + O(N))$ flops and the continuous time second-order Kalman filter (25-26, 39) require $(8\alpha N^2 + 2\alpha N^2 + 5Nm + 6Nr + (r + m)^2 + O(N))$ flops at each time step. To examine the relative efficiency of the first-order and second-order forms, several cases are presented as follows.

First, a worst case condition is examined whereby M , D and K are fully populated ($\alpha = 0.5 - \frac{1}{2N}$) and $r = m = N$. For this condition, the number of flops are:

$$\begin{cases} \text{First Order} & 6N^2 + O(N) \\ \text{Second Order Discrete} & 10N^2 + O(N) \\ \text{Second Order Continuous} & 18N^2 + O(N) \end{cases}$$

Thus, for non-sparse systems with large numbers of sensors and actuators relative to the system order, the first order Kalman filter is 300 percent more efficient than the second-order continuous Kalman filter solution presented herein.

For structural systems, M , and K are almost always banded. In addition, the number of sensors and actuators is usually small compared to the system order N . Hence, the value of α for which the second-order form becomes more computationally attractive than the first order form must be determined. If the assumption is made that the number of

actuators (m) and the number of measurements (r) is proportional to the bandwidth ($r = m = \alpha N$), the value of α which renders the second-order solution more efficient is readily obtained. For the second-order discrete Kalman filter, when $\alpha \leq 0.394$ the second-order form is more efficient. Similarly, the second-order continuous Kalman filter form is more efficient when $\alpha \leq 0.279$. Since α obtains values approaching 0 when a modal based structural representation is used with few sensors and actuators, the second-order form can be substantially more efficient than the classical first-order form. A more detailed discussion can be found in Belvin (1989).

Implementation and Numerical Evaluations

The second-order discrete Kalman filtering equation derived in (25) and (26) have been implemented along with the stabilized form of the controller u and the filtered measurements $\bar{z}$ in such a way the observer computational module can be interfaced with the partitioned control-structure interaction simulation package developed previously (Belvin, 1989; Belvin Park, 1991; Alvin and Park, 1991). Table 1 contrasts the present CSI simulation procedure to conventional procedures. It is emphasized that the solution procedure of the present second-order discrete Kalman filtering equations (25) and (26) follows exactly the same steps as required in the solution of symmetric, sparse structural systems (or the plant dynamics in the jargon of control). It is this attribute that makes the present discrete observer attractive from the simulation viewpoint.

The first example is a truss beam shown in Fig. 1, consisting of 8 bays with nodes 1 and 2 fixed for cantilevered motions. The locations of actuator and sensor applications as well as their directions are given in Table 2. Figures 2, 3 and 4 are the vertical displacement histories at node 9 for open-loop, direct output feedback, and dynamically compensated feedback cases, respectively. Note the effectiveness of the dynamically compensated feedback case by the present second-order discrete Kalman filtering equations as compared with the direct output feedback cases. Figure 5 illustrates a testbed evolutionary model of an Earth-pointing satellite. Eighteen actuators and 18 sensors are applied to the system for vibration control and their locations are provided in Tables 3 and 4. Figures 6, 7, and 8 are a representative of the responses for open-loop, direct output feedback, and dynamically compensated cases, respectively. Note that u_z response by the dynamically compensated case does drift away initially even though the settling time is about the same as that by the direct output feedback case. However, the sensor output are assumed to be noise-free in these two numerical experiments. Although the objective of the present paper is to establish the computational effectiveness of the second-order discrete Kalman filtering equations, we conjecture that for noise-contaminated sensor output for which one would apply dynamic compensated strategies, the relative control performance may turn

out to be the opposite. Further simulations with the present procedure should shed light on the performance of dynamically compensated feedback systems for large-scale systems as they are computationally more feasible than heretofore possible.

Table 5 illustrates the computational overhead associated with the direct output feedback vs. the use of a dynamic compensation scheme by the output present Kalman filtering equations. In the numerical experiments reported herein, we have relied on Matlab software package (Wolfram, 1988) for the synthesis of both the control law gains and the discrete Kalman filter gain matrices. It is seen that the use of the present second-order discrete Kalman filtering equations for constructing dynamically compensated control laws adds computational overhead, only an equivalent of open-loop transient analysis of symmetric sparse systems of order N instead of $2N \times 2N$ dense systems.

Summary

The present paper has addressed the advantageous features of employing the same direct time integration algorithm for solving the structural dynamics equations also to integrate the associated continuous Kalman filtering equations. The time discretization of the resulting Kalman filtering equations is further facilitated by employing a canonical first-order form via a generalized momenta. When used in conjunction with the previously developed stabilized form of control laws (Park and Belvin, 1991), the present procedure offers a substantial computational advantage over the solution methods based on a first-order form when computing with large and sparse observer models.

Computational stability of the present solution method for the observer equation has been assessed based on the stability analysis result of partitioned solution procedures (Park, 1980). To obtain a sharper estimate of the stable step size, a more rigorous computational stability analysis is being carried out and will be reported in the future.

Acknowledgements

The work reported herein was supported by a grant from Air Force Office of Scientific Research, F49620-87-C-0074 and a grant from NASA/Langley Research Center, NAG1-1021. The authors thank Drs. Anthony K. Amos and Spencer Wu of AFOSR for their interest and encouragement and Dr. Jer-Nan Juang of NASA/Langley Research Center who has encouraged us to work on second-order observers.

References

1. K. A. Alvin and K. C. Park, "Implementation of A Partitioned Algorithm for Simulations of Large CSI Problems," Center for Space Structures and Controls, University of Colorado at Boulder, CO., Report No. CU-CSSC-91-4, March 1991.
2. Arnold, W. F. and Laub, A. J. (1984), "Generalized Eigenproblem Algorithms and Software for Algebraic Riccati Equations," *Proceedings of the IEEE*, Vol. 72, No. 12, pp. 1746-1754.
3. Belvin, W. K. (1989), "Simulation and Interdisciplinary Design Methodology for Control-Structure Interaction Systems," PhD Thesis, Center for Space Structures and Controls, University of Colorado at Boulder, CO., Report No. CU-CSSC-89-10, July 1989.
4. Belvin, W. K. and Park, K. C. (1989), "On the State Estimation of Structures with Second Order Observers," *Proc. the 90th Structures, Dynamics and Materials Conference*, AIAA Paper No. 89-1241.
5. Belvin, W. K. and Park, K. C. (1990), "Structural Tailoring and Feedback Control Synthesis: An Interdisciplinary Approach," *J. Guidance, Control and Dynamics*, Vol. 13, No. 3, pp. 424-429.
6. Bender, D. J. and Laub, A. J. (1985), "Controllability and Observability at Infinity of Multivariable Linear Second-Order Models," *IEEE Transactions on Automatic Control*, Vol. AC-30, pp. 1234-1237.
7. Felippa, C. A. and Park, K. C. (1978), "Computational Aspects of Time Integration Procedures in Structural Dynamics, Part 1: Implementation," *Journal of Applied Mechanics*, Vol. 45, pp. 595-602.
8. Hashemipour, H. R. and Laub, A. J. (1988), "Kalman filtering for second-order models," *J. Guidance, Control and Dynamics*, Vol. 11, No. 2, pp.181-185.
9. Hughes, P. C. and Skelton, R. E. (1980), "Controllability and observability of linear matrix second-order systems," *J. Applied Mechanics*, Vol. 47, pp.415-420.
10. Jensen, P. S. (1974), "Transient Analysis of Structures by Stiffly Stable Methods," *Computers and Structures*, Vol. 4, pp.67-94.
11. Juang, J. N. and Maghami, P. G. (1990), "Robust Eigensystem Assignment for Second-Order Estimators," *Proc. of the Guidance, Navigation and Control Conference*, AIAA Paper no. 90-3474.
12. Kalman, R. E. (1961), "On the General Theory of Control Systems," *Proc. 1st International Congress on Automatic Control*, Butterworth, London, Vol. 1, pp. 481-491.
13. Kalman, R. E. and Bucy, R. S. (1961), "New results in linear filtering and prediction theory," *Trans, ASME J. Basic Engineering*, Vol. 83, pp. 95-108.

14. Kwakernaak, H. and Sivan, R. (1972), *Linear Optimal Control Systems*, Wiley-Interscience, New York.
15. Oshman, Y., Inman, D. J. and Laub, A. J. (1989), "Square-Root State Estimation for Second-Order Large Space Structures Models," *Journal of Guidance, Control and Dynamics*, Vol. 12, no. 5, pp.698-708.
16. Park, K. C.(1980), "Partitioned Analysis Procedures for Coupled-Field Problems: Stability Analysis," *Journal of Applied Mechanics*, Vol. 47, pp. 370-378.
17. Park, K. C. and Felippa, C. A. (1983), "Partitioned Analysis of Coupled Systems," in: *Computational Methods for Transient Analysis*, T. Belytschko and T. J. R. Hughes (eds.), Elsevier Pub. Co., pp. 157-219.
18. Park, K. C. and Belvin, W. K. (1989), "Stability and Implementation of Partitioned CSI Solution Procedures," *Proc. the 30th Structures, Dynamics and Materials Conference*, AIAA Paper No. 89-1238.
19. Park, K. C. and Felippa, C. A. (1984), "Recent Developments in Coupled-Field Analysis Methods," in: *Numerical Methods in Coupled Systems*, Lewis, R. W. et al(editors), John Wiley & Sons, pp. 327-352.
20. Wolfram, S., *MathematicaTM*, Addison-Wesley Pub. Co., 1988.

$$\begin{cases}
\text{Structure:} & a) \quad M\ddot{q} + D\dot{q} + Kq = f + Bu + Gw \\
& \quad q(0) = q_0, \quad \dot{q}(0) = \dot{q}_0 \\
\text{Sensor Output:} & b) \quad z = Hx + v \\
\text{Estimator:} & c) \quad M\ddot{\tilde{q}} + D\dot{\tilde{q}} + K\tilde{q} = f + Bu + ML_2\gamma \\
& \quad \tilde{q}(0) = 0, \quad \dot{\tilde{q}}(0) = 0 \\
\text{Control Force:} & d) \quad \dot{u} + F_2 M^{-1} Bu = F_2(M^{-1}\dot{\tilde{p}} + L_2\gamma) + F_1\dot{\tilde{q}} \\
\text{Estimation Error:} & e) \quad \dot{\gamma} + H_v L_2 \gamma = \dot{z} - H_v M^{-1}(\dot{\tilde{p}} - Bu) - H_d \dot{\tilde{q}}
\end{cases}$$

Table 1a Partitioned Control-Structure Interaction Equations

$$\begin{cases}
\text{Structure:} & a) \quad \dot{x} = Ax + Ef + \bar{B}u + \bar{G}w \\
& \quad q(0) = q_0, \quad \dot{q}(0) = \dot{q}_0 \\
\text{Sensor Output:} & b) \quad z = Hx + v \\
\text{Estimator:} & c) \quad \dot{\tilde{x}} = A\tilde{x} + Ef + \bar{B}u + L\gamma \\
& \quad \tilde{x}(0) = 0 \\
\text{Control Force:} & d) \quad u = -F\tilde{x} \\
\text{Estimation Error:} & e) \quad \gamma = z - (H_d \tilde{q} + H_v \dot{\tilde{q}})
\end{cases}$$

where

$$x = \begin{Bmatrix} q \\ \dot{q} \end{Bmatrix}, \quad \tilde{x} = \begin{Bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{Bmatrix}$$

and

$$\begin{aligned}
H &= [H_d \quad H_v], \quad L = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix}, \quad E = \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix} \\
A &= \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{bmatrix}, \quad \bar{B} = \begin{bmatrix} 0 \\ M^{-1}B \end{bmatrix}, \quad F = [F_1 \quad F_2]
\end{aligned}$$

Table 1b Conventional Control-Structure Interaction Equations

TABLE 2a:
Actuator Placement for Truss Example Problem

Actuator	Node	Component
1	2	y
2	18	y
3	9	y
4	9	x

TABLE 2b:
Sensor Placement for Truss Example Problem

Sensor	Type	Node	Component
1	Rate	2	y
2	Rate	18	y
3	Rate	9	y
4	Rate	9	x
5	Position	9	y
6	Position	9	x

TABLE 3:
Actuator Placement for EPS Example Problem

Actuator	Node	Component
1	97	x
2	97	z
3	96	x
4	96	z
5	65	y
6	68	y
7	59	y
8	62	y
9	45	y
10	45	z
11	70	y
12	70	z
13	95	x
14	95	y
15	95	z
16	95	ϕ_x
17	95	ϕ_y
18	95	ϕ_z

TABLE 4:
Sensor Placement for EPS Example Problem

Sensor	Type	Node	Component
1	Rate	97	x
2	Rate	97	z
3	Rate	96	x
4	Rate	96	z
5	Rate	65	y
6	Rate	68	y
7	Rate	59	y
8	Rate	62	y
9	Rate	45	y
10	Rate	45	z
11	Rate	70	y
12	Rate	70	z
13	Position	95	x
14	Position	95	y
15	Position	95	z
16	Position	95	ϕ_x
17	Position	95	ϕ_y
18	Position	95	ϕ_z

TABLE 5:
CPU Results for ACSIS Sequential and Parallel Versions

Model	Problem Type	Sequential	Parallel
54 DOF Truss	Transient	4.5	5.6
	FSFB	9.4	10.2
	K. Filter	13.0	10.7
582 DOF EPS7	Transient	98.6	100.3
	FSFB	190.2	294.5
	K. Filter	284.2	321.5

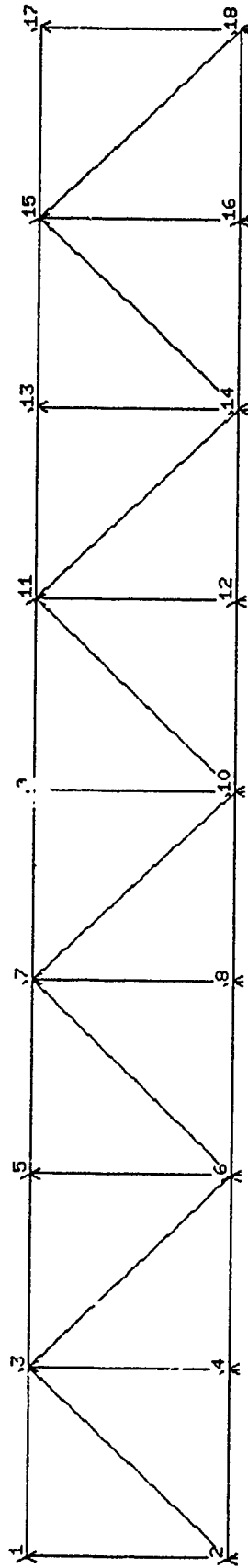


Figure 1: Truss Beam Problem

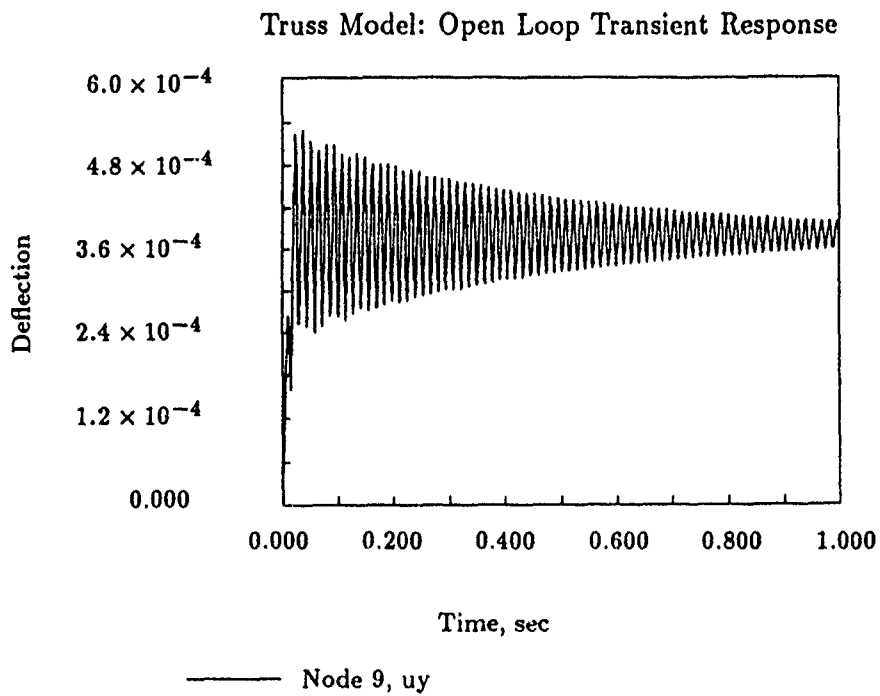


Figure 2: Truss Transient Response

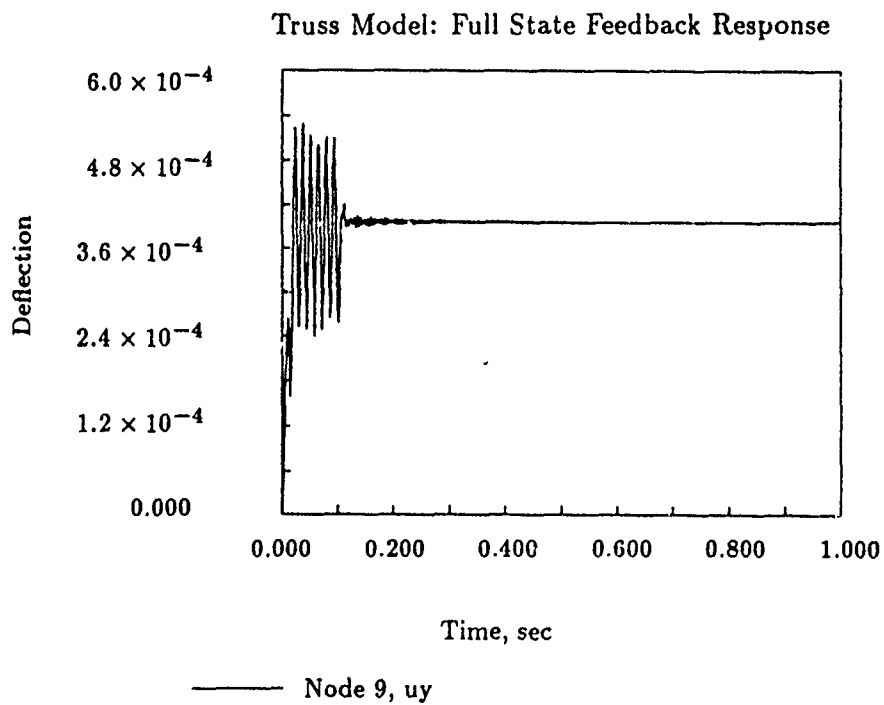


Figure 3: Truss FSFB Response

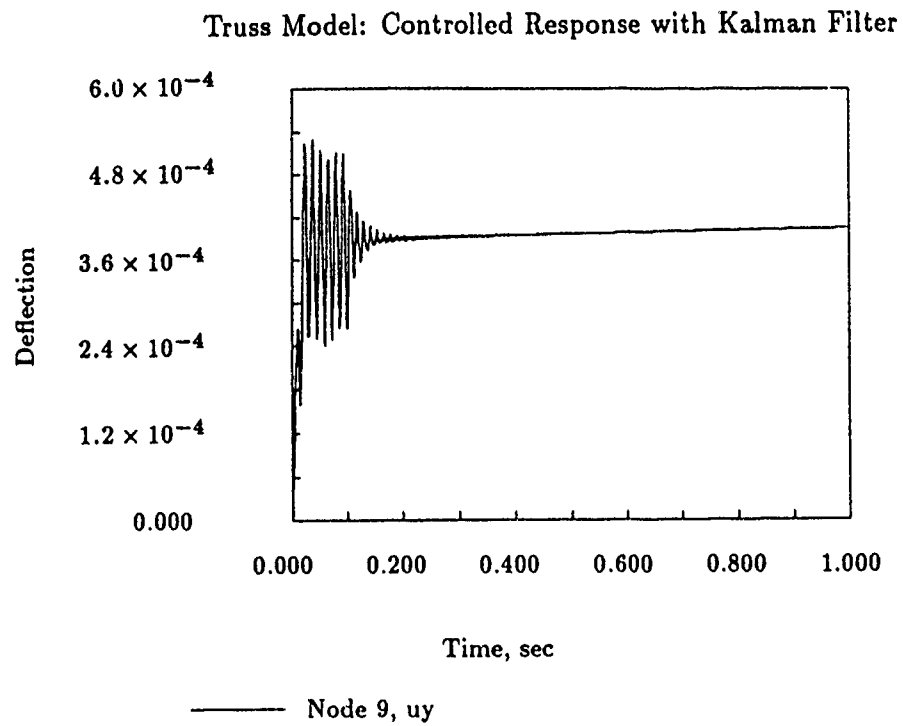


Figure 4: Truss Response with Filter

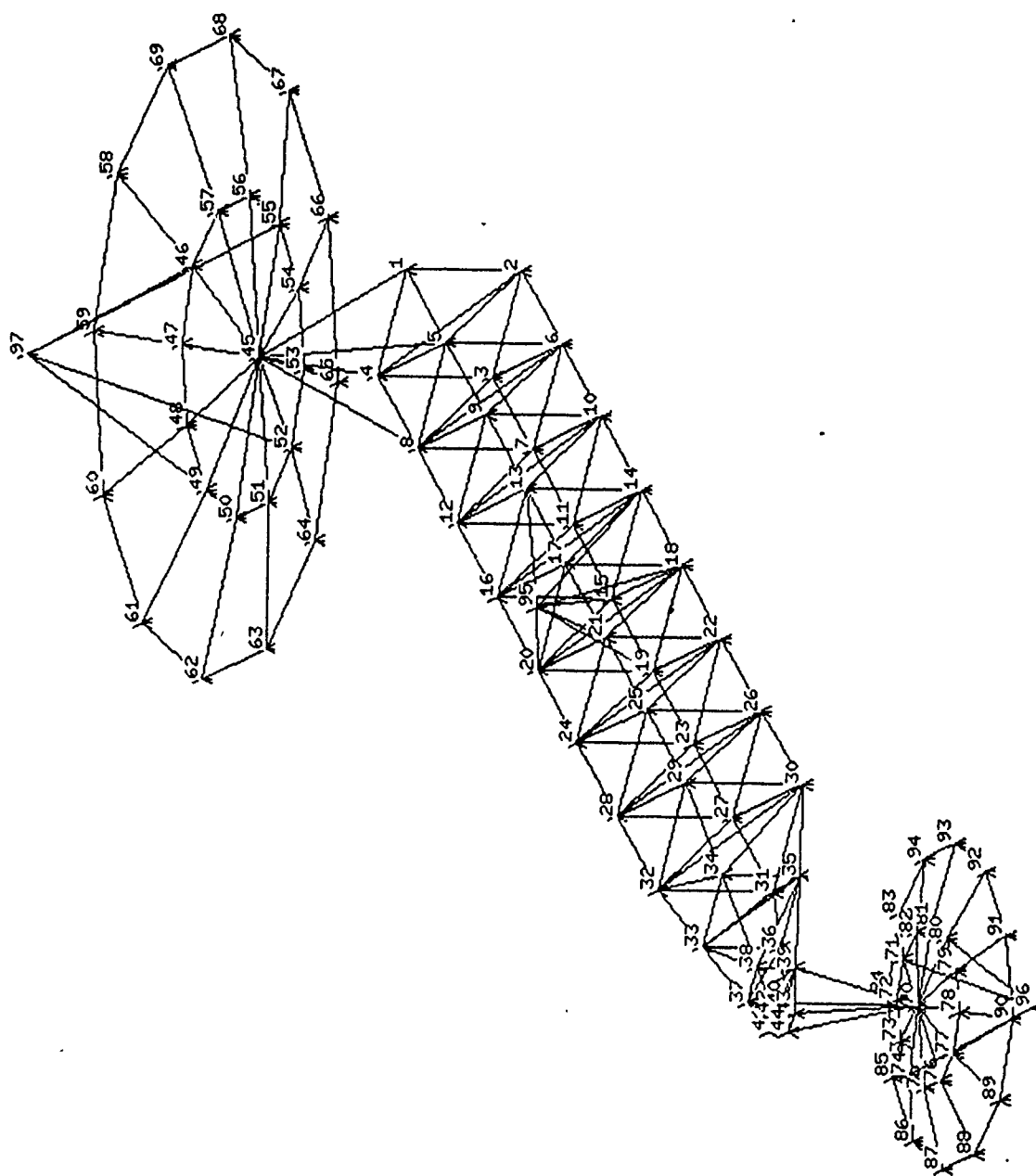


Figure 5: Evolutionary Earth-Pointing Satellite

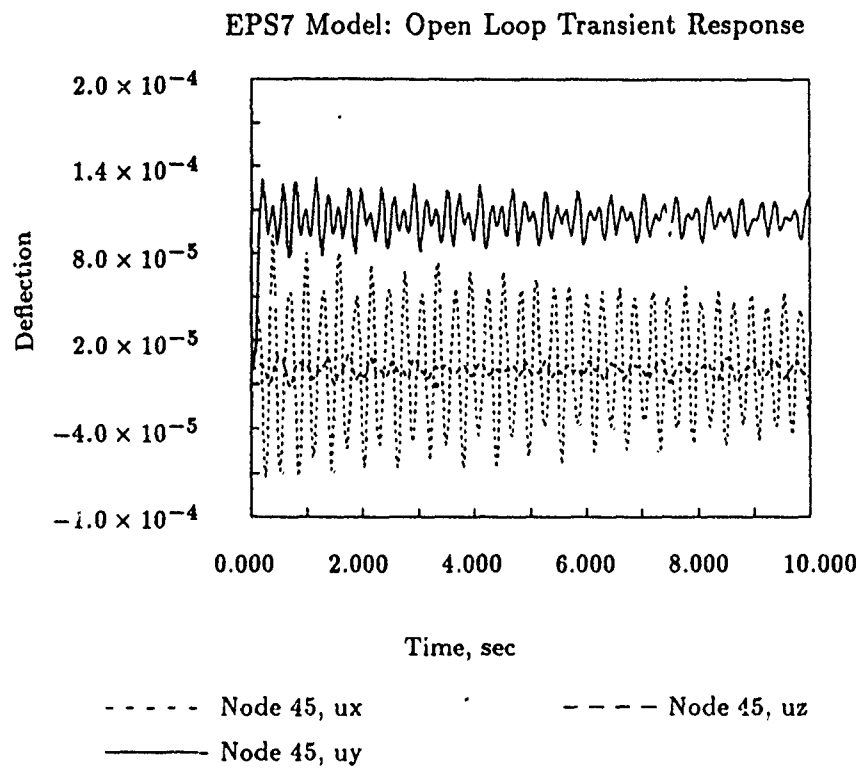


Figure 6: EPS Transient Response

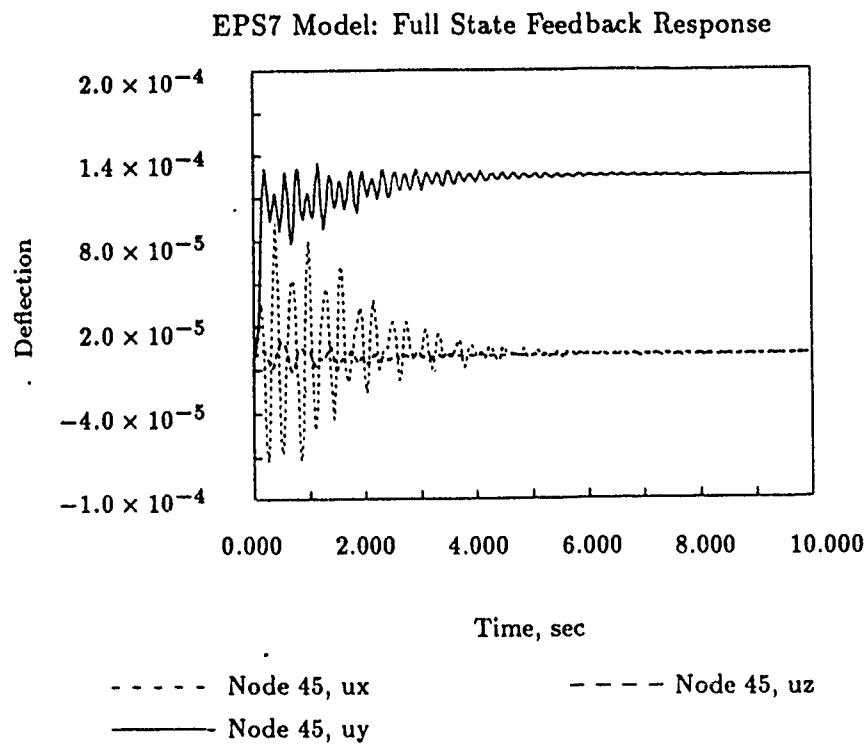


Figure 7: EPS FSFB Response

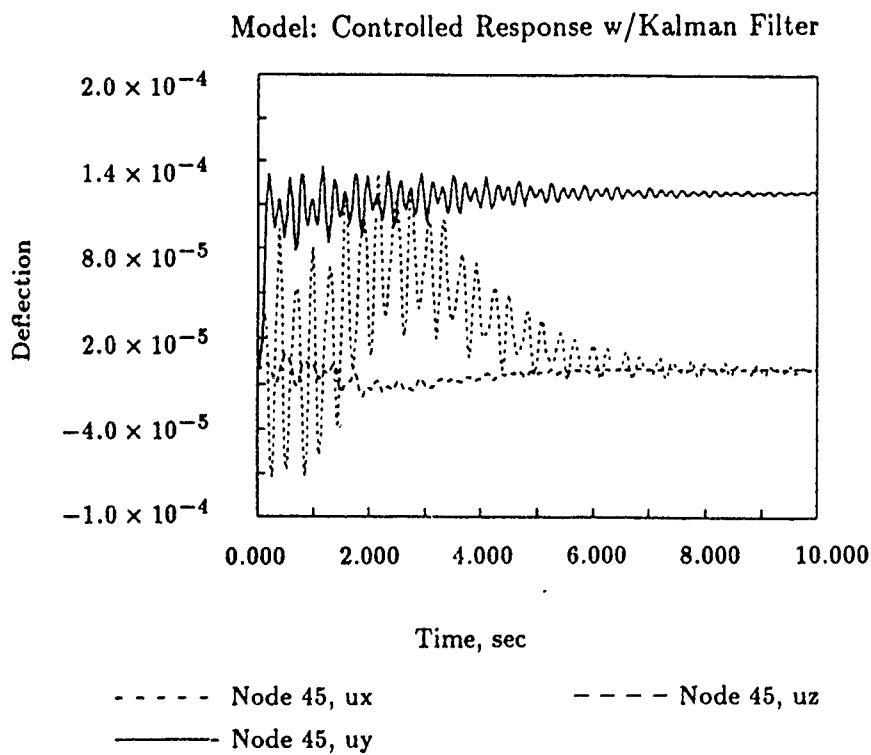


Figure 8: EPS Response with Filter

On the State Estimation of Structures with Second Order Observers

W. Keith Belvin¹
NASA Langley Research Center
Hampton, Virginia 23665

K. C. Park²
University of Colorado
Boulder, Colorado 80309

Abstract

The use of Linear Quadratic Regulator (LQR) control synthesis techniques implies the availability of full state feedback. For vibration control of structures, usually only a limited number of states are measured from which an observer model reconstructs the full state. This paper shows that using second order observers is a viable technique for reconstructing the unmeasured states of structures under mildly restrictive conditions. Moreover, the computational advantages of the second order observer as compared to a first order observer indicate that significantly larger observer models may be utilized. Numerical examples are used to demonstrate the performance of second order observers. The implications of second order observers in the development of Control/Structures Interaction (CSI) technology is discussed.

1. Introduction

Recent emphasis on the optimal interdisciplinary design of actively controlled spacecraft has led to the formation of a focused research activity within NASA called Control/Structures Interaction (CSI). The aim of the CSI activity is to coordinate University, Industry and Government research to develop the technology required to make active control of flexible spacecraft routine. To this end, several investigators have begun to exploit the second order form of the differential equations which describe structural dynamics. The controllability and observability of linear second order models has been studied by Laub and Arnold¹, Hughes and Skelton² and by Bender and Laub.³

Park and Belvin^{4,5} demonstrated the computational advantages of simulating the coupled CSI equations in second order form. Unlike classical first order simulation of the CSI equations, simulation in second order form enables the symmetry and sparsity of the structural equations to be exploited. Significant computational efficiency is gained by treating the CSI equations in second order form. Moreover, a natural partitioning of controls and structures is indicated by maintaining the structural equations in second order form.⁶

The linear quadratic regulator (LQR) control synthesis technique has excellent robustness properties. Safonov and

Athens⁶ have proven, when the full state is available, that the LQR technique provides 60° of phase margin and infinite gain margin. However, for space structures the full state is not available and an observer must be used in conjunction with LQR. No guarantees can be made on the robustness properties for the observer based LQR controller.⁷ Thus the observer plays a key role in the study of CSI when full state feedback is employed.

Observer state reconstruction suffers from model errors, measurement noise and unmodeled dynamics. Unmodeled dynamics result from the use of reduced order models.⁸ To reduce the level of unmodeled dynamics, the size of the observer could be increased, provided the observer solution can still be performed in real-time. The computational advantages associated with the second order observer may permit larger observers with no penalty in solution time as compared to the first order observer.

Second order observer models have received recent attention in the literature. Hashemipour and Laub⁹ utilized an optimal observer known as the Kalman Filter for discretized (time and space) second order structures models. In addition, robust computational procedures for solution of the Kalman Filter estimation error covariance matrices have been developed for second order models in Ref. 10. Unfortunately, the continuous time Kalman Filter has only been successfully developed in first order form.

It seems paradoxical that the optimal way to estimate the states of a second order system is with a first order observer model. Although there is more design freedom when using a first order model, the disadvantage of numerically simulating the CSI equations in first order form may prove that sub-optimal second order observers are most practical. It is from this perspective that some analytical and numerical results are presented herein to highlight the advantages and disadvantages of using second order observers to estimate the states of realistic spacecraft.

In section 2, the governing CSI equations are presented using nomenclature similar to Refs. 9 and 10. It includes the classical first order observer and a logical form for the second order observer. Stability and performance of the observers are discussed in Section 3. Practical computational aspects of the observers are examined in Section 4. Section 5 presents numerical results which demonstrate the viability of second order observers. In addition, implications of the second order observer on CSI technology are discussed.

¹ Structural Dynamics Division, NASA Langley Research Center. Member AIAA.

² Professor of Aerospace Engineering, University of Colorado. Member AIAA.

2. Controlled Structure Simulation Equations

A control-structure interaction system can be represented as shown in Fig. 1. The linear time-invariant discretized equations of motion for the structure (plant) may be described by the following set of continuous-time equations:

$$\begin{aligned} M\ddot{q} + D\dot{q} + Kq &= f + Bu + Gw \\ q(0) &= q_0, \quad \dot{q}(0) = \dot{q}_0 \\ z &= H_d q + H_v \dot{q} + v \\ u &= F_1 q + F_2 \dot{q} \end{aligned} \quad (2.1)$$

where M , D , K are the mass, damping and stiffness matrices, respectively. These matrices, typically derived using finite element modeling, are symmetric and sparse. The vector f represents known, state independent, applied forces, B is the actuator location matrix and u is the state dependent control force. The matrix G describes the disturbance location and w is the disturbance vector. The displacement and velocity initial conditions are given by q_0 and $\dot{q}_0$, respectively. The measured output of the system is z where v is a vector of measurement noise. The matrices H_d and H_v represent the displacement and velocity sensor locations and F_1 and F_2 are feedback gain matrices of the controller.

The conventional technique for dealing with multi-variable control problems is to convert the system to first order state-space form.

$$\begin{aligned} \dot{x} &= Ax + Ef + \tilde{B}u + \tilde{G}w \\ z &= Hx + v, \quad x(0) = x_0 \\ u &= Fx \end{aligned} \quad (2.2)$$

where

$$\begin{aligned} A &= \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} 0 \\ M^{-1}B \end{bmatrix} \\ E &= \begin{bmatrix} 0 \\ M^{-1}G \end{bmatrix}, \quad \tilde{G} = \begin{bmatrix} 0 \\ M^{-1}G \end{bmatrix} \\ H &= [H_d \ H_v], \quad F = [F_1 \ F_2] \\ x &= \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \end{aligned}$$

The control gain F may be synthesized by a number of techniques as given in standard texts such as Ref. 11. When full state feedback is used, and the full state cannot be measured, an observer model must be developed to estimate the states of the entire structure from the measured states. The form of the first order and second order observer models is discussed below.

2.1 First Order Observer Model

The observer model is usually constructed by augmenting (2.2) by a state correction term such that

$$\begin{aligned} \dot{\tilde{x}} &= \tilde{A}\tilde{x} + \tilde{E}f + \tilde{B}u + L(z - \tilde{H}\tilde{x}) \\ \tilde{x}(0) &= 0 \end{aligned} \quad (2.3)$$

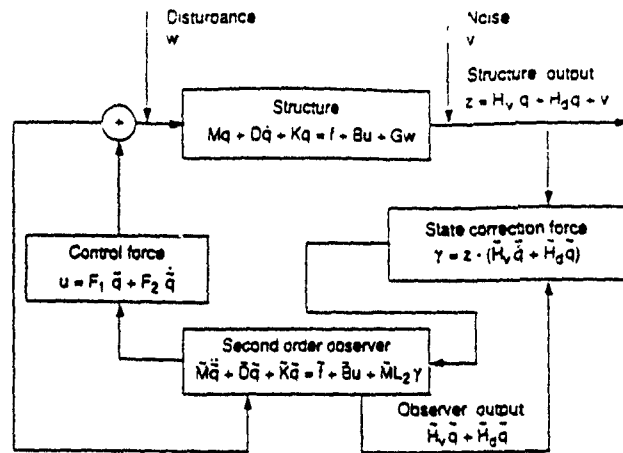


Figure 1. Typical Control/Structure Interaction System

where

$$L = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix}, \quad \tilde{x} = \begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{bmatrix}$$

and $\tilde{A}$, $\tilde{E}$, $\tilde{B}$ and $\tilde{H}$ are a reduced order description of the structural system, actuator and sensor locations. The vector $\tilde{x}$ is the estimated state. Adding the state correction term $L(z - \tilde{H}\tilde{x})$ forces the observer model to track the measured states of the structure. Since the disturbance is unknown, it is not included in the observer.

The observer gain matrix L can be synthesized by a number of techniques. If covariance matrices of the noise and disturbance processes in (2.1) are available, the Kalman Filter can be constructed.⁹⁻¹¹ The Kalman Filter provides an optimal balance of observer performance and noise rejection; however, a first order observer is required. Since a structure is represented with second order models, it seems natural to choose a second order observer model for estimation of structural states. The second order observer is described below.

2.2 Second Order Observer Model

A linear observer model could be derived by adding a state correction term to a reduced order form of (2.1), such that the estimated states $\tilde{q}$ and $\dot{\tilde{q}}$ are computed from

$$\begin{aligned} \tilde{M}\ddot{\tilde{q}} + \tilde{D}\dot{\tilde{q}} + \tilde{K}\tilde{q} &= \tilde{E}f + \tilde{B}u + \tilde{M}\tilde{L}_2\gamma \\ \tilde{q}(0) &= 0, \quad \dot{\tilde{q}}(0) = 0 \end{aligned} \quad (2.4)$$

where $\tilde{M}$, $\tilde{D}$ and $\tilde{K}$ represent a reduced order model of the structure and γ is a state correction term of the form

$$\gamma = z - (\tilde{H}_d \tilde{q} + \tilde{H}_v \dot{\tilde{q}})$$

The second order observer form given by (2.4) may be solved efficiently by keeping the control and state dependent state correction terms on the right-hand-side (RHS) of the equation. This permits the symmetry and sparsity of the observer mass, damping and stiffness matrices to be exploited by the

computational algorithms. References 4 and 5 present a partitioned solution procedure for this type of equation.

From (2.4) one observes the absence of L_1 , which must be null for a second order observer to be constructed. The absence of L_1 is needed to establish the identity relation $\ddot{q} = \dot{\ddot{q}}$. A second order system is written in first order form by using the relation that the time derivative of position is equal to the velocity as shown in the top row of equation (2.2). To transform a first order system to a second order one, the identity relation must exist. The effects of $L_1 = 0$ are described in the next section.

3. Stability and Performance of Observers

To study the ability of the observer to predict the states of the structure, the equation for the error between the actual states and the predicted states, $e = (q - \hat{q})$ is examined. To simplify the discussion, we consider an unreduced observer $\hat{A} = A$, $\hat{B} = B$, $\hat{H} = H$. The deterministic error is given by

$$\dot{e} = \begin{bmatrix} -L_1 H_d & I - L_1 H_v \\ -M^{-1}K - L_2 H_d & -M^{-1}D - L_2 H_v \end{bmatrix} e \quad (3.1)$$

From (3.1), it is seen that the error asymptotically goes to zero, provided the matrix $[A - LH]$ is stable. The next section describes the stability limitation of second order observers using a simple single-degree-of-freedom model.

3.1 Observer Stability for a SDOF Model

A deterministic single-degree-of-freedom (SDOF) model consisting of a spring (k), a damper (d) and a mass (m) is used in this section to highlight differences in stability between the first order and second order observer models. The system equations in first order form may be written for the structure as

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -k/m & -d/m \end{bmatrix} x + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} f + \begin{bmatrix} 0 \\ b/m \end{bmatrix} u \quad (3.2)$$

and for the observer as

$$\dot{\hat{x}} = \begin{bmatrix} -L_1 H_d & 1 - L_1 H_v \\ -k/m - L_2 H_d & -d/m - L_2 H_v \end{bmatrix} \hat{x} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} f + \begin{bmatrix} 0 \\ b/m \end{bmatrix} u + \begin{bmatrix} L_1 H_d & L_1 H_v \\ L_2 H_d & L_2 H_v \end{bmatrix} z \quad (3.3)$$

The stability of the observer can be determined from the poles of the error equation

$$\dot{e} = \begin{bmatrix} -L_1 H_d & 1 - L_1 H_v \\ -k/m - L_2 H_d & -d/m - L_2 H_v \end{bmatrix} e \quad (3.4)$$

The poles of $(A - LH)$ in (3.4) are

$$\lambda_{1,2} = \sigma \pm j\omega \quad (3.5)$$

where

$$\sigma = \frac{-(d/m + L_2 H_v + L_1 H_d)}{2}$$

$$\omega^2 = [4k + 2L_1 H_d d + 4m L_2 H_d - m(L_1 H_d + L_2 H_v)^2 - 2L_2 H_v d - 4L_1 H_v k - d^2/m]/4m$$

The pole locations given by (3.5) show the effect of the observer gains on the error between the actual and predicted states. The L_1 gain increases the rate of decay of the error when position measurements are used. The L_1 gain also effects the frequency of the error response when velocity measurements are used. The L_2 gain uses the velocity measurement to increase the rate of decay and the L_2 gain uses position measurement to effect the frequency of the error response.

From the above considerations, distinct differences are seen in the first order and second order observers ($L_1 = 0$). The first order observer can produce a stable reconstruction error utilizing position and/or velocity measurements. The second order observer poles are given by

$$\sigma = \frac{-(d/m + L_2 H_v)}{2}$$

$$\omega^2 = \frac{4k + 4m L_2 H_d - m(L_2 H_v)^2 - 2L_2 H_v d - d^2/m}{4m}$$

which show velocity measurements must be available to augment the error decay rate. For example, if the structure was undamped, ($d = 0$), and no velocity measurements were made ($H_v = 0$) then the first order observer would have the real part of the poles located at

$$\sigma = \frac{-L_1 H_d}{2}$$

Thus the first order observer would be stable. However, the second order observer would be unstable, $\sigma = 0$. Even though stability is gained by a small amount of damping, ($d \neq 0$), the second order observer would have poor performance without augmenting the damping through velocity feedback. Observer performance is discussed in the next section.

3.2 Performance Considerations

The performance of the observer depends on the eigenvalues and eigenvectors of the matrix $[A - LH]$. The eigenvalues determine the speed and rate of observer convergence. Rule of thumb choices for the eigenvalues should be somewhat larger than the closed loop system $[A + BF]$, but not exceedingly so, to prevent problems due to high frequency noise. The eigenvalues and eigenvectors of the observer matrix can be placed to maximize noise and disturbance rejection. Because the second order observer restricts the design, $L_1 = 0$, some of the freedom in placing the eigenvalues and eigenvectors is lost.

Robust methods for designing second order observer gains remains a fruitful area of research. The results herein have synthesized Kalman Filter gains and subsequently ignored the L_1 gain. This approach may or may not lead to a stable observer. If L_2 is not stabilizing it must be augmented. Even if L_2 is stabilizing some loss of performance is expected, par-

ticularly in terms of noise rejection. However, this may be a reasonable penalty for the increased computational speed gained by using a second order observer model which could be translated into reducing the level of unmodeled dynamics. The model error (uncertainty) would be the same for the first or second order observer. Future research is needed to determine the robustness characteristics of second order observers.

4. Computational Requirements for Simulation

Solution of the CSI equations is typically performed using digital computations. It is this consideration that leads to the study of second order observers for CSI technology. As shown in the following paragraphs, the second order observer provides the potential for larger observers and/or faster computational solutions than can be realized with the first order observer

4.1 First Order Observer Solution

The first order observer given by (2.3) may be solved by several methods. One popular method is to compute the matrix exponential of

$$\mathbf{A}_o = [\tilde{\mathbf{A}} + \tilde{\mathbf{B}}\mathbf{F} - \mathbf{L}\tilde{\mathbf{H}}] \quad (4.1)$$

so that

$$\tilde{\mathbf{x}}^{n+1} = \Phi(h)\tilde{\mathbf{x}}^n + \Gamma(h)(\mathbf{E}\mathbf{f}^n + \mathbf{L}\mathbf{z}^n) \quad (4.2)$$

where

$$\Phi(h) = e^{\mathbf{A}_o h}, \quad \Gamma(h) = \int_0^h e^{\mathbf{A}_o s} ds$$

and h is the temporal step size.

Defining N as the number of structural degrees of freedom, then $\Phi(h)$ is a $2N$ by $2N$ matrix.

Because $\Phi(h)$ is computed once for time invariant systems, the matrix exponential computation does not significantly impact the total time required for a long simulation. However, computing the matrix exponential for large systems does cause accuracy and storage problems. $\Phi(h)$ occupies $4N^2$ storage locations since the matrix is typically not symmetric or sparse. Independent of the number of sensors, (4.2) requires, as a minimum, multiplication of $\Phi(h)$ by a vector at each time step. This represents $4N^2$ operations at each step of the simulation.

An alternative approach to the matrix exponential route is to discretize the equations of motion in time to derive difference equations. This approach has been used by the present authors in Refs. 4 and 5. Midpoint implicit integration formulas were used to discretize (2.3) to derive a linear equation

$$\begin{aligned} \mathbf{S}\tilde{\mathbf{x}}^{n+1/2} &= \mathbf{g}^{n+1/2} \\ \tilde{\mathbf{x}}^{n+1} &= 2\tilde{\mathbf{x}}^{n+1/2} - \tilde{\mathbf{x}}^n \end{aligned} \quad (4.3)$$

where

$$\mathbf{S} = \mathbf{I} - \frac{h}{2}\mathbf{A}_o, \quad \mathbf{g} = \frac{h}{2}(\mathbf{E}\mathbf{f}^{n+1/2} + \mathbf{L}\mathbf{z}^{n+1/2}) + \tilde{\mathbf{x}}^n$$

To solve (4.3) an L-U decomposition is performed once for time invariant systems and subsequently a lower and upper triangular system is solved at each time step. Since the L-U decomposition of $\mathbf{S}$ is not sparse or symmetric, $4N^2$ storage locations are required. In addition, the two triangular system solutions require $4N^2$ operations. Thus, the first order observer requires the same computational resources for both the matrix exponential approach and the time discretization approach.

4.2 Second Order Observer Solution

The second order observer model given in (2.4) can be solved using the time discretization approach as presented in Refs. 4 and 5. A summary of the procedure is presented herein.

We express a set of the mid-point implicit formulas with the step size h :

$$\begin{cases} \tilde{\mathbf{q}}^{n+1/2} = \tilde{\mathbf{q}}^n + \delta \dot{\tilde{\mathbf{q}}}^{n+1/2}, & \delta = h/2 \\ \dot{\tilde{\mathbf{q}}}^{n+1/2} = \dot{\tilde{\mathbf{q}}}^n + \delta \ddot{\tilde{\mathbf{q}}}^{n+1/2} \\ \tilde{\mathbf{q}}^{n+1} = 2\tilde{\mathbf{q}}^{n+1/2} - \tilde{\mathbf{q}}^n \end{cases} \quad (4.4)$$

The selection of the mid-point implicit formula (or the trapezoidal rule) for controlled structures is due to its minimal frequency distortion and no numerical damping characteristics.

Implicit time discretization of (2.4) using (4.4) yields the following difference equation:

$$\begin{cases} \tilde{\mathbf{S}}\tilde{\mathbf{q}}^{n+1/2} = \tilde{\mathbf{g}}^{n+1/2} \\ \tilde{\mathbf{S}} = \tilde{\mathbf{M}} + \delta\tilde{\mathbf{D}} + \delta^2\tilde{\mathbf{K}} \\ \tilde{\mathbf{g}}^{n+1/2} = \delta^2(\mathbf{f}^{n+1/2} + \tilde{\mathbf{B}}\mathbf{u}^{n+1/2} + \tilde{\mathbf{M}}\mathbf{L}_2\gamma^{n+1/2}) \\ \quad + \tilde{\mathbf{M}}(\tilde{\mathbf{q}}^n + \delta\dot{\tilde{\mathbf{q}}}^n) + \delta\tilde{\mathbf{D}}\dot{\tilde{\mathbf{q}}}^n \\ \tilde{\mathbf{q}}^{n+1} = 2\tilde{\mathbf{q}}^{n+1/2} - \tilde{\mathbf{q}}^n \\ \dot{\tilde{\mathbf{q}}}^{n+1/2} = (\tilde{\mathbf{q}}^{n+1/2} - \tilde{\mathbf{q}}^n)/\delta \end{cases} \quad (4.5)$$

From (4.5) it can be seen that $\mathbf{u}^{n+1/2}$ and $\gamma^{n+1/2}$ are required to numerically solve for $\tilde{\mathbf{q}}^{n+1/2}$. The partitioned solution procedure^{5,6} uses equation augmentation to predict $\mathbf{u}^{n+1/2}$ and $\gamma^{n+1/2}$ thereby maintaining the symmetry and sparsity of the $\tilde{\mathbf{S}}$ matrix even after L-U decomposition. For the second order observer, $\tilde{\mathbf{S}}$ is an N by N matrix. If the bandwidth of $\tilde{\mathbf{S}}$ is proportional to N by a constant α :

$$\text{Bandwidth} = \alpha N$$

then, only αN^2 storage location are needed. Most importantly, only $2N(\alpha N + 1)$ operations are needed for the two triangular system solutions at each time step.

Comparing the second order observer minimum solution time to the first order observer minimum solution time yields

$$\frac{(2\alpha N^2 + 2N)}{4N^2} = \frac{\alpha + \frac{1}{N}}{2}$$

Note this is an approximate count of the number of operations as the right-hand-side of (4.2) and (4.5) require additional operations. Nevertheless, for a typical bandwidth of 10 percent, and $N \gg 1$, the second order observer requires approximately 5 percent of the number of operations as does the first order observer. (Reduced order observer

models based on a modal description of the structure yields $\alpha N = 1!$) Such a drastic reduction in computations may lead to new applications of observers in CSI technology.

5. Results and Discussion

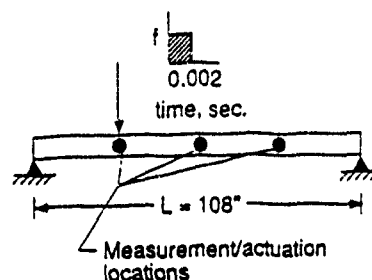
To evaluate the performance of the second order observers numerically, several examples are presented. The first example is relatively simple, a beam with an unreduced observer model. The second example is more realistic in that a large truth model is used in conjunction with a reduced order observer. The encouraging results from these examples prompt us to consider serious use of second order observers in CSI technology.

5.1 Beam Example

To study the effects of approximating a first order observer by a second order observer, the beam model shown in Fig. 2 has been used for numerical studies. The beam is subjected to a step load for 0.002 seconds and a regulator control system is activated at $t=0.01$ sec. A finite element model was constructed with 8 planer beam elements to form a 27 degree of freedom representation of the beam. A three mode modal space control law was used with the performance constraint that the vibration amplitude must be less than 0.025 in. within 0.1 sec. after the control system is activated. Lateral position and velocity measurements were made at each of the three locations indicated in Fig. 2 for a total of 6 measurements. A Kalman Filter was constructed to predict the unmeasured states of the system. Results for the unreduced structure and observer are shown in Fig. 3. The observer, shown by the dashed line, rapidly converges to the true position of the beam lateral deflection (measured at the point of loading).

To determine the viability of a second order observer, L_1 was set equal to zero and the observer equation was solved in second order form. The response of the second order observer is shown in Fig. 4. The second order observer is slower to converge to the actual beam deflection than the first order observer. Nevertheless, the second order observer did converge, thus, the observer stability was maintained when L_1 was neglected.

The beam response with the control law based on a perfect observer (all states are reconstructed with no error), the first order observer and the second order observer is presented in Fig. 5. Although the second order observer produced noticeable performance degradation in the initial time after the control system was activated, it achieved nearly the same level of vibration attenuation after $t=0.1$ sec. as did the perfect observer. Thus, the second order observer is quite viable for this example where both position and velocity measurements were available.



Properties	Open loop frequencies (Hz)
$E = 40E + 6 \text{ lb/in.}^2$	$f_1 = 56.06$
$G = 2.4E + 6 \text{ lb/in.}^2$	$f_2 = 233.22$
$\rho = 1.132E - 4 \text{ lb-s}^2/\text{in.}$	$f_3 = 556.73$
	$f_4 = 2593.6$

Figure 2. Pinned-Pinned Beam Structure

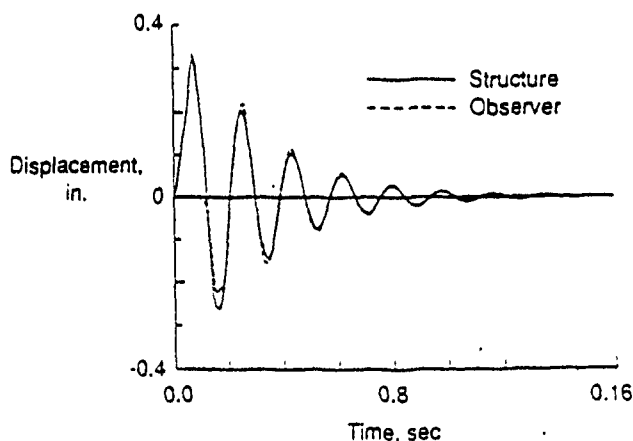


Figure 3. Beam and First Order Observer Response

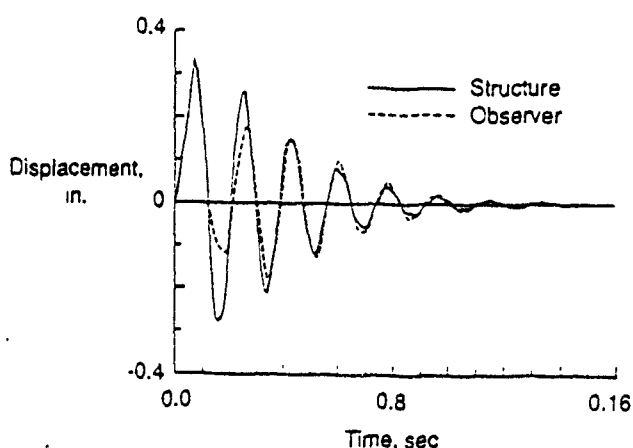


Figure 4. Beam and Second Order Observer Response

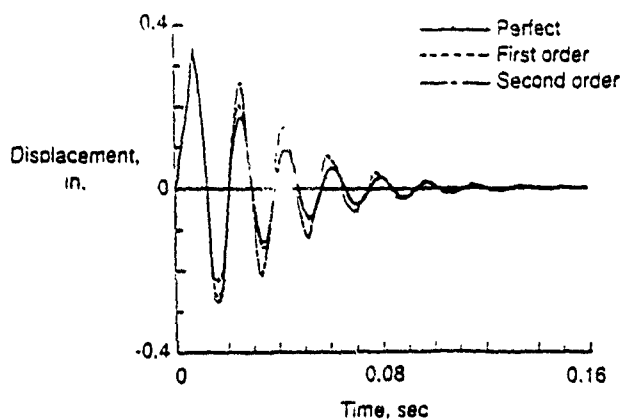


Figure 5. Comparison of Beam Response with Different Observers

5.2 Earth Pointing Satellite Example

A more realistic example of spacecraft being studied for active vibration control is the Earth Pointing Satellite (EPS) shown in Fig. 6. This model is a derivative of a geostationary platform designed for study of Earth Observation Sciences (EOS). Two flexible antennas are attached to a truss (bus). Typical missions involve pointing one antenna to earth, while tracking with the other antenna.

The truss has a three meter square cross section. It consists of 51 mm diameter by 1.59 mm wall thickness graphite epoxy tubes. The modulus of elasticity and mass density of the tubes are $275.9 \text{ E}+09 \text{ N/M}^2$ and 3250.0 Kg/M^3 , respectively. Although the antennas have the same tube dimensions as the truss, nonstructural mass make the effective tube density $= 9750.0 \text{ Kg/M}^3$ for the two antennas. The finite element representation of the EPS consisted of 570 degrees of freedom. Table 1. lists the first 20 frequencies of the EPS structure.

Table 1. EPS Vibration Frequencies (Hz.)

1.	0.000,	2.	0.000
3.	0.000,	4.	0.000
5.	0.000,	6.	0.000
7.	0.242,	8.	0.406
9.	0.565,	10.	0.656
11.	0.888,	12.	0.888
13.	1.438,	14.	1.536
15.	1.775,	16.	1.776
17.	3.026,	18.	3.026
19.	3.513,	20.	3.531

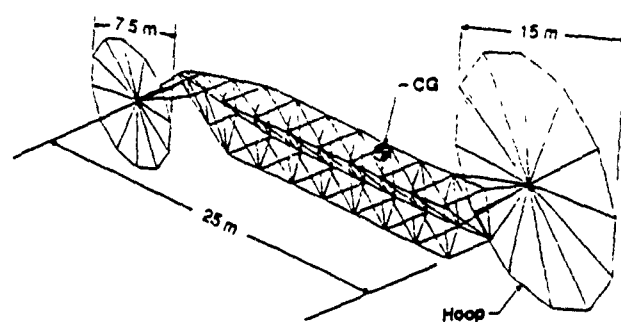


Figure 6. Earth Pointing Satellite Structure

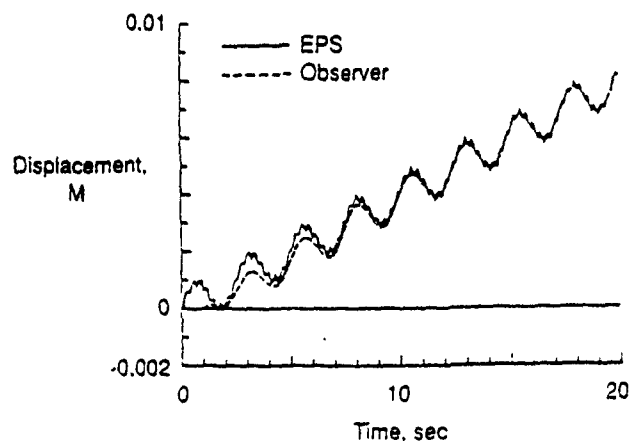


Figure 7. EPS Antenna and Second Order Observer Response

To study the viability of second order observers for this structure, sensors were located on the 15 M antenna hoop shown in Fig. 6 to measure the position and velocity of the antenna hoop. In addition, sensors were located at the center of gravity (CG) to measure the spacecraft attitude and attitude rate. The number of measurements used in the construction of Kalman Filter gains varied from 10 to 20 to create three different reduced order observer models. Again, the second order observer was derived by neglecting the L_1 gain which can result in an unstable observer for some problems. The observer was studied independent of a feedback control law by simply applying an impulsive loading to the open loop structure. An impulse was applied as an initial angular velocity about the z-axis of 100.0 rad/sec between the 15 M antenna and the truss. Note that this excitation will excite both rigid body and flexible vibrations of the EPS.

Figure 7 shows the structure and observer displacement response for 15 modes in the observer model, and a full 570 degree of freedom model of the EPS. Although the observer is stable, considerable error exists at the higher frequencies. Figure 8 shows the error between the EPS states and the observer states as the number of modes in the observer model was increased from 10 to 20. These results show the importance of larger observers in state estimation of realistic spacecraft.

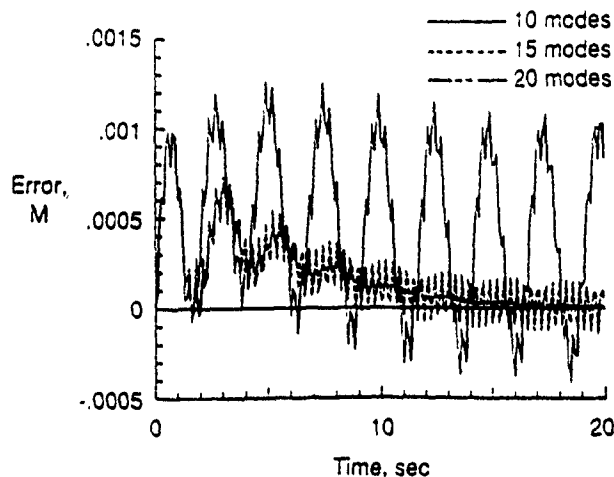


Figure 8. Reconstruction Error Convergence for EPS Observers

6. Concluding Remarks

Classical multivariable control observer gain synthesis algorithms produce first order observer models. Results have been presented which show that approximating the first order observers by a second order observer can be successfully carried out if velocity measurements are available.

This paper, in conjunction with previous works by the authors, has shown that second order observer models are computationally advantageous. The increased computational speed can be translated to larger observers and/or faster sampling rates. Larger observers allow the designer to control more 'modes' of the structure which is increasingly important for future spacecraft missions. On the other hand, if the designer chooses, he may translate the increased computational speed into faster measurement/control updates. This may prevent phase distortion in some applications and may actually enable observer based controllers when large bandwidths are necessary.

Advances in CSI technology will occur by exploiting the physical attributes of structures and structural models. This paper should serve to stimulate interest in the use of second order observers for state estimation of structures. Second order observers may prove sub-optimal as compared to the first order observer, nevertheless, practical considerations such as computational speed may be the decisive factor in developing CSI technology. More research is needed to determine if full state feedback is the control law of preference for controlled spacecraft. If the answer is affirmative, the second order observer promises to play an important role. However, much research is still needed in the areas of directly synthesizing second order observer gains and on determining robustness of the second order observer.

Acknowledgements

The contribution of the second author to the present work reported herein was supported by Air Force Office of Scientific Research under Grant F49620-87-C-0074. We thank Dr. Anthony K. Amos for his encouragement during the course of this study.

References

1. Laub, A. J. and Arnold, W. F., "Controllability and Observability Criteria for Multivariable Linear Second-Order Models," IEEE Transactions on Automatic Control, Vol. AC-29, No. 2, pp. 163-165, 1984.
2. Hughes, P. C. and Skelton, R. E., "Controllability and Observability for Flexible Spacecraft," Journal of Guidance and Control, Vol. 3, No. 5, pp. 452-459, 1980.
3. Bender, D. J. and Laub, A. J., "Controllability and Observability at Infinity of Multivariable linear Second-Order Models," IEEE Transactions on Automatic Control, Vol. AC-30, No. 12, pp. 1234-1237, 1985.
4. Park, K. C. and Belvin, W. K., "Partitioned Procedures for Control-Structure Interaction Analysis," Proceedings of the International Conference on Computational Engineering Science, Vol. 2, paper no. 64.iii, 1988.
5. Park, K. C. and Belvin, W. K., "Stability and Implementation of Partitioned CSI Solution Procedures," Proc. of the 30th Structures, Dynamics and Materials Conference, AIAA Paper no. 89-1238, April 3-5, 1989.
6. Safonov, M. G. and Athans, M., "Gain and Phase Margins for Multiloop LQG Regulators," IEEE Transactions on Automatic Control, Vol. AC-22, no. 2, April 1977.
7. Joshi, S. M. and Armstrong, E. S., "Robust Multivariable Controller Design for Flexible Spacecraft," NASA/DOD Control/Structures Interaction Technology, NASA CP-2447, 1986, pp. 547-562.
8. Balas, M. J., "Trends in Large Space Structure Control Theory: Fondest Hopes, Wildest Dreams," IEEE Transactions on Automatic Control, Vol. AC-27, no. 1, Feb. 1981.
9. Hashemipour, H. R. and Laub, A. J., "Kalman Filtering for Second-Order Models," Journal of Guidance, Control and Dynamics, Vol. 11, No. 2, pp. 181-186, 1988.
10. Oshman, Y., Inman, D. J., and Laub, A. J., "Square Root State Estimation for Second-Order Large Space Structures Models," To appear in the Journal of Guidance, Control and Dynamics, 1988.
11. Kwakernaak, H. and Sivan, R., "Linear Optimal Control Systems," Wiley-Interscience, New York, 1972.

Stability and Implementation of Partitioned CSI Solution Procedures

K. C. Park¹

Center for Space Structures and Controls
University of Colorado, Campus Box 429
Boulder, Colorado 80309

and

W. Keith Belvin²

Structural Dynamics Branch
NASA/Langley Research Center
Hampton, VA. 23665

ABSTRACT

This paper addresses computational stability and implementation aspects of a partitioned computational procedure that has been developed for control-structure interaction analysis. The procedure enables the analyst to conduct the interaction analysis by utilizing two modular single-field analysis packages: an active control synthesis package and a structural dynamics analysis package. The paper first addresses implementation issues, viz., several ways of enhancing both accuracy and the computational stability margin, modular programming of the procedure, and the selection of time-stepping formulas. In particular, the present analysis and implementation considerations suggest using the implicit mid-point formula for integrating the structures and the control system generalized forces. The stability analysis indicates that the maximum stable step size is not bound by the frequencies of the structure, but rather by the position feedback strength of the control forces.

1. Introduction

Realistic analysis of active control-structure interaction (CSI) problems pose a formidable challenge both to the dynamist and the control scientist. The emerging need for real-time simulation and control of CSI systems will engender a new activity called "computational controls" to meet this challenge. The prevailing practice in CSI simulations is by and large limited to a reduced order model of structures based on modal representations together with a limited number of sensors and actuators (see, e.g., Likins, 1970; Balas, 1978; Roberson, 1979; Skelton, 1982; Meirovitch and Silverberg, 1983; Horta, Juang, Junkins, 1985). Specifically, the control laws are expressed in terms of the generalized coordinates and their time derivatives which are computed using state estimators if full state feedback control is used. The resulting closed-loop system equations are usually cast in first-order form; this has been the prevailing

practice since modern control theory has been almost exclusively developed for the first-order form (e.g., Kalman and Bucy, 1961; Kwakernaak and Sivan, 1972; Arnold and Laub, 1984).

Typically, simulation tasks for CSI problems involve several computational elements and discipline-oriented models such as structural dynamics, control law synthesis, state estimation, actuator and sensor dynamics, thermal analysis, liquid sloshing and swirling, environmental disturbances, and maneuvering thrusts and torques. Because each of these computational elements can be large, it is usually not practical to assemble these computational elements into a single set of equations of motion and perform the analysis in its totality, which will be referred to as *simultaneous solution approach*. First, the equation size of the total system can be simply too large for many existing computers. Second, the solution of the coupled interaction equations may destroy the sparsity of the attendant matrices, thus requiring excessive computations and storage space. Most important of all, any changes in the model or in the computational procedures will effect many of the required analysis software modules and hence require a painstaking software verification effort.

The computational procedure which is described in the present paper has been motivated to alleviate the aforementioned difficulties that exist in the *simultaneous solution approach*. First, software development of any new capability is costly and time-consuming; thus, if at all possible, it is preferable to utilize existing single-field analysis modules to conduct the coupled-field interaction analysis. Second, the tasks for model generation and methods development of each field are best accomplished by relying on the experts of each single-field discipline. In order to accommodate both the software considerations and the single-field expertise, a *partitioned (or divide-and-conquer) analysis procedure* is proposed for control-structure interaction analysis (Park, 1988; Park and Belvin, 1988). The procedure abandons the conventional way of treating the CSI problems as one entity. Instead, it treats the structure (or plant), the observer, and the controller/observer interaction terms as separate entities. Thus, the CSI problem is recognized as a coupled-field problem and a divide-and-conquer strategy is adopted for the development of a real-time computational

¹ — Professor of Aerospace Engineering, University of Colorado. Member AIAA.

² — Structural Dynamics Division, NASA Langley Research Center. Member AIAA.

procedure. A similar concept has been successfully applied to other interaction analyses (Park, Felippa and DeRuntz, 1977; Park and Felippa, 1983; Zienkiewicz, 1984).

The proposed *partitioned analysis procedure* hinges on three software and computational aspects. First, because large-scale simulation of interdisciplinary problems must rely on an efficient and versatile data management system, the data manager is used to handle the necessary interprocessor communications among the single-discipline analysis modules. Second, at each discrete time increment, the equations of motion for each discipline are solved separately by considering the interaction terms as external disturbances or applied forces. Third, when necessary, computational stabilization and accuracy improvements are introduced through augmentations and/or equation modifications. It is important to note that such partitioned solutions of each discipline equations can be carried out either on a sequential or parallel machine if certain message passing and memory-conflict issues are handled appropriately.

Recently, the computational need and the physical insight that can be accrued from the second-order dynamic equations have motivated several investigators in control community to address the second-order state estimation and control law issues. They include the analysis of the second-order system equations by Hughes and Skelton (1980), Hashemipour and Laub (1987), and CSI design by Oz and Meirovitch (1982), Youssuf and Skelton (1984), Hale et al (1985), Hafka et al (1985), Junkins and Rew (1988), Khot et al. (1987), and Belvin and Park (1988, 1989), among others.

The objective of the paper is thus to describe the algorithmic nature of the partitioned CSI solution procedure, its implementation aspects and computational stability and accuracy characteristics. To this end, the paper is organized as follows. The discrete equations of motion for CSI problems are presented in Section 2, which includes the discrete equations of motion for structures subjected to control forces.

Section 3 introduces an equation augmentation technique to derive a differential equation for the interaction terms (i.e. the control law and the observer state correction force), so that, the interaction terms are obtained by solving differential equations rather than by back substitutions. It turns out that such an augmentation improves the numerical accuracy of the coupled CSI problem. Time discretization of the coupled CSI equations is carried out in Section 4. It is shown that a general canonical form of the equations of motion for structures and the state estimators leads to a computationally attractive discretization. A preliminary computational stability analysis of the present partitioned CSI solution procedure is covered in Section 5 and some example CSI analysis results are offered in Section 6. Finally, concluding remarks and further remaining challenges toward making the real-time CSI analysis and simulations a routine practice are discussed in Section 7.

2. Motivations for Partitioned CSI Simulation Procedures

A typical control-structure interaction system can be represented as shown in Fig. 1 (e.g., see Kwakernaak and Sivan, 1972). The discrete equations of motion for control-structure interaction systems may be described by

$$\begin{cases} \text{Structure:} & M\ddot{q} + D\dot{q} + Kq = f - Bu + Gw \\ \text{Sensor Output:} & z = Hx + v \\ \text{Estimator:} & \dot{\tilde{x}} = A\tilde{x} + Ef - \tilde{B}u + L(z - H\tilde{x}) \\ \text{Control Force:} & u = F\tilde{x} \end{cases} \quad (2.1)$$

where

$$x = \begin{Bmatrix} q \\ \dot{q} \end{Bmatrix}, \quad \tilde{x} = \begin{Bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{Bmatrix}$$

and

$$H = [H_d \ H_v], \quad L = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix}, \quad E = \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{bmatrix},$$

$$\tilde{B} = \begin{bmatrix} 0 \\ M^{-1}B \end{bmatrix}, \quad F = [F_1 \ F_2]$$

In the preceding equations, M is the mass matrix, D is the damping matrix, K is the stiffness matrix, $f(t)$ is the applied force, B is the actuator location matrix, G is the disturbance location matrix, q is the generalized displacement vector, w is a disturbance vector and the superscript dot denotes time differentiation. In (2.1b), z is the measured sensor output. The matrix H_d is the matrix of displacement sensor locations and H_v is the matrix of velocity sensor locations. The vector v is measurement noise. The state estimator in (2.1c) is assumed either to be based on the Kalman filter (Kalman, 1961; Kalman and Bucy, 1961) or based on a Luenberger observer (1964) if the system is deterministic. The superscript $\sim$ denotes the estimated states. The observer is governed by L , the filter gain ma-

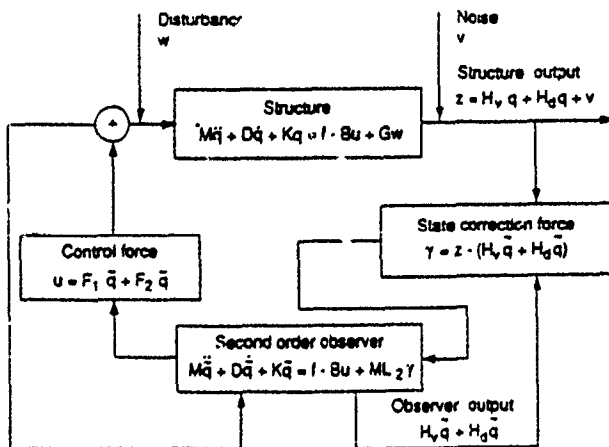


Figure 1. Typical Control/Structure Interaction System

trix. For the special case where L_1 is the null matrix, a second order state estimator can be expressed as

$$M\ddot{\bar{q}} + D\dot{\bar{q}} + K\bar{q} = f - Bu + ML_2\gamma \quad (2.2)$$

where

$$\gamma = z - (H_d\bar{q} + H_v\dot{\bar{q}})$$

This simplification is possible because for the second order observer, $\ddot{\bar{q}} = \ddot{q}$. See Belvin and Park (1989) for a more detailed description of second order observers. The actuator output, u , is a function of the state estimator variables, $\bar{q}$ and $\dot{\bar{q}}$, and F_1 and F_2 are control gains determined for example by pole-zero placement or from the solution of an optimal control problem.

The numerical solution of (2.1) begins with appropriate initial conditions, the feedback gain F and the filter gain L . Although the solution of the Kalman Filter Riccati equation to obtain the observer gains L (Hashemipour and Laub, 1988) remains a challenge in the CSI simulation endeavor, this task is deferred for a latter exposition. Hence, it is assumed steady state filter gains have been synthesized prior to the simulation.

It should be emphasized that the objective herein is to directly solve the coupled equations given in (2.1) in their natural forms. That is, to exploit the second order form of the structure equation (and also the observer equation if $L_1 = 0$). To justify this objective, the *simultaneous solution approach* for numerical simulation of (2.1) is examined. First, the structure equation is written in first order form

$$\dot{x} = Ax + Ef - \bar{B}u + \bar{G}w \quad (2.3)$$

Second, the *simultaneous solution approach* eliminates u and introduces the error equation by the deterministic form of (2.1) as:

$$\bar{e} = x - \bar{x} = \begin{Bmatrix} q - \bar{q} \\ \dot{q} - \dot{\bar{q}} \end{Bmatrix} = \begin{Bmatrix} \bar{e}_1 \\ \bar{e}_2 \end{Bmatrix} \quad (2.4)$$

so that

$$\begin{Bmatrix} \dot{x} \\ \dot{\bar{e}} \end{Bmatrix} = \begin{bmatrix} A - \bar{B}F & \bar{B}F \\ 0 & A - L\bar{H} \end{bmatrix} \begin{Bmatrix} x \\ \bar{e} \end{Bmatrix} + \begin{bmatrix} E \\ 0 \end{bmatrix} f \quad (2.5)$$

with appropriate initial conditions.

Of course, an equivalent second-order form for (2.5a) can be expressed from (2.1) as:

$$M\ddot{\bar{q}} + (D + BF_2)\dot{\bar{q}} + (K + BF_1)\bar{q} = f + B(F_1\bar{e}_1 + F_2\bar{e}_2) \quad (2.6)$$

and if $L_1 = 0$, then (2.5b) is

$$M\ddot{\bar{e}} + (D + ML_2H_v)\dot{\bar{e}} + (K + ML_2H_d)\bar{e} = 0 \quad (2.7)$$

where

$$e = \bar{e}_1 = q - \bar{q}$$

$$\dot{e} = \bar{e}_2 = \dot{q} - \dot{\bar{q}}$$

It is noted that the matrices BF_2 and BF_1 in the close-loop model equation (2.6) are in general dense and not symmetric. In addition, the matrices, L_2H_v and L_2H_d , in the error equation (2.7) are also in general dense and not symmetric. Thus, the embedding effects of both the controller and the state observer are seen to destroy the symmetry and sparsity of the open-loop system matrices, M, D and K and considerable software modifications of existing structural dynamics analysis programs would be required for large-scale CSI simulation purposes. Thus, a partitioned solution of the CSI equations is proposed to maintain the symmetry and sparsity of M, D and K which enhances the computational efficiency of the simulation.

3. Partitioned Solution of Control Laws

In order to mitigate the software and algorithmic difficulties associated with the asymmetric embedding of the controller and the state observer into the close-loop equations, a partitioned solution procedure (Park and Belvin, 1988) has been proposed for the control structure interaction equations. The exposition that follows addresses the computationally advantageous second order observers. However for completeness, the first order observer solution procedure is included in the Appendix.

First, instead of eliminating u from (2.1a) and (2.1b) via (2.1c), the control law is differentiated with respect to time to yield

$$\dot{u} = F_1\dot{\bar{q}} + F_2\ddot{\bar{q}} \quad (3.1)$$

Substituting $\ddot{\bar{q}}$ from (2.2) into the above equation, one obtains

$$\dot{u} + F_2M^{-1}Bu = F_2(M^{-1}\dot{\bar{p}} + L_2\gamma) + F_1\dot{\bar{q}} \quad (3.2)$$

where the generalized rate of momentum $\dot{\bar{p}}$ is given by

$$\dot{\bar{p}} = (f - D\dot{\bar{q}} - K\bar{q})$$

Although $F_2M^{-1}B$ is in general a full matrix, its size is relatively small as the size of u is proportional to the number of actuators placed on the structure. As a result of the equation augmentation for the controller (3.2), one can avoid the solution of the asymmetric equation, (2.6), by keeping the control force on the right-hand-side (RHS) of the equation via prediction with (3.2).

Similarly for the observer state dependent forcing function γ one derives

$$\dot{\gamma} + H_vL_2\gamma = \dot{z} - H_vM^{-1}(\dot{\bar{p}} - Bu) - H_d\dot{\bar{q}} \quad (3.3)$$

Again the matrix H_vL_2 is in general not sparse, however its size is only proportional to the number of measurements output from the structure.

Augmenting (2.2) by (3.2) and (3.3) the following set of differential equations for the observer state, control law and state correction are obtained

$$\begin{bmatrix} \dot{\mathbf{M}} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{q}} \\ \dot{\mathbf{u}} \\ \dot{\gamma} \end{Bmatrix} + \begin{bmatrix} \mathbf{D} & 0 & 0 \\ \mathbf{F}_2 \mathbf{M}^{-1} \mathbf{D} - \mathbf{F}_1 & \mathbf{I} & 0 \\ \mathbf{H}_d - \mathbf{H}_v \mathbf{M}^{-1} \mathbf{D} & 0 & \mathbf{I} \end{bmatrix} \begin{Bmatrix} \mathbf{q} \\ \mathbf{u} \\ \gamma \end{Bmatrix} + \begin{bmatrix} \mathbf{K} & \mathbf{B} & -\mathbf{M} \mathbf{L}_2 \\ \mathbf{F}_2 \mathbf{M}^{-1} \mathbf{K} & \mathbf{F}_2 \mathbf{M}^{-1} \mathbf{B} & -\mathbf{F}_2 \mathbf{L}_2 \\ -\mathbf{H}_v \mathbf{M}^{-1} \mathbf{K} & -\mathbf{H}_v \mathbf{M}^{-1} \mathbf{B} & \mathbf{H}_v \mathbf{L}_2 \end{bmatrix} \begin{Bmatrix} \mathbf{q} \\ \mathbf{u} \\ \gamma \end{Bmatrix} = \begin{Bmatrix} \mathbf{f} \\ \mathbf{F}_2 \mathbf{M}^{-1} \mathbf{f} \\ \mathbf{z} - \mathbf{H}_v \mathbf{M}^{-1} \mathbf{f} \end{Bmatrix} \quad (3.4)$$

Observe that the solution of (2.1a) and (3.4) can be carried out by a judicious employment of three software modules: the structural analyzer to obtain $\mathbf{q}$, the state estimator to obtain $\tilde{\mathbf{q}}$, and the solver for the interaction terms, $\mathbf{u}$, and γ .

To illustrate how one may proceed to solve for the structure (2.1a), the second order observer (2.2) (recall $\dot{\mathbf{q}} = \dot{\tilde{\mathbf{q}}}$), and the interaction terms (3.2) and (3.3), by the present partitioned procedure, the following solution steps are offered:

Predict $\tilde{\mathbf{q}}$: $\tilde{\mathbf{q}}_p^{n+1/2} = \tilde{\mathbf{q}}^n + \delta \dot{\tilde{\mathbf{q}}}^n$ (3.5)

Solve for $\mathbf{u}, \gamma$ Use the lower 2 equations of (3.4):
 $\mathbf{u}^{n+1/2}, \gamma^{n+1/2}$ (3.6)

Solve for $\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}$: $\mathbf{M} \ddot{\tilde{\mathbf{q}}}^{n+1/2} + \mathbf{D} \dot{\tilde{\mathbf{q}}}^{n+1/2} + \mathbf{K} \tilde{\mathbf{q}}^{n+1/2} = \mathbf{f}^{n+1/2} - \mathbf{B} \mathbf{u}^{n+1/2} + \mathbf{M} \mathbf{L}_2 \gamma^{n+1/2}$ (3.7)

Solve for $\mathbf{q}, \dot{\mathbf{q}}$: $\mathbf{M} \ddot{\mathbf{q}}^{n+1/2} + \mathbf{D} \dot{\mathbf{q}}^{n+1/2} + \mathbf{K} \mathbf{q}^{n+1/2} = \mathbf{f}^{n+1/2} - \mathbf{B} \mathbf{u}^{n+1/2} + \mathbf{G} \mathbf{w}^{n+1/2}$ (3.8)

The more general, and less computationally efficient, first order observer, (2.1c), is treated in the Appendix.

The solution of $\mathbf{q}, \tilde{\mathbf{q}}, \dot{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, \mathbf{u}$, and γ by (3.5) - (3.8) is usually carried out by direct time integration which is addressed in the next section.

4. Time Discretization of CSI Equations

Direct time integration of (3.5) - (3.8) can be carried out either by an implicit or explicit formula. Because of the small step size restriction imposed on any explicit integration formula, implicit integration formulas will be employed. (It should be mentioned that, when massively parallel computations become widely available, the programming simplicity of explicit formulas become so attractive that one may prefer explicit to implicit formulas for CSI simulations. This is particularly true when reduced order models are used which filter high frequency responses.) We express a set of the mid-point implicit formulas with the step size h :

$$\begin{cases} \mathbf{q}^{n+1/2} = \mathbf{q}^n + \delta \dot{\mathbf{q}}^{n+1/2}, & \delta = h/2 \\ \dot{\mathbf{q}}^{n+1/2} = \dot{\mathbf{q}}^n + \delta \ddot{\mathbf{q}}^{n+1/2} \\ \mathbf{q}^{n+1} = 2\mathbf{q}^{n+1/2} - \mathbf{q}^n \end{cases} \quad (4.1)$$

The selection of the mid-point implicit formula (or the

trapezoidal rule) for controlled structures is due to its minimal frequency distortion and no numerical damping characteristics (Dahlquist, 1963; Newmark, 1959; Park, 1988). It is very important to minimize numerical damping when studying control structure interaction.

Implicit time discretization of (3.8) by (4.1) yields the following difference equation for the structure:

$$\begin{cases} \mathbf{S} \mathbf{q}^{n+1/2} = \mathbf{g}^{n+1/2} \\ \mathbf{S} = \mathbf{M} + \delta \mathbf{D} + \delta^2 \mathbf{K} \\ \mathbf{g}^{n+1/2} = \delta^2 (\mathbf{f}^{n+1/2} - \mathbf{B} \mathbf{u}^{n+1/2}) + \mathbf{M} (\mathbf{q}^n + \delta \dot{\mathbf{q}}^n) \\ \quad + \delta \mathbf{D} \mathbf{q}^n + \delta^2 \mathbf{G} \mathbf{w}^{n+1/2} \\ \mathbf{q}^{n+1} = 2\mathbf{q}^{n+1/2} - \mathbf{q}^n, \quad \dot{\mathbf{q}}^{n+1/2} = (\mathbf{q}^{n+1/2} - \mathbf{q}^n) / \delta \end{cases} \quad (4.2)$$

From (4.2) it is seen that only $\mathbf{u}^{n+1/2}$ is required to numerically solve for the states of the structure. The procedure herein, is to predict $\tilde{\mathbf{q}}^{n+1/2}$

$$\tilde{\mathbf{q}}_p^{n+1/2} = \tilde{\mathbf{q}}^n + \delta \dot{\tilde{\mathbf{q}}}^n$$

and use $\mathbf{z}^{n+1/2}$ to solve the lower two equations of (3.4) for $\mathbf{u}^{n+1/2}$ and $\gamma^{n+1/2}$. (Note that $\mathbf{z}^{n+1/2}$ implies measuring the system output at the 1/2 time step. If this is not possible, using $\mathbf{z}_p^{n+1/2} = \mathbf{z}^n$ will produce a phase shift in the observer state of δ in time.)

Subsequently, (3.7) is solved for the observer state via

$$\begin{cases} \mathbf{S} \tilde{\mathbf{q}}^{n+1/2} = \tilde{\mathbf{g}}^{n+1/2} \\ \mathbf{S} = \mathbf{M} + \delta \mathbf{D} + \delta^2 \mathbf{K} \\ \tilde{\mathbf{g}}^{n+1/2} = \delta^2 (\mathbf{f}^{n+1/2} - \mathbf{B} \mathbf{u}^{n+1/2} + \mathbf{M} \mathbf{L}_2 \gamma^{n+1/2}) \\ \quad + \mathbf{M} (\tilde{\mathbf{q}}^n + \delta \dot{\tilde{\mathbf{q}}}^n) + \delta \mathbf{D} \tilde{\mathbf{q}}^n \\ \tilde{\mathbf{q}}^{n+1} = 2\tilde{\mathbf{q}}^{n+1/2} - \tilde{\mathbf{q}}^n, \quad \dot{\tilde{\mathbf{q}}}^{n+1/2} = (\tilde{\mathbf{q}}^{n+1/2} - \tilde{\mathbf{q}}^n) / \delta \end{cases} \quad (4.3)$$

Finally, equation of (4.2) is solved to compute the structural response at each time step.

To predict the interaction terms, $\mathbf{u}$ and γ , the lower two equations of (3.4) are solved using implicit integration as

$$\hat{\mathbf{S}} \mathbf{r}^{n+1/2} = \hat{\mathbf{g}}^{n+1/2} \quad (4.4)$$

where

$$\hat{\mathbf{S}} = \begin{bmatrix} \mathbf{I} + \delta \mathbf{F}_2 \mathbf{M}^{-1} \mathbf{B} & -\delta \mathbf{F}_2 \mathbf{L}_2 \\ -\delta \mathbf{H}_v \mathbf{M}^{-1} \mathbf{B} & \mathbf{I} + \delta \mathbf{H}_v \mathbf{L}_2 \end{bmatrix}$$

$$\hat{\mathbf{g}}^{n+1/2} = \begin{Bmatrix} \delta \mathbf{F}_2 \mathbf{M}^{-1} \\ -\delta \mathbf{H}_v \mathbf{M}^{-1} \end{Bmatrix} \mathbf{r}^{n+1/2} +$$

$$\begin{Bmatrix} \mathbf{F}_1 - \mathbf{F}_2 \mathbf{M}^{-1} \mathbf{D} - \delta \mathbf{F}_2 \mathbf{M}^{-1} \mathbf{K} \\ \delta \mathbf{H}_v \mathbf{M}^{-1} \mathbf{K} + \mathbf{H}_v \mathbf{M}^{-1} \mathbf{D} - \mathbf{H}_d \end{Bmatrix} \tilde{\mathbf{q}}_p^{n+1/2} +$$

$$\begin{Bmatrix} \mathbf{F}_2 \mathbf{M}^{-1} \mathbf{D} \\ \mathbf{H}_v \mathbf{M}^{-1} \mathbf{D} \end{Bmatrix} \tilde{\mathbf{q}}^n + \begin{Bmatrix} \mathbf{F}_2 \\ -\mathbf{H}_v \end{Bmatrix} \dot{\tilde{\mathbf{q}}}^n + \begin{Bmatrix} 0 \\ \delta \mathbf{z}^{n+1/2} \end{Bmatrix}$$

$$\mathbf{r}^{n+1/2} = \begin{Bmatrix} \mathbf{u}^{n+1/2} \\ \gamma^{n+1/2} \end{Bmatrix}$$

Solution of (4.4) at the half time step permits solution of (4.3) with the control and state correction forces on the right-hand-side of the equation. Since a prediction step is involved, one typically iterates to achieve a converged solution for $\tilde{q}^{n+1/2}$. However, as shown in Section 6, the augmentation procedure used herein produces very accurate results without iterating to obtain $\tilde{q}^{n+1/2} = \tilde{q}_p^{n+1/2}$ in most instances. Before demonstrating the performance of the procedure, the stability and accuracy of the method is examined in the next section.

5. Computational Stability and Accuracy of the Solution Procedure

Computational stability analysis of partitioned procedures for a general coupled system is still in an evolving stage (Park, 1980; Park and Felippa, 1983). Hence, the analysis herein applies the relevant results from Park (1980) and Park and Felippa (1983) in the present stability analysis of the partitioned CSI solution procedure. The partitioned CSI solution procedure presented in (4.1) - (4.3), although discretized by implicit time integration formulas, may suffer from computational instability as it involves extrapolations to obtain $u^{n+1/2}$ and $\gamma^{n+1/2}$. A complete stability analysis of the partitioned solution procedure for the coupled structural dynamics, observer and controller equations is difficult to perform unless the observer characteristics H, L and the controller characteristics B, F are specified. Hence, the analysis that follows is restricted to an ideal observer, i.e., $\gamma = 0$, so that only (4.1), (4.2) and the first row of (4.4) are considered. Furthermore, a restricted form of the controller is assumed (Belvin and Park, 1988):

$$u = \eta Kq + \zeta T^{-T} [T^T K T]^{1/2} T^{-1} \dot{q} \quad \text{with } B = I \quad (5.1)$$

where η and ζ are positive scalar coefficients that signify the strength of the velocity and position feedback, respectively, and T is a modal matrix defined by

$$q = T p, \quad T^T M T = I, \\ T^T K T = \Lambda = \text{diag} \{ \omega_1^2, \omega_2^2, \dots \} \quad (5.2)$$

If structural damping is neglected ($D = 0$) and every degree of freedom is controlled by (5.1), then, for each modal degree of freedom, p , and modal force, u , the following equations are obtained:

$$\begin{cases} u = \eta \omega^2 p + \zeta \omega \dot{p} \\ \ddot{p} + \omega^2 p = -u \end{cases} \quad (5.3)$$

Application of the partitioned CSI solution procedure (4.3) - (4.9) with $\gamma = 0$ to solve the above modal equation (5.3) yields

$$\begin{cases} p_p^{n+1/2} = p^n + \delta \dot{p}^n \\ (1 + \delta \zeta \omega) u_p^{n+1/2} = (\eta \omega^2 - \delta \zeta \omega^3) p_p^{n+1/2} + \zeta \omega p^n \\ (1 + \delta^2 \omega^2) p_p^{n+1/2} = -\delta^2 u_p^{n+1/2} + p^n + \delta \dot{p}^n \\ p^{n+1} = 2p_p^{n+1/2} - p^n, \quad \dot{p}^{n+1/2} = (p_p^{n+1/2} - p^n)/\delta, \\ \dot{p}^{n+1} = 2\dot{p}_p^{n+1/2} - \dot{p}^n \\ u^{n+1} = \eta \omega^2 p^{n+1} + \zeta \omega \dot{p}^{n+1} \end{cases} \quad (5.4)$$

Computational stability of the modal form of CSI partitioned equation (5.4) can be assessed by seeking a nontrivial solution of

$$\begin{Bmatrix} u^{n+1} \\ p^{n+1} \end{Bmatrix} = \lambda \begin{Bmatrix} u^n \\ p^n \end{Bmatrix} \quad (5.5)$$

such that

$$|\lambda| \leq 1 \quad (5.6)$$

for stability.

Substituting (5.5) into (5.4) and eliminating $\dot{p}$, one obtains

$$J \begin{Bmatrix} u \\ p \end{Bmatrix}^n = 0 \quad (5.7)$$

where

$$J = \begin{bmatrix} \delta(1 + \delta \zeta \omega) \cdot (\lambda + 1)^2 & 4\lambda(\delta^2 \zeta \omega^3 - \delta \eta \omega^2) + 2\zeta \omega(1 - \lambda) \\ \delta^2(\lambda + 1)^2 & (1 + \delta^2 \omega^2) \cdot (\lambda + 1)^2 - 4\lambda \end{bmatrix} \quad (5.8)$$

In order to test the stability requirement (5.6) on the characteristic equation, i.e., $\det[J] = 0$, one transforms $|\lambda| \leq 1$ into the entire left-hand plane of the z -plane:

$$\lambda = \frac{1+z}{1-z}, \quad |\lambda| \leq 1 \iff \text{Re}(z) \leq 0 \quad (5.9)$$

so that

$$J(z) = \frac{1}{(z^2 - 2z + 1)} \begin{bmatrix} 4\delta^2 \zeta \omega + 4\delta & \Sigma z^2 + \Delta z + \Theta \\ 4\delta^2 & 4z^2 + 4\delta^2 \omega^2 \end{bmatrix} \quad (5.10)$$

where

$$\begin{cases} \Sigma = 4 + \delta \eta \omega^2 - 4\delta^2 \zeta \omega^3 + 4\zeta \omega \\ \Delta = 8\delta^2 \zeta \omega^3 - 8\delta \eta \omega^2 - 4\zeta \omega \\ \Theta = 4\delta \eta \omega^2 - 4\delta^2 \zeta \omega^3 \end{cases}$$

Hence, the stability polynomial equation for (5.11) is obtained by setting $\det[J] = 0$:

$$(\delta^3 \zeta \omega^3 - \delta^2 \eta \omega^2 + 1) z^2 + (\delta \zeta \omega) z + (\delta^2 \eta + \delta^2) \omega^2 = 0 \quad (5.11)$$

A test of the polynomial equation (5.12) for possible positive real roots by the Routh-Hurwitz criterion (Gantmacher, 1959) indicates that the partitioned procedure as applied to the modal coupled equation (5.3) give a stable solution for no velocity feedback $\zeta = 0$ provided

$$h \leq \frac{2}{\sqrt{\eta \omega}} \quad (5.12)$$

If velocity feedback is present, the allowable step size for stability increases until $\zeta \geq \sqrt{\frac{4\eta^2}{27}}$ at which point the solution becomes unconditionally stable.

The stability analysis of the present partitioned CSI procedure as applied to a modal form of the coupled equations indicates that the position feedback dictates the allowable

step size for stability. Thus the highest frequency of the controller governs stability, not the highest frequency of the structure. Since most controllers are designed with reduced order structure models that ignore high frequency dynamics the present solution procedure is not unduly restricted by stability.

The step size must also be chosen based on accuracy considerations. Although the midpoint implicit integration formulas produce no artificial damping, there does exist a frequency distortion. The apparent frequency to true frequency ratio, F_a , is given by

$$F_a = \frac{N_s}{2\pi} \tan^{-1} \left(\frac{2\pi}{N_s(1 - \frac{\pi^2}{N_s^2})} \right) \quad (5.13)$$

where N_s is the number of steps per cycle, $N_s = (2\pi)/(h\omega)$. Figure 2 shows the frequency distortion as a function of the number of steps per cycle. There must be 18 steps per cycle to produce less than 1 percent frequency error. Thus, the step size based on 1 percent frequency error is

$$h \leq \frac{2\pi}{N_s \omega} = \frac{0.349}{\omega} \quad (5.14)$$

Comparing (5.12) to (5.14) one finds $\eta \geq 32.84$ before stability governs the step size and not the 1 percent allowable frequency distortion. Thus, for most practical considerations, accuracy of the present integration formulas rather than stability govern the selection of the integration step size h .

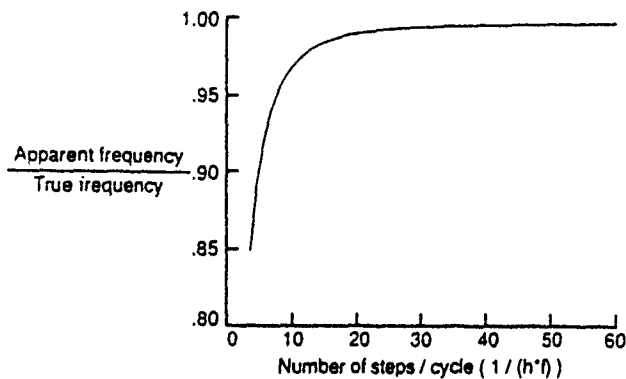


Figure 2. Frequency Distortion Using Midpoint Implicit Integration

6. Preliminary Numerical Experiments

The time-discretized CSI equations, (4.2)–(4.4), have been implemented into three modules: the structural analyzer, the state estimator and the solver for the interaction terms. The examples herein have concentrated on the structural analyzer and the control force interaction term. Additional examples with state estimation are presented in Belvin and Park (1989).

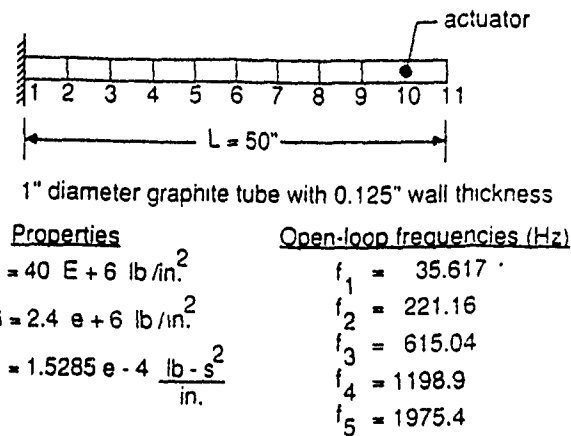


Figure 3. Cantilever Beam Example

The first numerical example is a beam with ten finite elements and an actuator placed on node 10 as shown Fig. 3. The actuator gains have been determined using a single mode in the control law. The beam is initially bent according to its first mode shape and the controller must regulate the response to one percent of its initial amplitude within 0.5 seconds. Figure 4 illustrates the tip displacement response and the control force (u) versus time. To assess the effectiveness of the present partitioned solution procedure, the response has been computed using both the conventional and the present technique for $N_s = 20$ and $N_s = 10$. The conventional technique does not use equation augmentation, (3.2), for prediction of $u^{n+1/2}$. At $N_s = 20$ both procedures yield similar results as shown in Fig. 5. However, for $N_s = 10$, the two procedures yield slightly different results. We have also computed the control force by post-processing Eq. (2.1d) instead of the augmented form (3.2) or (4.4a). Figure 6 compares the control force error computed by the present stabilized equation (4.4a) vs. that by post-processing (2.1d) for step sizes of 20 and 10 samples per response cycle. It is noted that the present partitioned solution procedure yields far better accuracy than the conventional method.

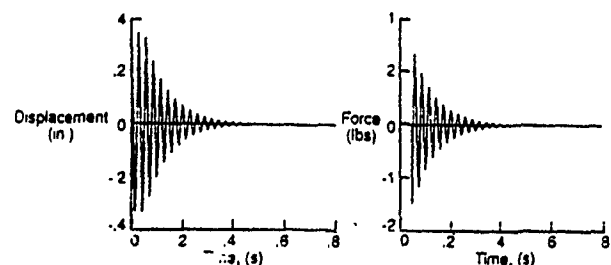


Figure 4. Beam Response

Of course, one can eliminate the solution errors associated with the conventional solution scheme. However, this requires solving the CSI equation in its entirety as given by (2.6) and (2.7), which cannot be carried out in a modular software environment.

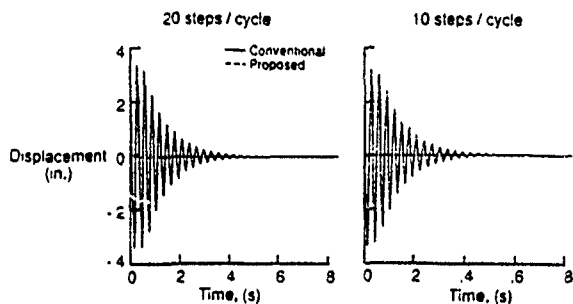


Figure 5. Effect of Integration Step Size on Beam Response

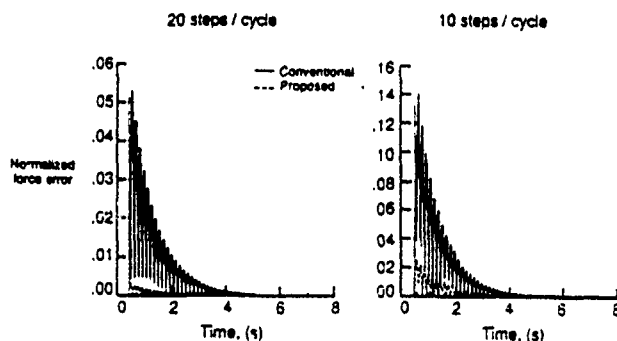


Figure 6. Beam Control Force Prediction Error

To illustrate the procedure on more realistic structures, we have performed transient response analyses on the truss-beam shown in Fig. 6. The truss was modeled by finite elements with one Timoshenko element from joint-to-joint. The model had 990 degrees of freedom. The truss is described in more detail in Belvin and Park, 1988. A modal space control law with seven modes was used to suppress vibrations of the beam induced by the loading shown in Fig. 7. Both position and rate feedback were used to reduce the vibration amplitude to 0.025 in. within 10 seconds after active control was initiated.

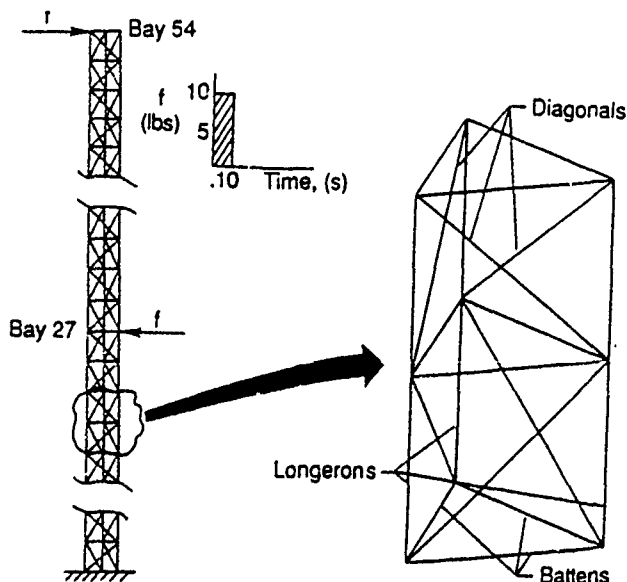


Figure 7. Truss-Beam Example

Figure 8 shows the truss response and Fig. 9 shows the control force prediction error for the proposed and conventional solution procedures. The proposed procedure is usually more than an order of magnitude more accurate than the conventional procedure because of the equation augmentation (3.2). This high accuracy removes the necessity of iterating during the prediction step.

Most importantly, the proposed technique, which exploits the second order form of the structure equation, can be used to solve this 990 degree of freedom problem without resorting to a truncated modal model. Thus simulation can be efficiently performed with "truth" models to verify performance and robustness of control laws developed from reduced order models. It is this feature that will alleviate much of the computational difficulties associated with the study of Controls/Structures Interaction.

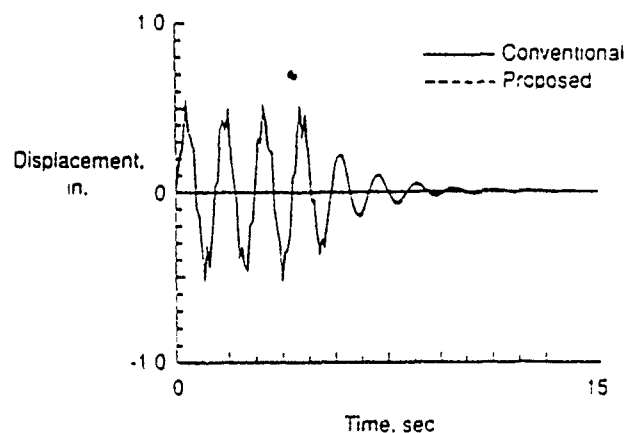


Figure 8. Truss-Beam Response

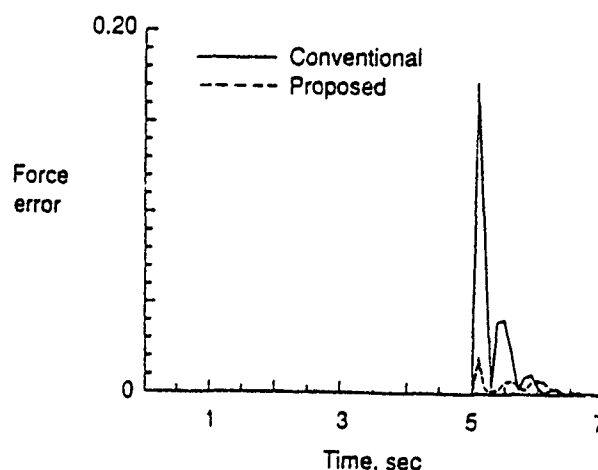


Figure 9. Truss-Beam Control Force Prediction Error

7. Summary

A partitioned procedure for an efficient solution of control-structure interaction systems has been presented. The development of the present partitioned procedure for conducting the CSI analysis has been motivated by a desire to perform such analyses by assembling several modular

software packages, rather than embedding additional interaction phenomena either into a structural analysis program or into a control-synthesis program. This *divide-and-conquer* computational strategy should prove increasingly important when additional phenomena such as large slewing motions, thermal transients, fluid-tank interaction and other environmental disturbances are to be considered.

The CSI simulations based on modular software elements has uncovered several theoretical and computational issues. First, the availability of a second order observer model is required to exploit the symmetry and sparsity for efficiency purposes. Second order observers are discussed in Belvin and Park (1989) with the application of the present procedure in mind.

Second, for time variant systems, the solution of a Riccati equation to determine the controller, u , with appropriate terminal conditions and the solution of another Riccati equation for the observer equation with given initial conditions can pose bottlenecks in real-time CSI simulations. Note that these two solutions are required to obtain the matrices F , and L . Parallel computing can potentially remove the aforementioned bottlenecks of the time variant problem.

The present partitioned CSI procedure can facilitate the use of large-scale structural analysis—at least in principle—for control-structure interaction analysis both by the structural dynamist and the control engineer. It is anticipated that real-time CSI simulations will become routine if adequate advances are made in the construction of reduced-order observers and, for time variant problems, if substantially faster parallel solution procedures for the Riccati equations become available.

Finally, we have not considered the intrinsic dynamics of both the actuator and the sensor. However, since the present procedure uses a differential form of the actuator and sensor equations they can be incorporated into the present computational procedure without too much difficulty. Further experiments and inclusions of more realistic CSI models are necessary before the present CSI simulation procedure can become a production-simulation tool. This is being carried out at present.

Acknowledgements

The contribution of the first author to the present work reported herein was supported by Air Force Office of Scientific Research under Grant F49620-87-C-0074. We thank Dr. Anthony K. Amos for his encouragement during the course of this study.

References

1. Arnold, W. F. and Laub, A. J. (1984), "Generalized eigenproblem algorithms and software for algebraic Riccati equations," *Proceedings of the IEEE*, 72, No. 12, 1746-1754.
2. Balas, M. J. (1978), "Active Control of Flexible Systems", *Journal of Optimization Theory and Applications*, 25, 415-436.
3. Belvin, W. K. and Park, K. C. (1988), "Structural Tailoring and Feedback Control Synthesis: An Interdisciplinary Approach," *Proc. the 29th Structures, Dynamics and Materials Conference*, AIAA Paper No. 88-2206, AIAA, 1-8.
4. Belvin, W. K. and Park, K. C. (1989), "On the State Estimation of Structures with Second Order Observers," *Proc. the 30th Structures, Dynamics and Materials Conference*, AIAA Paper No. 89-1241, April 3-5, 1989.
5. Gantmacher, F. R. (1959), *The Theory of matrices*, 2. Chelsea, New York, N. Y., Chapter XV.
6. Gear, C. W. (1971), *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, New Jersey.
7. Haftka, R. T., et al. (1985), "Sensitivity of Optimized Control Systems to Minor Structural Modifications," 26th Structures, Structural Dynamics and Materials Conference, Orlando, FL April 15-17.
8. Hale, A. L., Lisowski, R. J. and Dahl, W. E. (1985), "Optimal Simultaneous Structural and Control Design of Maneuvering Flexible Spacecraft," *Journal of Guidance, Control and Dynamics*, 8, 86-93.
9. Hashemipour, H. R. and Laub, A. J. (1988), "Kalman filtering for second-order models," *J. Guidance, Control and Dynamics*, 11(2), 181-185.
10. Horta, L. G., Juang, J. N. and Junkins, J. L. (1985), "A sequential Linear optimization Approach for Controller Design," AIAA Paper 85-1971-cp.
11. Hughes, P. C. and Skelton, R. E. (1980), "Controllability and observability of linear matrix second-order systems," *J. Applied Mechanics*, 47, 415-420.
12. Junkins, J. L., and Rew, D. W. (1988), "Unified Optimization of Structures and Controllers", *Large Space Structures: Dynamics and Controls*, edited by Atluri, S. N. and Amos, A. K., Springer-Verlag, 323-353.
13. Kalman, R. E. (1961), "On the General Theory of of Control Systems," *Proc. 1st International Congress on Automatic Control*, Butterworth, London, 1, 481-491.
14. Kalman, R. E. and Bucy, R. S. (1961), "New results in linear filtering and prediction theory," *Trans, ASME J. Basic Engineering*, 83, 95-108.
15. Khot, N. S., et al. (1987), "Optimal Structural Modifications to Enhance the Optimal Active Vibration Control of Large Flexible Structures," 26th Structures, Structural Dyn., and Materials Conf. Orlando, FL, April 15-17.
16. Kwakernaak, H. and Sivan, R. (1972), *Linear Optimal Control Systems*, Wiley-Interscience, New York.
17. Likins, P. W. (1970), "Dynamics and Control of Flexible Space Vehicle," NASA TR-32-1329.
18. Luenberger, D. G. (1964), "Observing the state of a linear system," *IEEE Trans. Mil. Electron*, 8, 74-80.

19. Meirovitch, L. and Silverberg, L. M. (1983), "Globally Optimal Control of Self-Adjoint Distributed Systems," *Optimal Control and Applications*, 4(4), 365-386.
20. Oz, H. and Meirovitch, L. (1980), "Optimal Modal-Space Control of Flexible Gyroscopic Systems," *Journal of Guidance and Control*, 3, 220-229.
21. Park, K. C. (1975), "An Improved Stiffly Stable Method for Direct Integration of Nonlinear Structural Dynamics Equations," *Journal of Applied Mechanics*, 42, 464-470.
22. Park, K. C., Felippa, C. A. and DeRuntz, J. A. (1977), "Stabilization of staggered solution procedures for fluid-structure interaction analysis," in: *Computational Methods for Fluid-Structure Interaction Problems*, Belytschko, T. and Geers, T. L. (editors), ASME, AMD Vol. 26, New York, N. Y., 95-124.
23. Park, K. C. (1980), "Partitioned Analysis Procedures for Coupled-Field Problems: Stability Analysis," *Journal of Applied Mechanics*, 47, 370-378.
24. Park, K. C. and Felippa, C. A. (1983), "Partitioned Analysis of Coupled Systems," in: *Computational Methods for Transient Analysis*, T. Belytschko and T. J. R. Hughes (eds.), Elsevier Pub. Co., 157-219.
25. Park, K. C. (1988), "Computational Issues in Control-Structure Interaction Analysis," *Large Space Structures: Dynamics and Control*, edited by Atluri, S. N., Springer-Verlag, 115-131.
26. Park, K. C. and Belvin, W. K. (1988), "Partitioned Procedures for Control-Structure Interaction Analysis," *Computational Mechanics '88*, S. N. Atluri and G. Yagawa (editors), Vol. 2, Springer-Verlag, 64.iii.1-4.
27. Park, K. C. and Chiou, J. C. (1988), "Stabilization of Computational Procedures for Constrained Dynamical Systems," *Journal of Guidance, Control and Dynamics*, 11, 365-370.
28. Skelton, R. (1982), *Theory and Applications in Optimal Control in Aerospace Systems*, AGARD Publication No. 251, ed. by P. Kant, Brussels, Belgium.
29. Roberson, R. E. (1979), "Two Decades of Spacecraft Attitude Control," *J. Guidance, Control and Dynamics*, 2, 3-8.
30. Yousuff, A. and Skelton, R. E., (1984), "Controller Reduction by Component Cost Analysis," *IEEE Trans. Auto. Contr.*, AC-24, 52-530.
31. Zienkiewicz, O. C. (1984), "Coupled Problems and Their Numerical Solution," *Numerical Methods in Coupled Systems*, ed. by R. W. Lewis, P. Bettess and E. Hinton, John Wiley & Sons, 35-58.

Appendix - First Order Observer Solution

The first order observer can be numerically integrated as described below. Implicit mid-point formulas are used in the time discretization. All variables are defined in (2.1).

The first order observer takes the form

$$\dot{\bar{x}} = A\bar{x} + Ef - \bar{D}u + L(z - H\bar{x}) \quad (A.1)$$

Combining like terms, we derive

$$\dot{\bar{x}} = A_o\bar{x} + Ef + Lz \quad (A.2)$$

where

$$A_o = \begin{bmatrix} -L_1H_d & I - L_1H_v \\ -M^{-1}(K - BF_1) - L_2H_d & -M^{-1}(D - BF_2) - L_2H_v \end{bmatrix}$$

Time discretization yields

$$(I - \delta A_o)\bar{x}^{n+1/2} = \delta(Ef^{n+1/2} + Lz^{n+1/2}) + \bar{x}^n \quad (A.3)$$

$$\bar{x}^{n+1} = 2\bar{x}^{n+1/2} - \bar{x}^n \quad (A.4)$$

$$u^{n+1/2} = F\bar{x}^{n+1/2} \quad (A.5)$$

Equations (3.5 - 3.7) would be replaced by (A.3) in the integration procedure. Then, (A.5) would be used to solve (3.8) for the structural states. The computational efficiency of the first order observer (A.3) is discussed in Belvin and Park (1989).

A GENERAL APPROACH TO NONLINEAR FE COMPUTATIONS ON SHARED-MEMORY MULTIPROCESSORS

Charbel FARHAT and Luis CRIVELLI

*Department of Aerospace Engineering and Center for Space Structures and Controls, University of
Colorado at Boulder, Boulder, CO 80309-0429, U.S.A.*

Received 12 March 1987

Revised manuscript received 9 June 1988

A computational strategy for nonlinear finite element computations on shared-memory multiprocessors is presented. It exploits all the parallelism inherent in the finite element method. Both iterative and direct solution methods are implemented in the same environment. Explicit computations are carried out at the element level. Implicit computations are processed at the subdomain level. An element coloring scheme is used to eliminate *critical regions* for the whole class of explicit computations. To facilitate load balancing among the processors in a posteriori nonlinear problems, a dynamic remapping of the processors on the computational tasks is introduced. Numerical experiments are conducted on Alliant FX/8 and Cray2. Very high rates of efficiency are achieved on both multiprocessors.

1. Introduction

Until recently, most increases in the speed of finite element computations have come via the speed of vector processors. Now, the true potential for execution time improvement lies in concurrent multiprocessors. "A multiprocessor is a computer with two or more central processing units, each of which executes instructions independently of the others except when a processor needs to communicate or synchronize with one or more of the others" [1]. Two basic architectures are being pursued: (1) the local-memory multiprocessors where each processor has his independent attached memory and communicates with other processors by sending messages through an interconnect scheme. Examples of such machines are Intel's iPSC and JPL/Caltech's MARK III hypercubes; (2) the shared-memory multiprocessors where the processors access a common memory via some form of interconnection mechanism. The Alliant FX/8 and Cray2 machines fall within this category.

Farhat et al. [3] have designed a computational strategy based on a direct method for solving structural mechanics linear problems on local-memory multiprocessors. Nour-Omid and Park [4] have proposed a parallel implementation of a preconditioned conjugate gradient method for the solution of linear systems of finite element equations of Caltech's hypercube.

At the time of writing this paper, and with the exception of the Connection Machine with its 65,536 processors, shared-memory multiprocessors are offering a much larger crunching number power than local-memory machines. Since it is still not clear which architecture will dominate the next generation of super-multiprocessors, our goal is to fully exploit the computational power of the fastest and currently available machines. We are particularly

interested in the *concurrent supercomputers* which consist today of four (Cray2) to eight (ETA-10) interconnected vector processors capable of working simultaneously on the same problem. Hence, even though the algorithms we present in this paper are suitable for finite element parallel computations on local-memory multiprocessors, their computer implementation is tailored to fully exploit the architectural features of the shared-memory machines.

In order to achieve the goal set forth above, a general framework for parallel nonlinear computations on shared-memory multiprocessors is proposed. This framework features two mapping techniques: (1) a mapping associated with element-by-element explicit computations and (2) a mapping associated with subdomain-by-subdomain implicit manipulations. All the computations that are carried out at the element level and all the numerical algorithms that are based on matrix-vector products such as *preconditioned conjugate gradient* and *dynamic relaxation* are handled almost asynchronously. Expensive *critical regions* are bypassed by processing the elements in a specific order. Two levels of parallelism are embedded in this framework, namely, (1) concurrency at the outer loop and (2) vectorization at the inner loop. The problem of achieving load balancing among the processors in a *posteriori* nonlinear problems is solved via the combination of the two different mappings of the processors on the computational tasks. Finally, *The Force* [5] is used to produce a portable code among this class of multiprocessors. The same code has been run without major modifications on Alliant FX/8 and Cray2.

The remainder of the paper is organized as follows. Section 2 presents a general framework for parallel element-by-element computations. This framework preserves the vectorization level of parallelism and builds on top of it a concurrency level. In Section 3 the implementation of numerical iterative algorithms within this framework is discussed. The efficient implementation of a concurrent direct method is also described in Section 4. It is used as an inner loop within a *Newton-like* method for an alternative solution of a *posteriori* nonlinear problems. For this case, two different processor-mapping schemes are combined to achieve a load balance among the processors. In all cases, at each iteration and within each computational step, fork/join procedures are replaced by *Barriers* [5], so that overhead costs are minimized. Numerical experiments are reported in Section 5 and concluding remarks are offered in Section 6.

2. A framework for parallel EBE computations

Most of the computations involved in a finite element analysis are carried out at the element level. These include, for example, the formation of element stiffnesses and the computation of field derivatives. Consequently, these operations can be, in principle, carried out in parallel without any synchronization. Practically, after the elemental calculations have been achieved, one still needs to satisfy some architectural requirements when considering the local accumulation of these computations on a specific class of multiprocessors. The solution of a system of equations constitutes another important phase of a finite element analysis. It would be very natural, especially from a parallel processing viewpoint, if the selected numerical solution algorithm could also operate at the element level. Generally, this is the case for iterative solution schemes based on matrix-vector products. It is well known that such products can be achieved without the need for assembling the matrix or the vector. However, when im-

plemented on a shared-memory multiprocessor, some of the element-by-element (EBE) computations may lead to what is termed *critical regions*, which are known to slow down the speed of parallel computations. This phenomenon is illustrated below.

In Fig. 1, a mesh node P is shown attached to four distinct elements E_1, E_2, E_3, E_4 .

Let u_P be a mathematical unknown at node P . Typically, u_P is the accumulation of independent elemental contributions $u_P^{(E)}$. A simple segment of FORTRAN code that evaluates $u_P = \sum_{E=E_1}^{E=E_4} u_P^{(E)}$ would be:

```
UP = 0
DO 100 IE = 1,4
  UE      = FUNCTION(IE,AND_OTHER_PARAMETERS)
  UP      = UP + UE
100 CONTINUE
```

For this example case, if each of the four elements is assigned to a different processor via the index IE, each processor will have its *private* copy of the temporary work UE, and the independent quantities $u_P^{(E)}$, represented here by UE, will be evaluated in parallel and stored in distinct physical locations of the memory. However, the four running processors cannot *correctly* update in parallel the value of u_P , represented here by UP, because the statement $UP = UP + UE$ is recursive. Such a portion of a code where a processor needs to store into a memory location used concurrently by another processor is usually termed *critical region*. Since each processor reads and stores in UP, explicit synchronization has to be invoked. This can be done as follows:

```
UP = 0
DO 100 IE = 1,4
  UE      = FUNCTION(IE, AND_OTHER_PARAMETERS)
  Critical Section
  UP = UP + UE
  End Critical Section
100 CONTINUE
```

The *Critical Section* construct as described in [5] guarantees that only one process at a time will execute the block of code nested between the *Critical* and *End Critical* statements. Hence,

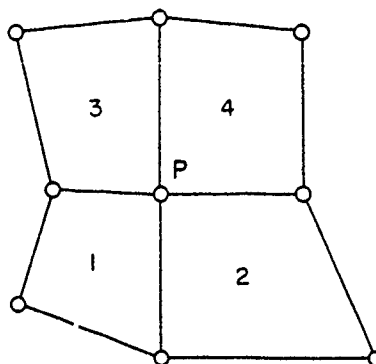


Fig. 1. A mesh node attached to four elements.

the computations are serialized at this level and each processor updates sequentially the shared value UP with its private contribution UE so that a correct value for UP is output. The critical region described above slows down the speed of computations in two different ways. First, it serializes the computations at this level. Second, it requires a synchronization construct that constitutes an overhead. Since such a critical region may occur at each node of a finite element mesh, the aggregate serializations and overheads will significantly degrade the sought-after speedup.

Critical regions in parallel EBE computations result from the fact that a node is usually common to several elements. Since element-level computations can be done in an arbitrary order, most of the critical regions can be avoided by processing the discrete elements in a sequence where no two of them are adjacent. This idea appeared first in Berger et al. [10] in their work on assembling in parallel a global stiffness matrix on a shared-memory multiprocessor. It was then reused by Hughes et al. [6] for vectorizing the evaluation of nodal residuals in EBE computations. In this paper, we extend its application to various types of asynchronous element processing and discuss its sensitivity to mesh topology as well as its impact on multiprocessing efficiency.

Consider a mesh including several types of elements (truss, beam, shell, etc.). Groups G_i of elements of a similar type are first defined. Then, each group G_i is split into n_c lists L_j of internally disjoint elements via a graph coloring scheme, where n_c is the number of colors required to separate G_i . The nonnumerical algorithm that performs the splitting should be general-purpose in the sense that it should apply to regular and irregular two- and three-dimensional meshes. Moreover, it should minimize the number of required colors n_c , which is the same as the number of unicolor lists L_j , and hence—as will be shown—it should minimize the amount of necessary synchronizations. The parallel processing of the mesh elements is done in sweeps as the discussion demonstrates. Within each group G_i , the lists of elements L_j are processed one after the other. For each list L_j , the processors are mapped asynchronously onto the scattered elements which are processed in parallel. Accumulation operations such as $UP = UP + UE$ need not be synchronized within the same list L_j since they do not constitute critical regions. This is because no two elements within L_j share a common node and hence no two processors will operate on the same UP within L_j . Synchronization is needed only between two sweeps of computations that correspond to the processing of two different lists L_j . Hence, if n_n denotes the total number of nodes within a group G_i , bypassing the critical regions reduces the amount of synchronization from n_n to n_c which is usually less than 10. Vectorization of EBE computations is carried out as explained in [6]. Within each unicolor list L_j , elements are assigned to processors by blocks of α NREG, where α is an integer and NREG is the number of available vector registers.

In summary, parallel processing is achieved at two levels. At the outer loop, blocks of internally disjoint elements are processed concurrently. At the inner loop, EBE computations within a block are fully vectorized. Synchronization occurs only between the processing of two distinct sweeps of computations. An irregular finite element mesh with 1200 elements is shown in Fig. 2(a). When applied to this mesh the coloring scheme yields four different colors. Therefore the mesh elements are reorganized for parallel computations into four distinct lists L_j of internally disjoint elements, each containing 300 of them (an integer in the figure represents a color).

Hence, if the mesh shown above were to be analyzed, hardware synchronization calls would

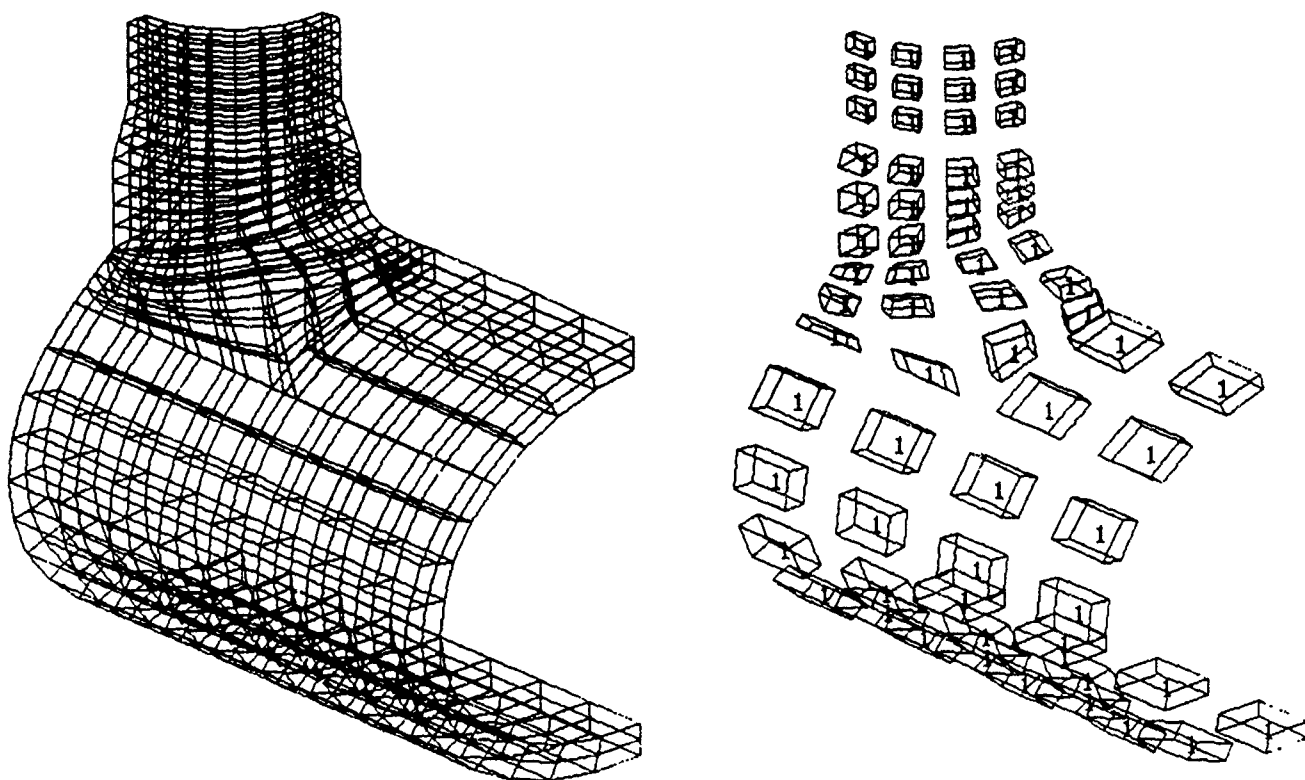


Fig. 2. (a) Coloring of an irregular finite element mesh.

invoked only four times per iteration and 300 elements would be processed asynchronously between two synchronization points. Clearly, a minimum number of colors means a minimum number of lists L_i , each containing a maximum number of internally disjoint elements, which results in a maximum ratio of parallel work done per processor/amount of synchronization overhead. However, the number of colors required to separate a mesh is generally a function of the connectivity pattern of the elements. Basically, one needs to avoid situations where a large number of elements share a common node. Fortunately this is always possible, as illustrated below. Consider the finite element mesh of Fig. 2(b). Because 36 elements in this mesh share a common node, 38 colors and therefore 38 lists L_i , some of which contain only one element, are required for parallel processing. In Fig. 2(c) the same geometrical domain is remeshed; the total number of elements is kept approximately the same but a different meshing technique is used. In this case, only 5 colors and therefore 5 lists L_i and 5 synchronization procedures are needed for parallel processing; it is therefore far more efficient. The organization of the doubly parallel EBE computations is summarized in Table 1.

Parallel iterative solution algorithms

Discrete equilibrium equations arising from finite element nonlinear formulations may be written in the general compact form

$$r(u, p, \theta) = 0, \quad (1)$$

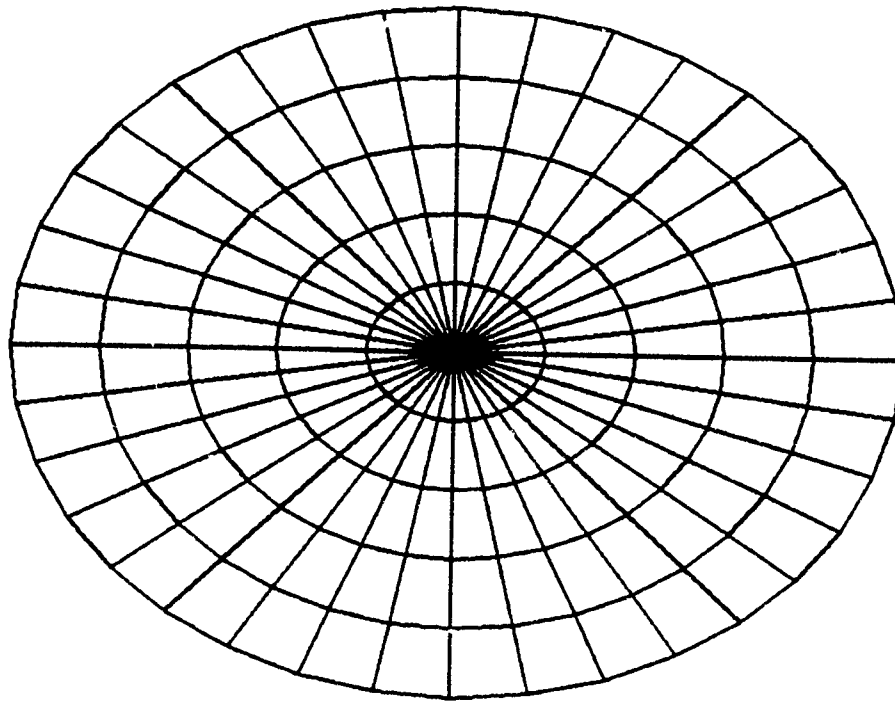


Fig. 2. (b) Inefficient meshing.

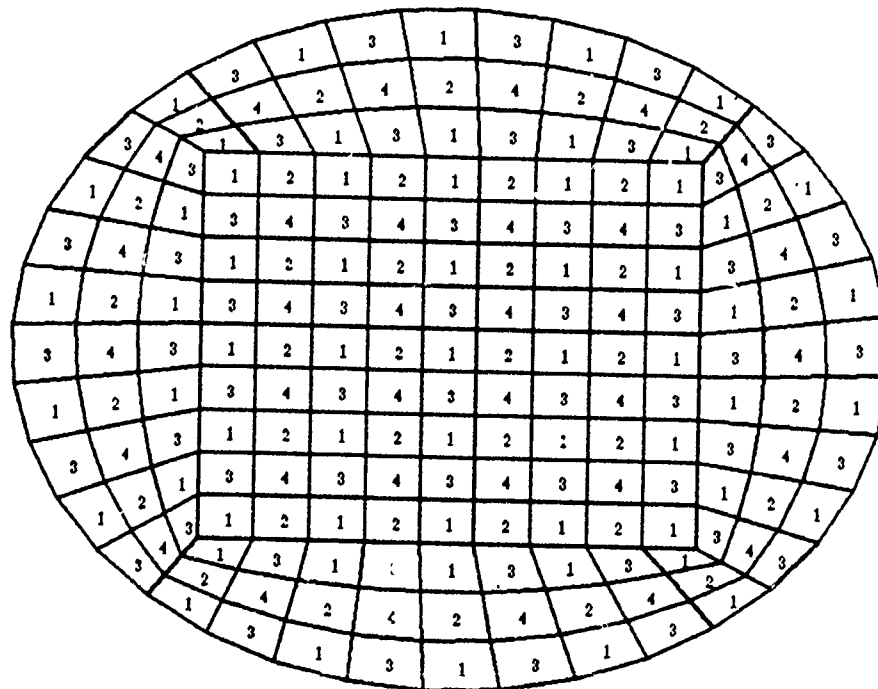
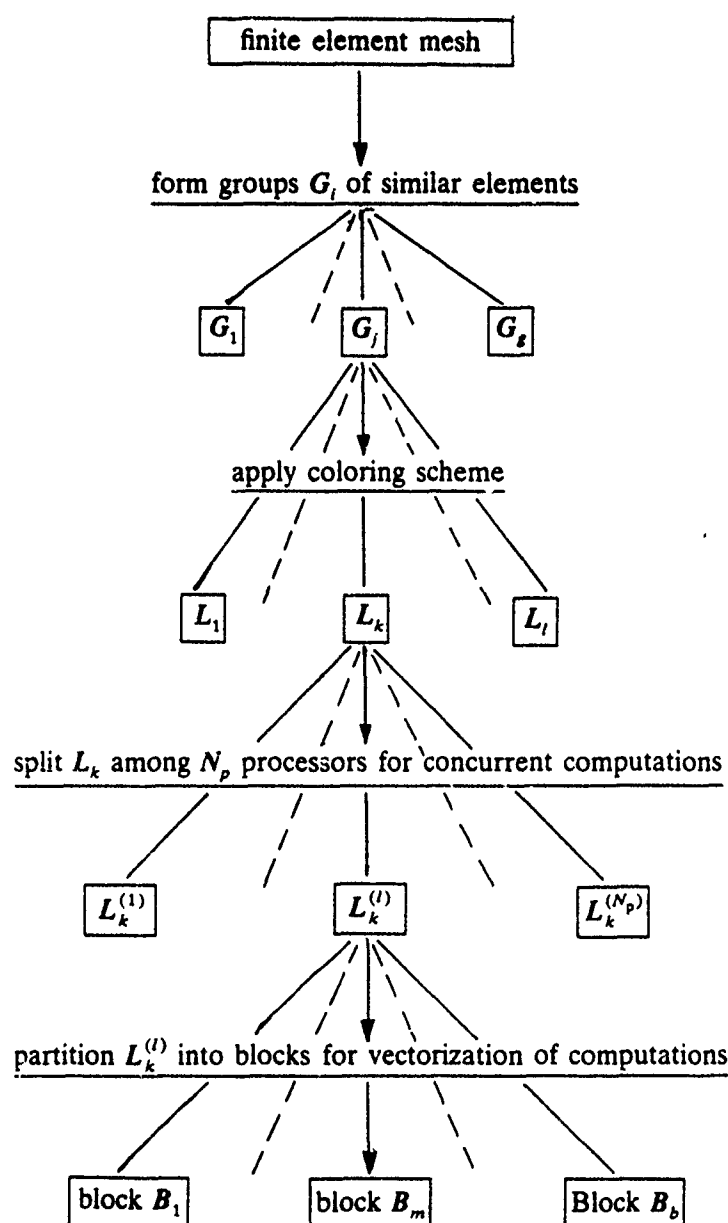


Fig. 2. (c) Efficient meshing.

where $\mathbf{u}$ denotes the unknown vector of generalized displacements (rotations, temperatures, etc.) at the nodes of the discretized geometrical domain, $\mathbf{p}$ denotes a set of control parameters, $\boldsymbol{\theta}$ is a functional of past history of the generalized deformation gradients, and $\mathbf{r}$ denotes the residual vector of out-of-balance generalized forces (moments, fluxes, etc.). Equation (1) covers all geometrical nonlinearities, material nonlinearities, and several types of boundary condition nonlinearities.

Table 1
Organization of doubly parallel computations



The Newton-Raphson method and its numerous variants, collectively known as *Newton-like* methods, are the most popular class of methods for the solution of (1) on conventional computers. These methods can embed either an iterative or a direct algorithm for the solution of a linearized system of equations. The use of iterative schemes for the solution, at each step, of the linearized system of equations has two desirable advantages: (1) it efficiently exploits the sparsity of the involved matrices and therefore requires less storage than direct schemes; (2) it provides a means of controlling the accuracy of the solution. The preconditioned conjugate gradient (PCG) has emerged over the last decade as a favorite algorithm for solving large sparse systems of linearized equations on sequential, vector [11, 12, 13], and parallel [14, 15, 16] computers. However, conventional *Newton-like* methods may behave poorly near bifurcation points and often fail to handle path-dependent problems such as plastic flow,

where the stiffness matrix may oscillate wildly as the solution changes by small amounts. For such problems, explicit dynamic relaxation (DR) is a very robust iterative computational strategy. Since one would like to select the right strategy for the right problem, we have decided to focus on the implementation within the same environment of both *Newton-like* and explicit dynamic relaxation algorithms. In this section, we consider PCG as a solver for *Newton-like* methods. A direct solver will be presented in the following section.

Newton-like methods for solving nonlinear systems of equations having the general form $r(y) = 0$ are usually related to the following iteration scheme:

For $k = 0, 1, 2, \dots$ until convergence do:

$$\text{Solve } r'(y_k)x_k = -r(y_k),$$

$$\text{Set } y_{k+1} = y_k + x_k.$$

In structural applications, $r'(y_k)$ becomes the stiffness matrix $K(u_k)$ evaluated in the displacement state u_k . The quantities x_k and $-r(y_k)$ become respectively the vector of unknown displacement increments δu_{k+1} and the vector of out-of-balance forces $\delta f(u_k)$. Given a preconditioning matrix P and an initial approximation δu_0 to the solution increment δu , a PCG algorithm generates a sequence of improved approximations δu_n to which corresponds a sequence of residuals $r_n = \delta f - K \delta u_n$. A sequential version of the algorithm specified to finite element computations is summarized in Table 2.

The stiffness K is usually a symmetric positive-definite matrix assembled from elemental symmetric positive-semidefinite matrices $K^{(e)}$. However, it is well known that since K is involved here only multiplicatively, it needs not be assembled and stored. Let $v_n^{(e)}$ denote the contribution of the e th finite element to the assembled vector $v_n = \sum_{e=1}^{e=n} v_n^{(e)}$, and let $v_n^{a(e)}$ denote the localization to the e th element of v_n after it has been assembled. It follows that the quantity $v_n^t K v_n$ can be performed by accumulating element level computations

$$v_n^t K v_n = \sum_{e=1}^{e=n} v_n^{a(e)t} K^{(e)} v_n^{(e)},$$

and an inner product of two finite element vectors can be computed element-by-element as in

Table 2
PCG algorithm

(S_0) Initialize

$$v_0 = r_0 = \delta f - K \delta u_0$$

$$n = 1, 2, \dots$$

(S_n) Iterate

If $\|r_{n-1}\| < \text{tolerance}$ terminate.

$$\text{Solve } P r_{n-1}^* = r_{n-1}$$

$$\gamma_n = r_{n-1}^t r_{n-1}^*$$

$$v_n = r_{n-1}^* + (\gamma_n / \gamma_{n-1}) v_{n-1} \quad (v_1 = r_0^*)$$

$$\alpha_n = \gamma_n / (v_n^t K v_n)$$

$$\delta u_{n+1} = \delta u_n + \alpha_n v_n$$

Compute r_n using equilibrium principle.

the example below

$$r_{n+1}^t r_{n+1}^* = \sum_{e=1}^{e=n} r_{n+1}^{a(e)T} r_{n+1}^{*(e)}.$$

It is important to note that in both cases, one of the two vectors still needs to be assembled before the inner product is performed. The combination of two vectors is trivially done element-by-element as shown below

$$\delta u_n + \alpha_n w_n = \sum_{e=1}^{e=n} \delta u_n^{(e)} + \alpha_n w_n^{(e)}.$$

Finally, if P is Hughes' preconditioner [6] or a simple Jacobi preconditioner, the entire PCG algorithm can be applied using only element-level computations. Consequently, these computations are perfectly parallelized within the framework discussed in Section 2. Note that they are also naturally vectorizable within each process.

Dynamic relaxation is a robust iterative method for solving highly nonlinear systems. Unlike the conjugate gradient method, it does not need to be embedded within a Newton outer loop. The algorithm solves the nonlinear discrete quasistatic finite element equations

$$r(u, f) = S(u) - f = 0$$

by viewing them as the steady-state solution of the second-order pseudo-dynamic problem

$$M\ddot{u} + C\dot{u} + S(u) - f = 0, \quad (2)$$

where M and C are fictitious mass and damping matrices constructed in a way that achieves computational efficiency when integrating (2) with the central difference scheme. To preserve the explicit form of the central difference integrator M must be diagonal and C is chosen for example as $C = cM$. Parameter c and stepsize δt are selected to obtain the fastest convergence:

$$\begin{aligned} \delta t &\leq 2/(\omega_{\max}^2 + \omega_{\min}^2)^{(1/2)}, \\ c &= 2\omega_{\min}/(1 + \omega_{\min}^2/\omega_{\max}^2)^{(1/2)}, \end{aligned}$$

where $\omega_{\min}$ and $\omega_{\max}$ are respectively the lower and higher pseudo-frequencies of the pseudo-dynamic problem. These quantities need not be computed exactly. Rough estimates for both pseudo-frequencies are sufficient [7]. More details on the selection of M , C , c , and δt may be found in [20]. Underwood proposed in [7] an adaptive DR scheme, where the integration parameters change from step to step. Here, we follow his work except for the adaptive lowest pseudo-frequency, $\omega_{\min}^a$, where we found that

$$\left(\frac{\delta u^t \delta r}{\delta u^t M \delta u} \right)^{1/2}$$

is a more effective approximation to $\omega_{\min}^a$. The resulting algorithm is displayed in Table 3.

Table 3
DR algorithm

(S ₀)	Initialize	$u_0 = \dot{u}_0 = 0$ $r_0 = f$ $\alpha_1 = 1$ $\beta_1 = 1$
(S _n)	Iterate	$n = 1, 2, \dots$ Compute r_n using equilibrium principle. If $\ r_n\ _2 < \text{tolerance}$ terminate. $\dot{u}_n = \beta_n \dot{u}_{n-1} + \alpha_n \delta t M^{-1} r_{n-1}$ $u_n = u_{n-1} + \delta t \dot{u}_n$ $\omega_n^a = \left(\frac{\delta u^T \delta r}{\delta u^T M \delta u} \right)^{1/2}$ $\alpha_{n+1} = 1 / (1 + \delta t \omega_n^a)$ $\beta_{n+1} = (1 - \delta t \omega_n^a) / (1 + \delta t \omega_n^a)$

Clearly, the computations involved in the dynamic relaxation method are similar to those of the conjugate gradient method. Hence, they are performed in parallel element-by-element as discussed previously. Note also that these computations are perfectly vectorizable within each process.

4. Concurrent direct method

Direct solution techniques have been popular among engineers, mainly because of two advantages they possess over iterative schemes: (1) they are robust for ill-conditioned systems which often arise in the analysis of flexible space structures, and (2) their execution time can be estimated for any given problem. They can be sensitive to round-off error (matrices with high condition number), but their main disadvantage is that they suffer from having excessive storage requirements for large matrices, so that an out-of-core solution is often required. However, they are still attractive for supercomputers such as Cray2, where the main memory can store up to 256 million double-precision words. In this section, we are concerned with the design of a concurrent direction method and its implementation on a super-multiprocessor. The resulting algorithm is to be used as the inner loop of *Newton-like* methods for the solution at each step k of the linearized system of equations

$$K(u_k) \delta u_{k+1} = \delta f(u_k).$$

Also, we introduce a dynamic remapping of the processors on the computational tasks that achieves a load balance in a posteriori nonlinear problems (for example problems with material nonlinearities, where the number and location of elements that may yield at a loading step are not known in advance).

In Fig. 3, the previous irregular finite element domain D is shown subdivided into a system of N_p subdomains D_j , where N_p is the number of available processors (here N_p was set to four).

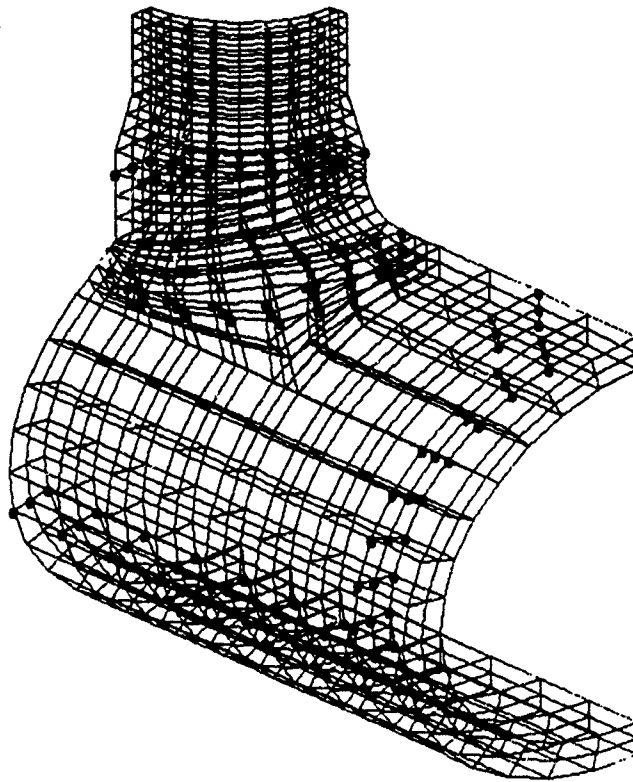


Fig. 3. Decomposition into four subdomains.

The mesh nodes which are common to the subdomain interfaces are not part of any D_j ; they define a unique global interface noted by D_I . The finite element preprocessor that performs this domain decomposition has been presented in [8]. It has been designed to achieve two basic goals, namely, (1) to deliver subdomains D_j that require equal time to process, and (2) to minimize the size of the global interface D_I in order to minimize the storage requirements (see Table 4, step (S3)). Numbering first the nodal point unknowns within D_j and last the ones within D_I results in an *arrow* pattern for the stiffness matrix (Fig. 5). Each diagonal block K_{jj} represents the local stiffness of a subdomain D_j . An off-diagonal block K_{ji} denotes the coupling stiffness between D_j and D_i . Block K_{II} is the stiffness of the global interface D_I .

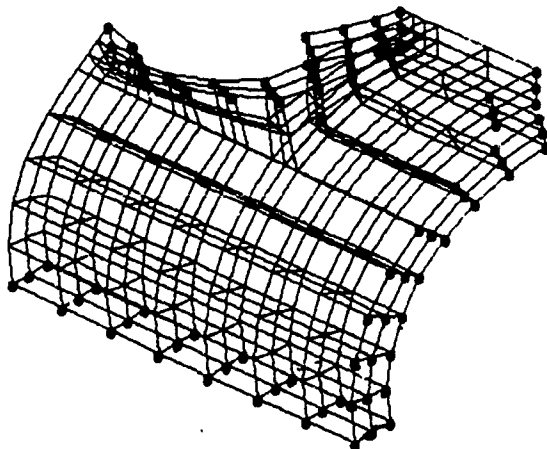


Fig. 4. Typical subdomain with its interface nodes.

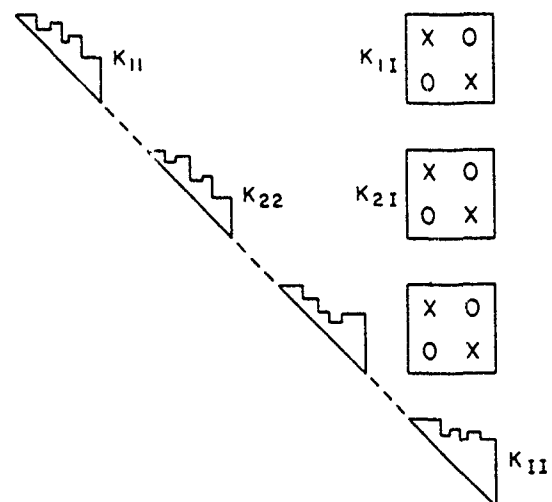


Fig. 5. Pattern of the resulting stiffness matrix.

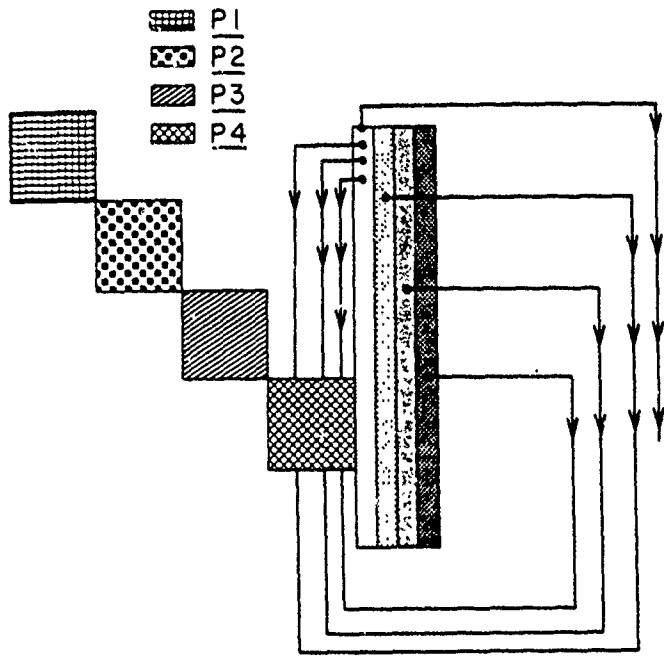
Table 4
Direct solution algorithm

(S1)	Factor	$K_{jj} = L_j D_j L_j^T$	$(j = 1, \dots, N_p)$
(S2)	Compute	$K_{ii}^T K_{jj}^{-1} K_{ji}$ and $K_{ji}^T K_{jj}^{-1} \delta f_j$	$(j = 1, \dots, N_p)$ $(j = 1, \dots, N_p)$
(S3)	Modify	$K_{ii} = K_{ii} - \sum_{j=1}^{N_p} K_{ji}^T K_{jj}^{-1} K_{ji}$ and $\delta f_i = \delta f_i - \sum_{j=1}^{N_p} K_{ji}^T K_{jj}^{-1} \delta f_j$	
(S4)	Solve	$K_{ii} \delta u_i = \delta f_i$	
(S5)	Back-solve	$K_{jj} \delta u_j = \delta f_j - K_{ji} \delta u_i$	$(j = 1, \dots, N_p)$

Similarly, $u_j(f_j)$ denotes the subvector of $u(f)$ which corresponds to the nodal point unknowns (forces) lying in D_j .

A hybrid data structure is used to implement the direct solution algorithm. Diagonal blocks K_{jj} and K_{ii} are stored in symmetric skyline/profile form. The mesh nodes are renumbered in parallel, each processor p_j renumbering a subdomain D_j , in order to minimize the storage requirements associated with the diagonal blocks K_{jj} . Because of their important sparsity off-diagonal blocks K_{ji} are stored column-wise in packed lists of nonzero elements. The algorithm for the solution of the complete finite element systems is given in symbolic form in Table 4. Initially each set of $\{K_{jj}, K_{ji}, \delta f_j, \delta u_j\}$ is assigned to an individual processor p_j so that steps (S1) and (S5) are trivially carried out in parallel. Note that in these factorization (S1) and back-solve (S5) phases, each processor operates on distinct locations of the shared memory so that a maximum efficiency is achieved. The subproblem in step (S4) is treated with a parallel active column equation solver [9]. Steps (S2) and (S3) require special attention. First note that because of symmetry in both $K_{ji}^T K_{jj}^{-1} K_{ji}$ and K_{ii} , only half of the computations in steps (S2) and (S3) need be performed. Moreover, given the data structure of the off-diagonal blocks K_{ji} , the computations in step (S2) are carried out column-by-column. After step (S1) is done, if the initial mapping, that is the mapping of each processor p_j onto the set $\{K_{jj}, K_{ji}, \delta f_j, \delta u_j\}$, is maintained through the subsequent computations, step (S3) would produce critical regions; each processor might have to modify the same half-column (or half-row) of K_{ii} at the same time. Hence, to avoid these critical regions we define a new mapping of the processors that is activated at the beginning of step (S2) and deactivated at the end of step (S3). Processor p_i is now mapped onto columns $i + kN_p$ of each of K_{ji} , $j = 1, 2, \dots, N_p$. Step (S2) is then carried out column by column in parallel. In this way, each processor is also mapped onto different entries of K_{ii} so that step (S3) does not involve any critical region. Moreover, to avoid memory contentions when fetching the same K_{jj} for operations on distinct columns of K_{ji} while computing $K_{ji}^T K_{jj}^{-1} K_{ji}$, the processors are initially interleaved: processor p_i starts first with the columns of K_{i1} , then $K_{(i+1)1}$, $\dots$, then picks up K_{i2} to $K_{(i-1)1}$. In summary, this mapping allows steps (S2) and (S3) to be executed in parallel without the need for any synchronization. Figure 6 summarizes the organization of the complete concurrent computations with the synchronization points.

Before the solution phase is activated, each processor is mapped onto a subdomain D_j . It forms the elemental stiffnesses corresponding to its subdomain and assembles the matrices K_{jj}



Processor j

Factor K_{jj}

Synchronize

Loop for $k = j$ to $k = N_p$

Loop for $i = j$ to $i = n_1$ step N_p

Compute $(K'_{ji} K_{jj}^{-1} K_{ji})[* , i], (* > i)$

Modify $K_{ii}[i, *], (* > i)$

Loop for $k = 1$ to $k = j - 1$

Loop for $i = j$ to $i = n_1$ step N_p

Compute $(K'_{ji} K_{jj}^{-1} K_{ji})[* , i], (* > i)$

Modify $K_{ii}[i, *], (* > i)$

Loop for $i = 1$ to $i = n_1$ step N_p

Compute $(K'_{ji} K_{jj}^{-1} \delta f_i)[i]$

Modify $\delta f_i[i]$

Synchronize

Join parallel solver for step (S4)

Synchronize

Form $\delta f_j = \delta f_j - K_{ji} \delta u_i$

Back-solve $K_{jj} \delta u_j = \delta f_j^*$

Fig. 6. Individual processor computations/synchronizations.

and K_{ji} . Then it computes the contribution of the elements of D_j to K_{ii} . After each iteration k , the number of elements which yield may vary from subdomain to subdomain. Hence, the initial mapping of the processors would result in an inefficient concurrent update of the tangent stiffness matrix $K(u_k)$, since some processors would have to work less than others and hence would have to wait for each other. The same is true for the state determination phase where the complexity of the computation of the internal forces depends on the constitutive equation that applies to the particular element in a specific subdomain. In order to achieve a much better load balance, two distinct processor mappings are used. Mapping_1 is the set of submappings defined previously for the concurrent direct solution of the complete system of linearized equations. Mapping_2 is the coloring scheme presented in Section 2. When mapping_2 is activated, the elements within a list L_i are processed concurrently for stiffness updates and state determination, on the basis of "first processor available first processor assigned". In other words, mapping_2 is exactly the general framework for parallel EBE computations (Section 2).

5. Software implementation and numerical results

A prototype for finite element computations on shared-memory multiprocessors is under development at the Center for Space Structures and Controls (University of Colorado at

Boulder). The software architecture and the numerical algorithms presented in this paper are one part of it. A test version has been implemented on Encore Multimax, Sequent Balance, Alliant, and Cray2 using the *Force* macros [5]. CDC's ETA-10 is the next target supermultiprocessor. The *Force* provides a FORTRAN-style parallel programming language utilizing an extensive set of parallel constructs. It offers two desirable advantages:

- (1) It insulates the programmer from process management, leaving him free to concentrate on the synchronization issues of parallel programming.
- (2) It ensures the portability of his code to several different shared-memory multiprocessors. Basically, the same code is run on any machine where the *Force* has been installed.

On Alliant computers, the compiler recognizes inherent parallelism at the DO loop level without the need for the programmer to invoke any explicit parallel construct. This may facilitate the work for the programmer. However, his code would not run on the Cray2, for example, because software support for multitasking on this super-multiprocessor is at the library level, where the user makes calls to ask the system for multitasking functions. For synchronization it is commonly necessary to wait until all processes have terminated a given task or to make sure that at any given time only one process modifies a variable. The procedures for these synchronizations are machine dependent. The *Force* relieves the programmer from the burden of modifying his code in order to port it to a new multiprocessor. Because only the *Force* constructs need be reprogrammed from one multiprocessor to another, his precious code need not be modified. For example, if the desire is to request all processes to wait until the longest one has terminated, the same *Force* construct "*Barrier*" is invoked on any multiprocessor. On the Cray2 running under UNICOS, the *Force* preprocessor will read the simple "*Barrier*" statement and generate the following complex FORTRAN code:

```
CALL LOCKON(BARLCK)
IF (FFNBAR.LT.(NP-1)) THEN
  FFNBAR = FFNBAR + 1
  CALL LOCKOFF(BARLCK)
  CALL LOCKON(BARWIT)
ENDIF
IF (FFNBAR .EQ. (NP - 1)) THEN
  ENDIF
IF(FFNBAR.EQ.0) THEN
  CALL LOCKOFF(BARLCK)
ELSE
  FFNBAR = FFNBAR - 1
  CALL LOCKOFF(BARWIT)
ENDIF
```

which invokes the appropriate UNICOS multitasking software utilities.

The parallelization of each computational step in the algorithms described in this paper is managed by the *barrier* concept as described in [5], rather than the fork/join mechanism.

In a fork/join mechanism [17], a single instruction stream would fork within some subroutine into multiple streams which would perform a parallel computation and then join

Boulder). The software architecture and the numerical algorithms presented in this paper are one part of it. A test version has been implemented on Encore Multimax, Sequent Balance, Alliant, and Cray2 using the *Force* macros [5]. CDC's ETA-10 is the next target supermultiprocessor. The *Force* provides a FORTRAN-style parallel programming language utilizing an extensive set of parallel constructs. It offers two desirable advantages:

- (1) It insulates the programmer from process management, leaving him free to concentrate on the synchronization issues of parallel programming.
- (2) It ensures the portability of his code to several different shared-memory multiprocessors. Basically, the same code is run on any machine where the *Force* has been installed.

On Alliant computers, the compiler recognizes inherent parallelism at the DO loop level without the need for the programmer to invoke any explicit parallel construct. This may facilitate the work for the programmer. However, his code would not run on the Cray2, for example, because software support for multitasking on this super-multiprocessor is at the library level, where the user makes calls to ask the system for multitasking functions. For synchronization it is commonly necessary to wait until all processes have terminated a given task or to make sure that at any given time only one process modifies a variable. The procedures for these synchronizations are machine dependent. The *Force* relieves the programmer from the burden of modifying his code in order to port it to a new multiprocessor. Because only the *Force* constructs need be reprogrammed from one multiprocessor to another, his precious code need not be modified. For example, if the desire is to request all processes to wait until the longest one has terminated, the same *Force* construct "*Barrier*" is invoked on any multiprocessor. On the Cray2 running under UNICOS, the *Force* preprocessor will read the simple "*Barrier*" statement and generate the following complex FORTRAN code:

```
CALL LOCKON(BARLCK)
IF (FFNBAR.LT.(NP-1)) THEN
  FFNBAR = FFNBAR + 1
  CALL LOCKOFF(BARLCK)
  CALL LOCKON(BARWIT)
ENDIF
IF (FFNBAR .EQ. (NP - 1)) THEN
  ENDIF
IF(FFNBAR.EQ.0) THEN
  CALL LOCKOFF(BARLCK)
ELSE
  FFNBAR = FFNBAR - 1
  CALL LOCKOFF(BARWIT)
ENDIF
```

which invokes the appropriate UNICOS multitasking software utilities.

The parallelization of each computational step in the algorithms described in this paper is managed by the *barrier* concept as described in [5], rather than the fork/join mechanism.

In a fork/join mechanism [17], a single instruction stream would fork within some subroutine into multiple streams which would perform a parallel computation and then join

into a single stream before returning from the subroutine. The disadvantage of this scheme lies in its poor performance. It is well known (Amdahl's second law) that even a small amount of sequential code in an otherwise parallel program can ruin the sought-after speed-up. Moreover, the overhead associated with fork/join mechanisms tends to be expensive. For example, it has been found that on the CRAY X-MP, a numerical parallel algorithm should perform at least 100,000 vector floating-point operations per fork/join operation to be efficient [18].

The *barrier* is a control-oriented synchronization mechanism. Its semantics are that all processors pause until the slowest processor reaches the barrier, then continue their prescribed job. It is more efficient than the fork/join mechanism because process environments need not be allocated or released [19].

The synchronization with the barrier of the parallel computational steps described earlier is illustrated in Fig. 7(a) for the particular case of a 4-processor machine and a 3-color problem. Basically, for each color characterizing a list L_j of internally disjoint elements, each processor p_i executes the same vector instructions on a corresponding subset of elements $L_j^{(i)}$, then waits until all processors have finished with processing their assigned elements before moving to the next color. The alternative use of fork/join mechanisms to perform the parallel processing of each color would require the allocation then the release of process environments three times, once for each forking and once for each joining of the processes (Fig. 7(b)).

The parallel computations presented in this paper are illustrated with the static nonlinear analysis of a space structure submitted to point loading. Each bay of the truss beam structure shown in Fig. 8 is made out of a four-longeron beam that has simple single-axis pivot hinges and double folds for efficient packaging. A series of battens and diagonals connect the longeron together at joints. The global structure has 1204 nodes, 3600 active degrees of freedom, and 3905 elements. The static nonlinear analysis is performed first with Jacobi-PCG and DR on the Alliant FX/8 (8 processors). The pattern of connectivity of the structure is relatively congested (up to 9 elements converging at a common node); it took 9 colors to partition the mesh into lists of internally disjoint elements. The repartition of the elements among the lists L_j is displayed in Table 5. Note that each list contains enough elements to justify the cost of the two global synchronization points needed for managing the parallel computations.

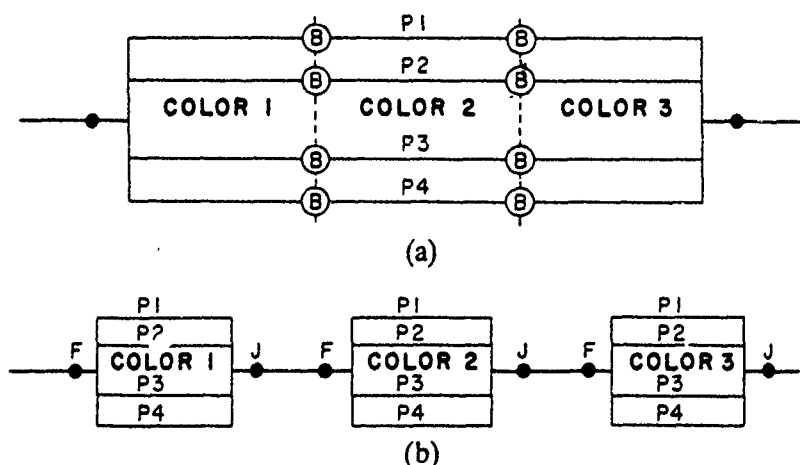


Fig. 7. (a) Synchronization with barriers. (b) Synchronization with fork/join mechanism.

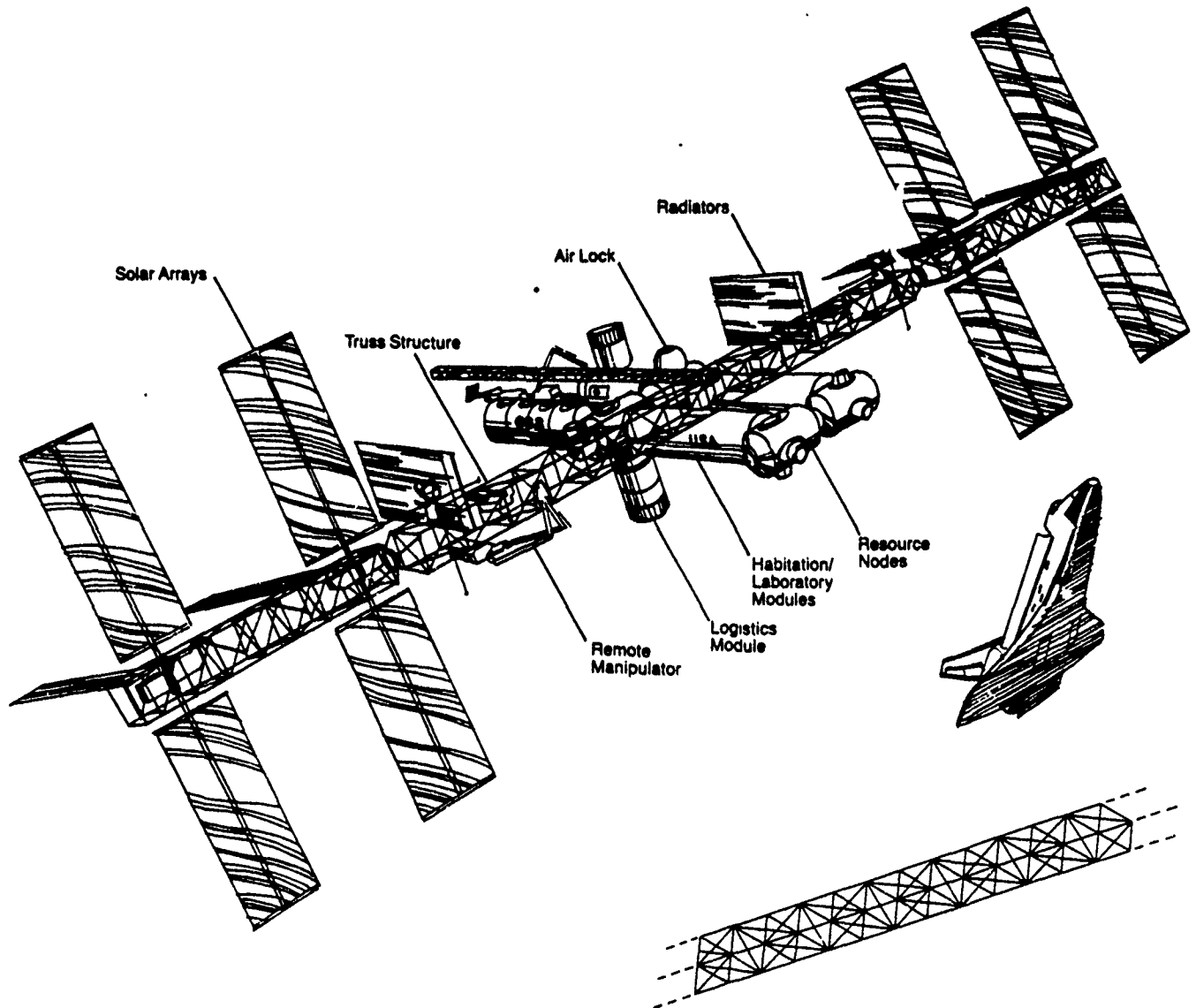


Fig. 8. Truss beam space structures, 1204 nodes, 3600 elements.

Experiments run with one to seven processors (the eighth being always used by the system manager) are reported in Table 6. The speed-up, S_{N_p} , is defined as the ratio T_1/T_{N_p} , where T_1 denotes the CPU time elapsed using only one processor and T_{N_p} denotes the CPU time elapsed using N_p processors. It is very important to note that T_1 measures the performance of a sequential version of the code different from the parallel one, in the sense that it does not contain any of the synchronization system calls. The efficiency rate is defined as the speed-up per processor. It is reported between parentheses, next to the achieved speed-up.

Table 5

Repartition of 3905 elements among 9 colors, number of elements is 3905, number of colors is 9

L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9
602	602	601	452	300	300	448	300	300

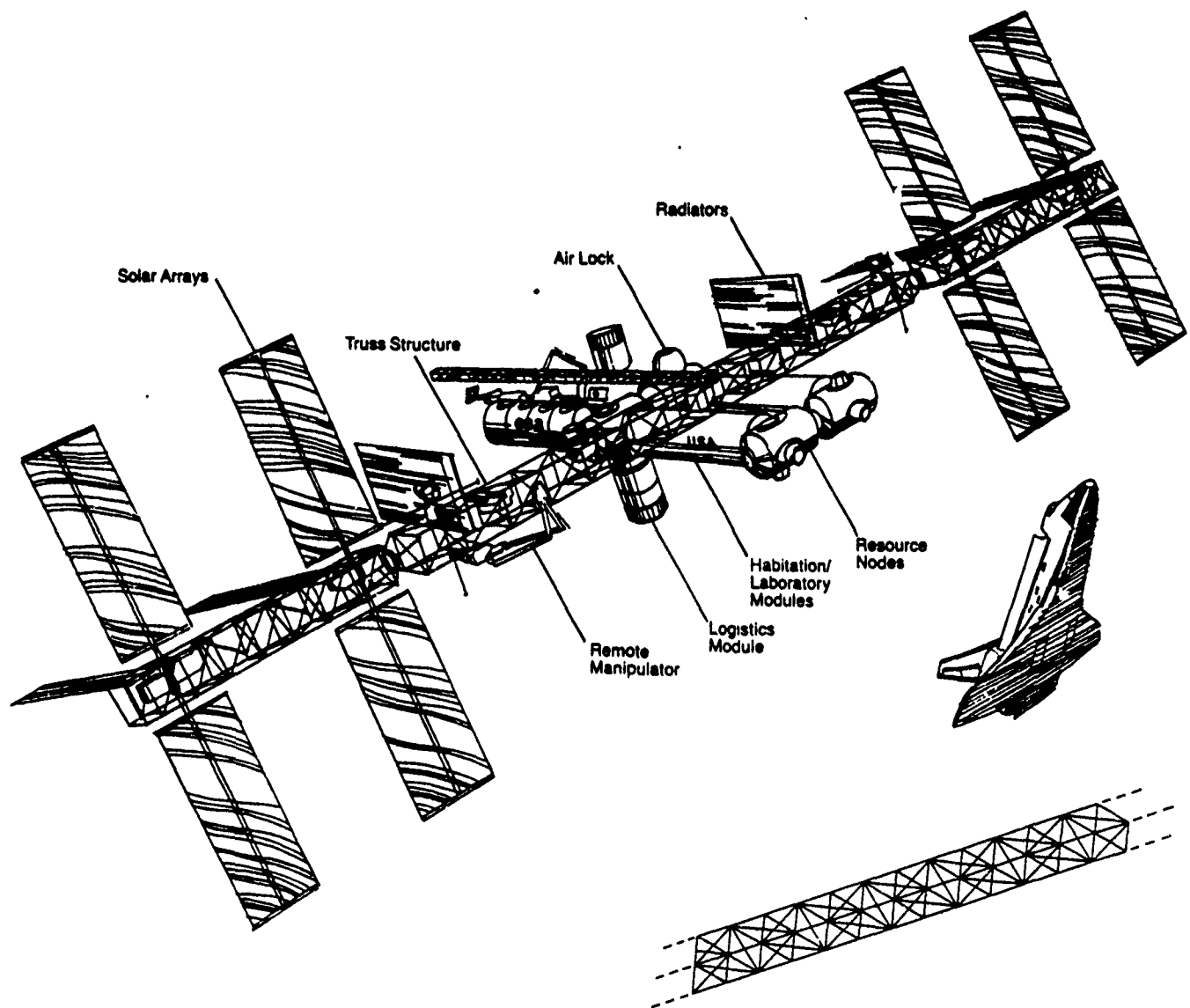


Fig. 8. Truss beam space structures, 1204 nodes, 3600 elements.

Experiments run with one to seven processors (the eighth being always used by the system manager) are reported in Table 6. The speed-up, S_{N_p} , is defined as the ratio T_1/T_{N_p} , where T_1 denotes the CPU time elapsed using only one processor and T_{N_p} denotes the CPU time elapsed using N_p processors. It is very important to note that T_1 measures the performance of a sequential version of the code different from the parallel one, in the sense that it does not contain any of the synchronization system calls. The efficiency rate is defined as the speed-up per processor. It is reported between parentheses, next to the achieved speed-up.

Table 5
Repartition of 3905 elements among 9 colors, number of elements is 3905, number of colors is 9

L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9
602	602	601	452	300	300	448	300	300

Table 6

Performance of parallel Jacobi-PCG and parallel DR on Alliant FX/8, problem size is 3600 dof, number of colors is 9

Number of processors	Speed-up for PCG	Speed-up for DR
2	1.92 (96%)	1.95 (98%)
3	2.89 (96%)	2.94 (98%)
4	3.77 (94%)	3.84 (95%)
5	4.50 (90%)	4.55 (91%)
6	5.13 (86%)	5.22 (87%)
7	5.56 (79%)	5.57 (80%)

The results in Table 6 demonstrate that relatively high rates of efficiency can be achieved. This is because a considerable amount of computations is performed in parallel between two synchronization points. For an increasing number of processors, the slight decrease in efficiency is due to bus contention.

The analysis is repeated with the concurrent direct solver on the Cray2. Since four processors are available on this supercomputer, the structure is partitioned into one, two, three, and four substructures analyzed respectively with one, two, three, and four processors. Resulting speed-ups and efficiency rates are summarized in Table 7.

Note that in each case, the algorithm for domain decomposition described in [8] performs extremely well at minimizing the number of interface degrees of freedom. Consequently, the stiffness submatrix K_{II} associated with D_I , which suffers fill-in after condensation of the degrees of freedom in each subdomain D_j , requires very little computer storage. This may not be the case for three-dimensional continuum meshes, but even for these problems, the Cray2 offers adequate memory space. Moreover, the resulting subdomains are perfectly balanced in each case. The efficiency of our implementation is clearly demonstrated by the resulting speed-ups. Similar performances are expected to be achieved on the ETA-10, which offers 8 superprocessors and 288 million double-precision words.

5. Conclusion

A general approach to nonlinear finite element parallel computations on shared-memory multiprocessors has been presented. The basic software architecture is built around an efficient framework that minimizes the overhead required for parallel accumulation of element-by-element concurrent computations. Two levels of parallelism are exploited: con-

Table 7

Performance on the concurrent direct solver on Cray2, problem size is 3600 d.o.f

D_1 (dof)	D_2 (dof)	D_3 (dof)	D_4 (dof)	D_I (dof)	Number of processors	Speed-up
1794	1794	—	—	12	2	1.98 (99%)
1192	1192	1192	—	24	3	2.97 (99%)
889	889	889	888	45	4	3.80 (95%)

currency at the outer-loop and vectorization at the inner-loop. A highly portable prototype code has been developed using the *Force*. It features a parallel implementation of iterative algorithms such as Jacobi and EBE preconditioned conjugate gradient and dynamic relaxation, and a concurrent direct solver based on substructuring. The parallel iterative algorithms are very efficient for solving very large sparse problems on multiprocessors where the shared memory cannot accommodate an in-core solution. On the other hand, the direct solver may be necessary for the analysis of very flexible space structures which are inherently ill-conditioned. The achieved speed-ups on Alliant FX/8 and Cray2 confirm the potential of this approach to parallel computing on shared-memory multiprocessors. The extension of this work to massively parallel multiprocessors is being carried out on the Connection Machine and will be the subject of a forthcoming paper.

Acknowledgment

The authors acknowledge the valuable contributions of their colleagues Muhammad Benten and Professors Harry Jordan and K.C. Park at the University of Colorado, Boulder, CO, to the development of this challenging technology. They would like also to thank Dr. Jerry Housner at NASA Langley for arranging NAS-Cray2 computer time. The first author wishes to acknowledge the partial support of a CDC PACER Fellowship Award, with Dr. R.F. Woodruff as technical monitor, and partial support by NASA Langley under Grant NAG-1-756, with Dr. Jerry Housner as technical monitor. The second author acknowledges partial support by the Air Force Office of Scientific Research under Grant F49620-87-C-0074, with Dr. Anthony K. Amos as technical monitor.

References

- [1] D.A. Poplawski, Parallel computer architectures, *Appl. Math. Comput.* 20 (1) (1986) 41-51.
- [2] C. Farhat, Multiprocessors in computational mechanics, Ph.D. Thesis, University of California at Berkeley, Berkeley, CA, 1986.
- [3] C. Farhat, E. Wilson and G. Powell, Solution of finite element systems on concurrent processing computers, *Engrg. Comput.* 2 (3) (1987) 157-165.
- [4] B. Nour-Omid and K.C. Park, Solving structural mechanics problems on the CALTECH hypercube machine, *Comput. Meth. Appl. Mech. Engrg.* 61 (1987) 161-176.
- [5] H. Jordan, M. Benten and N. Arenstorf, *Force user's manual*, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO, 1987.
- [6] T.J.R. Hughes, R.M. Ferencz and J.O. Hallquist, Large-scale vectorized implicit calculations in solid mechanics on a Cray X-MP/48 utilizing EBE preconditioned conjugate gradients, *Comput. Meths. Appl. Mech. Engrg.* 61 (1987) 215-248.
- [7] P.G. Underwood, Dynamic relaxation techniques: A review, in: T. Belytschko and T.R.J. Hughes, eds., *Computational Method for Transient Analysis* (North-Holland, Amsterdam, 1982) 245-267.
- [8] C. Farhat, A simple and efficient automatic FEM domain decomposer, *Comput. & Structures* 28 (5) (1988) 579-602.
- [9] C. Farhat, A parallel active column equation solver, *Comput. & Structures* 28 (4) (1988) 289-304.
- [10] P. Berger, P. Brouaye and J. Syre, A mesh coloring method for efficient MIMD processing in finite element problems, in: *Proceedings International Conference on Parallel Processing* (1982) 41-46.

- [1] L. Hayes and P. Devloo, An element by element block iterative method for large nonlinear problems, in: W.K. Liu, T. Belytschko and K.C. Park, eds., *Proceedings International Conference on Innovative Methods for Nonlinear Problems* (Pineridge Press, Swansea, U.K., 1984) 51-62.
- [2] G. Carey and B. Jiang, Element-by-element preconditioned conjugate gradient algorithm for compressible flow, in: W.K. Liu, T. Belytschko and K.C. Park, eds., *Proceedings International Conference on Innovative Methods for Nonlinear Problems* (Pineridge Press, Swansea, U.K., 1984) 41-49.
- [3] H.A. Van Der Vorst, The performance of FORTRAN implementations for preconditioned conjugate gradients on vector computers, *Parallel Computing* 3 (1986) 49-58.
- [4] K. Law, A parallel finite element solution method, *Comput. & Structures* 23 (6) (1986) 845-858.
- [5] J. Kowalik and S. Kumar, An efficient parallel block conjugate gradient method for linear equations, in: *Proceedings 1982 International Conference on Parallel Processing* (1982) 47-52.
- [6] R. Benner and G. Montry, Overview of preconditioned conjugate gradient (PCG) methods in concurrent finite element analysis, Internal Rept. from the Advanced Computer Science Project, NTIS Rep. No. SAND85-2727, Fluid and Thermal Sciences Department, Sandia National Laboratories, Albuquerque, NM, 1986.
- [7] J.B. Dennis and E.C. Van Horn, Programming semantics for multiprogrammed computations, *Comm. ACM* (3) (1966) 143-155.
- [8] M. Seager, Parallelizing conjugate gradient for the Cray X-MP, *Parallel Computing* (1986) 273-279.
- [9] H. Jordan, Structuring parallel algorithms in an MIMD, shared memory environment, *Parallel Computing* 3 (1986) 93-110.
- [20] M. Papadrakakis, A method for the automated evaluation of the dynamic relaxation parameters, *Comput. Meths. Appl. Mech. Engrg.* 25 (1981) 35-48.