NPS-ME-99-009

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Efficient Nonlinear Transient Dynamic Analysis for Structural Optimization Using an Exact Integral Equation Formulation

by

Joshua H. Gordis
Beny Neta

May 1999

Prepared for: National Science Foundation

This report was prepared for THE NATIONAL SCIENCE FOUNDATION - EARTHQUAKE HAZARD MITIGATION PROGRAM

and funded by THE NATIONAL SCIENCE FOUNDATION

This report was prepared by:

[illegible]
Joshua H. Gordis
Associate Professor of Mechanical Engineering

[illegible]
Beny Neta
Professor of Mathematics

Reviewed by:

[illegible]
Terry McNelley
Chairman, Dept. of Mechanical Engineering

Released by:

[illegible]
D. W. Netzer
Associate Provost and Dean of Research

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE 23 November 1999	3. REPORT TYPE AND DATES COVERED Progress Report – July 1998 through November 1999

4. TITLE AND SUBTITLE Efficient Nonlinear Transient Dynamic Analysis for Structural Optimization Using an Exact Integral Equation Formulation	5. FUNDING NUMBERS ATM-9713481
6. AUTHOR(S) Joshua H. Gordis and Beny Neta	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) The National Science Foundation Earthquake Hazard Mitigation Program Dr. S. C. Liu, Program Director 4201 Wilson Blvd. Room P60 Arlington VA 22230	10. SPONSORING/MONITORING AGENCY REPORT NUMBER #97-13481
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release: distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report serves to document progress made to date on the National Science Foundation Project #97-13481, Earthquake Hazard Mitigation Program. The focus of this phase of the project is the development of an improved solution algorithm for fast transient analysis of large, locally nonlinear structures using time domain structural synthesis. The report documents the development of several algorithms intended to improve upon the original algorithm developed by the first author. The last algorithm developed meets the stated goals of the project. This algorithm is a newly developed recursive, block-by-block convolution solution to the governing nonlinear integral equation. As is demonstrated with a realistically large nonlinear base excitation problem (51,500 DOF finite element model), the new algorithm provides a 78% reduction in time required for the nonlinear transient base excitation solution, as compared with traditional direct integration. The new algorithm provides an even greater reduction in computer time for subsequent analysis. The nonlinear base isolation solutions calculated using the new algorithm take approximately 7 seconds, as compared with the direct integration solution which takes approximately 30 minutes. The rapid reanalysis capability will facilitate the development of numerical optimization for the design of nonlinear isolation. The theory of transient synthesis is documented, along with a new proof of the exponential convergence properties of an iterative solution to the governing nonlinear integral equation.

14. SUBJECT TERMS Structural dynamics, earthquake, isolation, nonlinear transient analysis, structural synthesis	15. NUMBER OF PAGES 77
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT none	18. SECURITY CLASSIFICATION OF THIS PAGE none	19. SECURITY CLASSIFICATION OF ABSTRACT none	20. LIMITATION OF ABSTRACT none

Progress Report

Efficient Nonlinear Transient Dynamic Analysis for Structural Optimization Using an Exact Integral Equation Formulation

(NSF Project #97-13481)

Joshua H. Gordis
Department of Mechanical Engineering
jgordis@nps.navy.mil

Beny Neta
Department of Mathematics
bneta@nps.navy.mil

Naval Postgraduate School
Monterey, CA 93943-5000

Prepared for the

National Science Foundation
Earthquake Hazard Mitigation Program
Dr. S. C. Liu, Program Director

Principal Author Contact Information:

Joshua H. Gordis
Naval Postgraduate School
Department of Mechanical Engineering – Code ME/Go
Monterey, CA 93943-5146
(831) 656-2866 voice
(831) 656-2238 FAX
jgordis@nps.navy.mil

ABSTRACT

This report serves to document progress made to date on National Science Foundation Project #97-13481, Earthquake Hazard Mitigation Program. The focus of this phase of the project is the development of an improved solution algorithm for fast transient analysis of large, locally nonlinear structures using time domain structural synthesis. Time domain structural synthesis is a general and exact formulation for transient problems in structural modification, substructure coupling, and base excitation. The formulation is characterized by the governing equation of the synthesis, which is a nonlinear Volterra integral equation. The governing equation makes use of impulse response functions calculated for those coordinates of the (sub)structures subjected to forces of synthesis (e.g. modification forces, coupling forces). This physical coordinate formulation provides for a largely unrestricted and exact model reduction, in that only those coordinates of interest need be retained in the synthesis.

The report documents the development of several algorithms intended to improve upon the original algorithm developed by the first author. The last algorithm developed is presented first in this report, as this algorithm meets the stated goals of the project. This algorithm is a newly developed recursive, block-by-block convolution solution to the governing nonlinear integral equation. As is demonstrated with a simple but realistically large nonlinear base excitation problem (51,500 DOF finite element model), the new algorithm provides a 78% reduction in time required for the nonlinear transient base excitation solution, as compared with traditional direct integration calculated using a widely-used commercial finite element program. This very large savings in computer time is obtained for a single analysis, i.e. assuming no prior calculations have been made for the impulse response functions of the (sub)structures. The new algorithm provides an even greater reduction in computer time for all subsequent analyses. As shown in the example problem, once all required impulse response functions have been calculated, the nonlinear base isolation solutions calculated using the new recursive, block-by-block convolution algorithm take approximately 7 seconds, as compared with the direct integration solution which takes approximately 30 minutes. This rapid reanalysis capability will facilitate the development of numerical optimization for the design of nonlinear isolation. The theory of transient synthesis is documented, along with a new proof of the exponential convergence properties of an iterative solution to the governing nonlinear integral equation.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of the National Science Foundation, Dr. S. C. Liu, Program Director, Earthquake Hazard Mitigation.

This work is dedicated to my family: Christy, Hannah, and Joseph.

Joshua H. Gordis
Monterey, CA 1999

The fact of harmony between Heaven and Earth and Man does not come
from a physical union, from a direct action,
it comes from a tuning on the same note producing vibrations in unison.

Tong Tshung-chu
Second Century, b.c.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The transient analysis of large and complex structural systems is a computationally demanding task exacerbated by the presence of structural and mechanical nonlinearities. The computational demand of these problems prohibits the repeated analyses required in a design effort, such as in structural optimization where various responses are required for the calculation of the objective function, constraints, sensitivities, and for the generation of approximations to be used within the optimizer.

A class of nonlinear structural dynamics problems with numerous applications is characterized by the presence of *localized* nonlinearities. For the purposes of this work, this class of problems is defined as follows:

> Definition of a Locally Nonlinear Model: A model where the nonlinear load paths do not contain any internal degrees-of-freedom (DOF), i.e. each nonlinear load path (nonlinear element) is associated solely with DOF shared by linear load paths (elements).

This class of problems can be further informally restricted by recognizing that the formulations to be developed in what follows provide a greater reduction in computing time (as compared with direct integration) for models where there are relatively few nonlinear load paths, or in other words, where the number of DOF associated with nonlinear load paths is small relative to the total number of DOF in the model. The problem of nonlinear earthquake isolation of a linear structure falls into this category, wherein the isolator provides a nonlinear load path between the building model DOF and "ground."

A locally nonlinear model allows for algorithmic approaches which exploit this characteristic to achieve reduced computer time requirements. The most common strategy for this exploitation is where the localized nature of the nonlinearities facilitates their treatment as terms on the right hand side, i.e. applied loads of differential equations, albeit loads which are functions of system responses, and possibly with time, see for example [1].

The approach taken in this work is to treat the problem as a physical coordinate (non-modal) structural modification problem, wherein the nonlinear elements are "installed" into the linear model as structural modifications. The structural modification formulation belongs to the broader category of physical coordinate *structural synthesis* [2-6], which includes substructure coupling and constraint imposition as well. Such an approach not only provides a substantial reduction in solution times, but provides for a generality in the definition of the problem and a flexibility in its application which is unique. We will describe structural synthesis here.

In a sense, structural synthesis treats the nonlinear element responses as applied loads as well. However, what distinguishes structural synthesis from other numerical approaches are the following characteristics:

- The governing equations for structural synthesis are exact.
- An implicit exact model reduction is available, in that, as a minimum only those DOF directly associated with nonlinear elements and applied loading need be retained. Any additional physical DOF of interest to the analyst can be retained as well. In other words, the transient synthesis solution time is independent of model size.
- Very general nonlinearities can be treated.
- The linear portion(s) of the model is solved once.
- Very fast solution times are obtained, an intrinsic property of the formulation.

1.1 Summary of Results Reported for Year One

The goal of the Year One phase of this project is stated here:

The development of a highly efficient and general formulation for nonlinear transient structural synthesis which will provide fast linear or nonlinear transient re-analysis with arbitrary loading for large linear structural FE models which can have localized, but generally nonlinear components of arbitrary magnitude. The nonlinear formulation will be an extension of prior research of the PI [3].

This goal is met by the algorithm developed in Section 3 of this report.

1.2 Notes on Algorithms Developed – Organization of Report

The work reported herein results in the development of a fast numerical solution algorithm for the governing equation of locally nonlinear transient structural synthesis. This is the primary goal of the Year One effort. In pursuit of this goal, three general recursive algorithms were developed, and these algorithms are numbered in the order in which they were developed. However, the third algorithm developed resulted in the recursive block-by-block convolution algorithm (Recursive Algorithm 3) which meets the stated goals (Section 1.1) for this phase of the project. This algorithm will be the focus of continued work, and this algorithm will be presented first (Section 3).

We will detail all of the algorithms developed, including those either deemed ineffective for the task at hand (nonlinear isolation simulation) or surpassed by algorithms subsequently developed (and described herein) for the purpose of documenting "lessons learned," and to document novel results obtained, of potential value in other areas. We summarize briefly the algorithms here.

The governing equation for transient structural synthesis is a nonlinear Volterra integral equation, involving a convolution-type kernel [3]. The convolution-type kernel suggests the recursive transition-matrix approach to the solution of first-order ordinary differential equations as a potential improvement over the (non-recursive) iteration solution presented by Gordis and Radwick [3], which itself demonstrated an order-of-magnitude reduction in computing time required, relative to direct nonlinear transient integration. A recursive transition-matrix approach is taken in the work of Inaudi and De La Llera [7], and this work provided additional motivation in this direction. However, in [7], the recursion was based on the transition matrix for the system model, and hence requires the calculation of a large matrix exponential, with no provision for model reduction. We must therefore consider such an approach to be of limited value for large structural models. Our goal was to explore the development of a recursive transition-matrix algorithm appropriate for large models, i.e. having an implicit model reduction. The developments and results of this phase of the work are detailed in Section 4, and include Recursive Algorithms 1 and 2, which are summarized here.

The first two of the above-mentioned algorithms are based on this transition-matrix approach. These algorithms are as follows:

1.2.1 Recursive Algorithm 1 (RA1)

RA1 is included as a necessary precursor to the algorithm RA2 which follows, rather than due to any particular novelty or computational value. The algorithm RA1 is based on transition matrices constructed from the modal transient response in first-order (state-space) form. As will be shown, this algorithm is inherently unstable, and can only be made stable by the inclusion of unrealistic levels of system damping. It is included, however, as a precursor for the following algorithm, and as a baseline for comparison purposes.

1.2.2 Recursive Algorithm 2 (RA2)

This algorithm develops a new recursive transition matrix solution for transient structural dynamics. The algorithm defines a *complex modal state vector*, and an associated transition matrix which is based on second-order modal differential equations. The motivation for the development of this algorithm was to avoid the formulation of the problem in terms of first-order differential equations, i.e. such as in RA1, in order to maintain consistency of the formulation with common tools for structural dynamics analysis, e.g. commercial finite element programs. Furthermore, as will be shown, RA2 achieves a significant computational advantage relative to

RA1, as RA2 is more "diagonalized." Unfortunately, RA2 is also inherently unstable, again, only stabilized by the inclusion of substantial system damping.

1.2.3 Recursive Predictor-Corrector Algorithms 1 and 2 (RA1PC and RA2PC)

The inherent instability of algorithms RA1 and RA2 motivated their reformulation as *predictor-corrector* algorithms, similar to the approach in [7]. We will refer to these implicit versions as RA1PC and RA2PC.

Algorithms RA1 and RA2 are reformulated as predictor-corrector algorithms, with a single corrector step. The stability calculations for RA1PC and RA2PC indicate that it is possible to stabilize these algorithms by going to implicit versions. However, the results indicate that further development of these algorithms would be necessary in order to improve on the performance already obtained by Gordis and Radwick [3]. Hence, Recursive Algorithm 3 is developed.

1.2.4 Recursive Algorithm 3: Block-by-Block Convolution

This algorithm is markedly different from RA1 and RA2 in that no transition matrix is employed. This algorithm preserves the physical coordinate formulation originally developed by Gordis [2] and Gordis and Radwick [3], and hence preserves the implicit and unrestricted exact model reduction, concomitant with the formulation. The algorithm will be shown to be exponentially convergent, for a general class of nonlinearities. The current version of the software makes use of an FIR filter to calculate the convolution sum, as implemented in the MATLAB® built-in convolution function ("CONV"). It should be noted that the FIR implementation in MATLAB® is not optimally efficient as one can develop a filter which computers only those elements required. However, this characteristic of MATLAB's convolution algorithm will yield conservative computer time comparisons. Recursive Algorithm 3 (RA3) is the first algorithm discussed, in Section 3 of this report.

2. *INTEGRAL EQUATION FORMULATION FOR TRANSIENT STRUCTURAL SYNTHESIS*

As the Year One task focuses on the development of an improved solution algorithm for the governing equation for locally nonlinear transient structural synthesis, we provide the background in the relevant theory. The reader is referred to Gordis [2] and Gordis and Radwick [3] for the complete development. The following section is abstracted from [3].

The theory defines a transient analysis that is independent of model size, in that the theory is cast in physical coordinates (i.e. non-modal), and the transient analysis is done using only those structural DOF of interest. These DOF must include, as a minimum, those associated with the nonlinear elements, which are treated independently of the (linear) model. Additionally, other DOF for which synthesized response information is desired can be included as needed. Therefore, it is possible to synthesize the transient response for an arbitrarily large model using a minimal number of DOF, the minimum number defined only by the number of nonlinear elements in the model. This unique feature of the theory has been demonstrated for the linear transient formulation [2], the nonlinear transient formulation [3], and in the frequency domain in [4-6]. Functioning as a re-analysis procedure, the formulation directly calculates the new transient response for a system resulting from structural changes and/or coupling with other structures, without a reassembly or full reanalysis. These features will be more fully described in what follows.

Structural synthesis also provides for substructure coupling. This feature allows nonlinear elements to be isolated by division of the system into substructures, which provides additional computational advantages in that the synthesis can exploit inherent physical boundaries in a problem. One immediate benefit resulting is that linear substructures are solved once, which constitutes a significant savings in that the eigensolution is typically the most expensive part of a dynamic analysis [8]. The synthesis is used to connect the substructures through the nonlinear elements, and to calculate the combined system nonlinear transient response.

Each substructure is described by impulse response functions calculated at the DOF where nonlinear elements are to be installed, where loads are applied, and at other DOF for which synthesized nonlinear transient response is required. For each linear substructure, the IRF are most efficiently calculated using modal superposition. However, the use of modal superposition for IRF calculation does not render structural synthesis a "modal method," for the following reason. The IRF are calculated using a sufficient number of modes to ensure convergence. Once these converged IRF are calculated, they are indistinguishable (to a given level of precision) from the "exact" IRF, which are indeed physical quantities. We can contrast this approach with modal methods of structural synthesis, where the convergence relative to the number of retained modes is determined from the *solution* of the synthesized system. Typically, a modal synthesis procedure generates, from substructure solutions, a coupled system model for which an additional solution is required. Furthermore, modal synthesis methods are inherently approximate. The governing equations of structural synthesis to be developed here are exact. The only errors incurred are those avoidable errors resulting from retaining an insufficient number of modes, and those errors common to all numerical integration schemes, which are discretization errors, and truncation errors where numerical derivatives are used.

2.1 Notation Used

In what follows, all vector quantities are denoted by boldface type, e.g. $\mathbf{x}$. A scalar element of a vector quantity will be denoted by a subscripted plainface type, e.g. x_i, which denotes the ith element of the vector $\mathbf{x}$. We will define various coordinate sets, which often are defined as subsets of a vector of coordinates. For example, if the coordinate vector $\mathbf{x}$ is comprised of two subsets of coordinates, say the α set and the β set, we will denote these coordinate subsets as $\mathbf{x}_\alpha$ and $\mathbf{x}_\beta$. The coordinate set contained in the vector $\mathbf{x}$ is the union of subsets α and β. The vector $\mathbf{x}$ can be written as $\mathbf{x} = \lfloor \mathbf{x}_\alpha^T \quad \mathbf{x}_\beta^T \rfloor^T$. A particular vector is denoted by a subscripted boldface symbol, such as the i^{th} eigenvector, denoted $\boldsymbol{\phi}_i$.

We will also need to distinguish quantities associated with various mathematical domains such as physical, modal, state-space, and the vector quantities comprised thereof. Quantities comprised of, or associated with physical (nodal) coordinates will have no embellishment. Quantities comprised of, or associated with modal coordinates will be denoted with the tilde embellishment $(\tilde{\bullet})$. Quantities comprised of, or associated with state-space coordinates will be denoted with the caret embellishment $(\hat{\bullet})$. Quantities comprised of, or associated with the complex modal coordinates (to be defined) will be denoted with the bar embellishment $(\bar{\bullet})$. In general, these will also be evident from context.

The various algorithms presented involve discrete time histories, e.g. $\mathbf{x}(k\Delta t)$ or $\mathbf{x}(kT)$ where the sample length is $\Delta t = T$ in seconds. We will abbreviate this notation with the following superscript notation, $\mathbf{x}(k\Delta t) = \mathbf{x}(kT) = \mathbf{x}^{(k)}$.

2.2 Theory

In the context of the physical coordinate synthesis formulation to be developed, a structural system is defined to consist of one or more uncoupled substructures. A single governing equation for nonlinear transient synthesis will be derived and this equation will address each of the following three general analysis categories:

(1) Structural modification - the addition and/or removal of linear and/or nonlinear structural elements

(2) Prescribed base motion - application of base motion to structure through linear and/or nonlinear elements

(3) Substructure coupling - the joining of substructures (a linear analysis)

Each of the above analysis categories defines a set of DOF. Referring to Figure 1, a structural system is shown, comprised of two substructures, each of an arbitrary number of DOF.

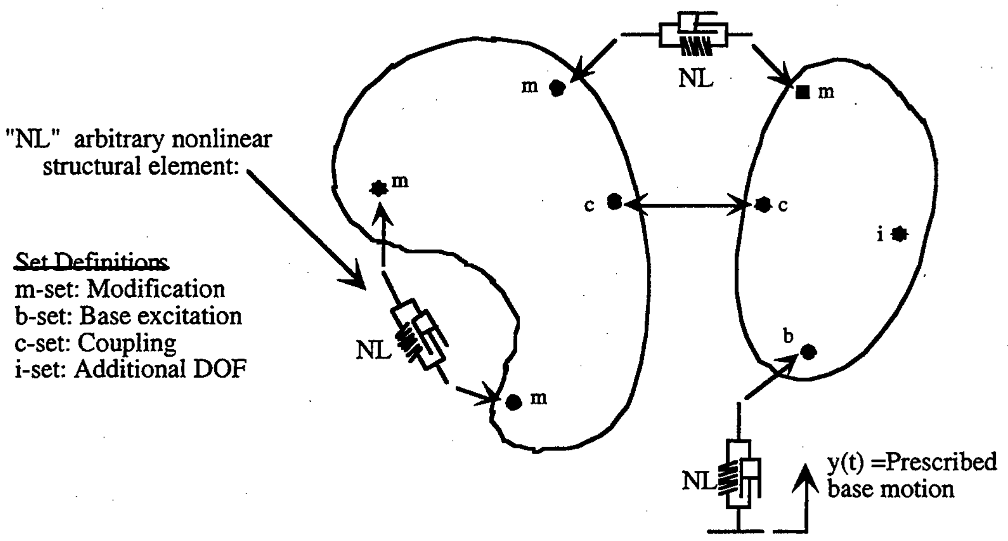

Figure 1. General structural system for synthesis comprised of two substructures

The subset of DOF associated with structural modifications is denoted the "m-set" and may include DOF from all substructures. The subset of DOF associated with prescribed base motion excitation is denoted the "b-set" and again, may include DOF from all substructures. The subset of DOF associated with substructure coupling is denoted the "c-set" and may include DOF from all substructures. Given that all nonlinear elements are installed in the synthesis, the substructures are linear, and hence coupling is a linear synthesis [2]. The subset of DOF denoted the "i-set" refers to those system DOF about which synthesized response information is required, but are not directly involved with the synthesis, either modification, base excitation, or coupling. The DOF sets are:

m-set:	Modification
b-set:	Base excitation
c-set:	Coupling
i-set:	Additional DOF

While much of the attention in this work will focus on single substructure systems, we will often refer to "substructures" as opposed to "structures" in keeping with the formulation of the problem as one of structural synthesis.

2.3 Basic Definitions for a Substructure Model

A finite element (FE) discretization produces the following linear (sub)structure model:

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{f}_e(t) \tag{2.1}$$

where the mass matrix **M** is (N x N) symmetric positive definite, the damping matrix **C** is (N x N) symmetric positive-semidefinite, and the stiffness matrix **K** is (N x N) symmetric positive-semidefinite. The inertial nodal displacement vector **x**(t) is (N x 1), as are the nodal velocity and acceleration, denoted $\dot{\mathbf{x}}(t)$ and $\ddot{\mathbf{x}}(t)$ respectively. The loading vector $\mathbf{f}_e(t)$ contains time-dependent loads acting at the nodal degrees-of-freedom (DOF). The FE model possesses N DOF. In the case of base excitation (e.g. ground motions), the loading vector can be written as,

$$\mathbf{f}_e(t) = \mathbf{K}_g\mathbf{y}(t) + \mathbf{C}_g\dot{\mathbf{y}}(t) \tag{2.2}$$

where $\mathbf{K}_g$ and $\mathbf{C}_g$ are the stiffness and damping matrices associated with (multiple) base displacement and velocity excitation. Note that applied forces and moments, and base excitation may co-exist.

The mass and stiffness matrix possess the eigenpairs, $\omega_i, \boldsymbol{\phi}^{(i)}$, i=1,2,..,N. The eigenvalue ω has the units sec^{-1}. The structure may possess up to 6 rigid-body modes, distinguished by ω_i=0, i = 1,2,..,r $\leq$ 6 where r is the number of rigid body modes possessed by the FE model. The following orthogonality relations are assumed to hold for all linear (sub)structures: $\mathbf{\Phi}^T\mathbf{M\Phi} = \mathbf{I}$ and $\mathbf{\Phi}^T\mathbf{K\Phi} = \mathrm{diag}(\omega^2)$, i.e. the modal matrix $\mathbf{\Phi}$is the mass-normalized. Also, for lightly damped structures, $\mathbf{C}$ is commonly defined to be proportional, and hence, $\mathbf{C} = \mathbf{M\Phi}\,\mathrm{diag}(2\varsigma\omega)\mathbf{\Phi}^T\mathbf{M}$ where ζ_i is the dimensionless damping ratio for the ith mode. This expression is valid for $0 \le \zeta < 1$ (underdamped). For an underdamped structure, the ith damped natural frequency is $\omega_{d_i} = \omega_i\sqrt{1-\varsigma_i^2}$.

2.4 Inertial and Relative Coordinate Subsets

Given the importance of base excitation to this work, the use of relative coordinates instead of inertial coordinates will allow the direct application of prescribed acceleration to the model, eliminating the need to differentiate a base displacement or velocity time history. Of course, if inertial displacements or velocities are to be recovered, the acceleration time history must be integrated.

We can define a set of prescribed ground displacement and velocity time histories, $y_i(t)$, $\dot{y}_i(t)$, i=1,2,...,g. We also define g subsets $\mathbf{z}_j$ of the inertial coordinates $\mathbf{x}$, where each member of each subset is to be replaced with a coordinate relative to a specific ground motion, $y_i(t)$. These coordinate subsets are denoted $\mathbf{z}_j(t)$, j = 1,2,...,g, where each relative coordinate set, $\mathbf{z}_j$ has n_j coordinates. This set definition allows any subset of inertial coordinates to be redefined as relative to any of multiple ground motions being applied to the model, and allows the retention of m inertial coordinates, $\mathbf{x}_m$, if required. Therefore, the length of the $\mathbf{z}$ vector is $N = m + \sum_{j=1}^{g} n_j$.

The coordinate and ground motion vectors can be constructed as follows,

$$\mathbf{z} = \begin{Bmatrix} \mathbf{x}_m \\ \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_g \end{Bmatrix}, \qquad \mathbf{y} = \begin{Bmatrix} y_1(t) \\ \vdots \\ y_g(t) \end{Bmatrix},$$

and the coordinate transformation between the exclusively inertial coordinate vector and the mixed inertial/relative vector is,

$$\mathbf{x} = \mathbf{z} + \mathbf{G}\mathbf{y}. \tag{2.3}$$

Transforming Eq. (2.1) provides,

$$\mathbf{M}\ddot{\mathbf{z}} + \mathbf{C}\dot{\mathbf{z}} + \mathbf{K}\mathbf{z} = \mathbf{f}_e(t) - \mathbf{M}\mathbf{G}\ddot{\mathbf{y}} - \mathbf{C}\mathbf{G}\dot{\mathbf{y}} - \mathbf{K}\mathbf{G}\mathbf{y}. \tag{2.4}$$

Since **CG=C**$_g$ and **KG=K**$_g$ (see Eq. (2.2)), we have,

$$\mathbf{M}\ddot{\mathbf{z}} + \mathbf{C}\dot{\mathbf{z}} + \mathbf{K}\mathbf{z} = \mathbf{f}_e^{(r)}(t) - \mathbf{M}\mathbf{G}\ddot{\mathbf{y}} = \mathbf{f}_e^{(r)}(t) - \mathbf{M}_g\ddot{\mathbf{y}} \tag{2.5}$$

where the superscript (r) is used here to denote that elements of the loading vector containing base excitation terms have been "zero-ed," consistent with the transformation to relative coordinates. We have kept the base acceleration terms separate from $\mathbf{f}_e^{(r)}$ for clarity, although we can redefine $\mathbf{f}_e = \mathbf{f}_e^{(r)} - \mathbf{M}_g\ddot{\mathbf{y}}$, recognizing that

$$\mathbf{M}\ddot{\mathbf{z}} + \mathbf{C}\dot{\mathbf{z}} + \mathbf{K}\mathbf{z} = \mathbf{f}_e \tag{2.6}$$

is essentially the same equation as Eq. (2.1), but where the use of the symbol **z** emphasizes that relative coordinates are included.

2.5 Derivation of Governing Equation of Nonlinear Transient Synthesis

The total solution for (linear) transient response can be written in terms of the convolution integral,

$$\mathbf{x}(t) = \mathbf{x}_h(t) + \int_0^t \mathbf{H}(t-\tau)\mathbf{f}(\tau)d\tau, \tag{2.7}$$

where **x** is the total forced response, $\mathbf{x}_h$ is the homogeneous solution, **f** is the excitation vector, and these vectors are partitioned according to the previously defined sets of DOF, e.g.

$$\mathbf{x}(t) = \begin{Bmatrix} \mathbf{x}_i(t) \\ \mathbf{x}_c(t) \\ \mathbf{x}_m(t) \\ \mathbf{x}_b(t) \end{Bmatrix} \qquad \mathbf{f}(t) = \begin{Bmatrix} \mathbf{f}_i(t) \\ \mathbf{f}_c(t) \\ \mathbf{f}_m(t) \\ \mathbf{f}_b(t) \end{Bmatrix} \tag{2.8}$$

This partitioning is implicit in all matrix equations which follow, unless otherwise indicated. The matrix **H** is the impulse response function (IRF) matrix, any element of which can be written as,

$$H_{ij}(t) = \sum_{p=1}^{r} \phi_i^p \phi_j^p t + \sum_{p=r+1}^{n} \frac{\phi_i^p \phi_j^p}{\omega_{dp}} e^{-\varsigma_p \omega_p t} \sin\left(\omega_{dp} t\right), \tag{2.9}$$

where ϕ_i^p is the ith element of the pth eigenvector of the substructure prior to synthesis, ω_p and ω_{dp} are the pth undamped and damped natural frequencies, respectively, ζ_p is the pth modal damping ratio, r is the number of rigid body modes, and $n \leq N$ is the number of modes required for convergence. The number of elastic modes is n-r. Note that the IRF matrix **H** contains elements from all substructures involved in the synthesis, and is partitioned as described above.

We can decompose each excitation component into an externally applied portion and a component due to synthesis, as follows. The i-set DOF are by definition subject only to externally applied excitations, so

$$\mathbf{f}_i(t) = \mathbf{f}_i^e(t) \tag{2.10}$$

where the superscript "e" indicates an externally applied excitation. The c-set DOF are subject to externally applied excitations as well as coupling reactions hence,

$$\mathbf{f}_c(t) = \mathbf{f}_c^e(t) - \mathbf{R}\tilde{\mathbf{f}}_c^*(t) \tag{2.11}$$

where the ~ overstrike indicates the coupling reaction, **R** is a Boolean matrix reflecting the equilibrium which exists between the coupled DOF [4], and the * superscript indicates a synthesized (unknown) quantity. The m-set DOF are subject to externally applied excitations as well as reactions due to the modifications hence

$$\mathbf{f}_m(t) = \mathbf{f}_m^e(t) - \mathbf{f}_m\left(\mathbf{x}_m^*(t), \dot{\mathbf{x}}_m^*(t)\right) = \mathbf{f}_m^e(t) - \mathbf{f}_m^*(t) \tag{2.12}$$

where the reactions due to the modifications are an arbitrary nonlinear function $\mathbf{f}_m^*(t)$ of the synthesized displacements and velocities (accelerations and time). The b-set DOF are subject to externally applied excitations as well as excitations due to prescribed base motion **y** acting through an arbitrarily nonlinear structural element, typically involving displacement- and velocity-dependent forces only, i.e.

$$\mathbf{f}_b(t) = \mathbf{f}_b^e(t) - \mathbf{f}_b\left(\mathbf{x}_b^*(t) - \mathbf{y}(t), \dot{\mathbf{x}}_b^*(t) - \dot{\mathbf{y}}(t)\right) = \mathbf{f}_b^e(t) - \mathbf{f}_b^*(t) \tag{2.13}$$

Therefore, the governing equation for synthesis is,

$$\boxed{\mathbf{x}^*(t) = \mathbf{x}(t) - \int_0^t \mathbf{H}(t-\tau)\mathbf{f}^*\left(t, \tau, \mathbf{x}^*(\tau), \dot{\mathbf{x}}^*(\tau)\right)d\tau \tag{2.14}}$$

where **x**(t) contains both the initial displacement and response due to externally applied excitations,

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \mathbf{H}(t-\tau)\mathbf{f}^e(\tau)d\tau \tag{2.15}$$

and **f***(t) are the synthesized reactions acting on all DOF sets,

$$\mathbf{f}^*(t) = \begin{Bmatrix} 0 \\ \mathbf{R}\tilde{\mathbf{f}}_c^*(t) \\ \mathbf{f}_m^*(t) \\ \mathbf{f}_b^*(t) \end{Bmatrix} \tag{2.16}$$

Equation (2.14) is a nonlinear Volterra integral equation of the second kind, and is the central equation of this work. Direct solution is possible for linear problems; for nonlinear problems iterative solutions are required, and these exploit the contractive nature of the integral operators in achieving excellent convergence properties, as will be demonstrated.

2.6 *Solution of Governing Equation for Synthesis: Uniqueness and Convergence*

In order to investigate issues of solution uniqueness and convergence of Eq. (2.14), the method of Picard Iteration will be used [9]. A few preliminary concepts and definitions from analysis are included, without proof, for completeness. Proofs may be found in [10, Ch. 1].

We define the following metric on the space of continuous vector functions $C^{(n)}[0,t']$ as:

$$d(\mathbf{x}(t),\mathbf{y}(t)) = \max_{t\in[0,t']} \|\mathbf{x}(t) - \mathbf{y}(t)\|_\infty \equiv \|\mathbf{x} - \mathbf{y}\| \tag{2.17}$$

where $\mathbf{x}(t), \mathbf{y}(t) \in C^{(n)}[0,t']$, hence defining a metric space $C \equiv (C^{(n)},d)$. A (scalar) sequence $(\mathbf{x}_n)$ in the metric space $(C^{(n)},d)$ is *convergent* if there is an $\mathbf{x} \in C^{(n)}$ such that $\lim_{n\to\infty} d(\mathbf{x}_n,\mathbf{x}) = 0$. Then, $\mathbf{x}$ is the limit of $(\mathbf{x}_n)$ and $\lim_{n\to\infty} \mathbf{x}_n = \mathbf{x}$, or $\mathbf{x}_n \to \mathbf{x}$. A sequence $(\mathbf{x}_n)$ in the metric space $(C^{(n)},d)$ is *Cauchy*-convergent if for every $\varepsilon > 0$ there is an $N=N(\varepsilon)$ such that $d(\mathbf{x}_m,\mathbf{x}_n) < \varepsilon$ for every $m,n > N$. The metric space $C^{(n)}$ is *complete* if every Cauchy sequence in $C^{(n)}$ converges, i.e. has a limit which is an element of $C^{(n)}$. This distinct and important characteristic of completeness is needed as there exist spaces in which Cauchy sequences do not converge. Such spaces are said to be incomplete. It can be noted here that every convergent sequence in $C^{(n)}$ is a Cauchy sequence.

A *fixed point* of a mapping T: $C^{(n)} \rightarrow C^{(n)}$ of a set $C^{(n)}$ into itself is an $\mathbf{x} \in C^{(n)}$ which is mapped onto itself, $\mathbf{Tx} = \mathbf{x}$. Let $C = (C^{(n)}, d)$ be a metric space. A mapping T: $C^{(n)} \rightarrow C^{(n)}$ is a *contraction* on $C^{(n)}$ if there is a positive real number $\alpha < 1$ such that for all $\mathbf{x},\mathbf{y} \in C^{(n)}$,

$$d(\mathbf{Tx},\mathbf{Ty}) \le \alpha d(\mathbf{x},\mathbf{y}) \tag{2.18}$$

Central to the effort of demonstrating the uniqueness of a solution to Eq. (2.14) is the Banach Fixed Point Theorem. Consider again the non-empty and complete metric space C = $(C^{(n)}, d)$ and let T: $C^{(n)} \rightarrow C^{(n)}$ be a contraction on $C^{(n)}$. Then T has precisely one fixed point [10, Sec. 5.1].

We now examine the nonlinear Volterra integral equation,

$$\mathbf{x}^*(t) = \mathbf{x}(t) - \int_0^t \mathbf{H}(t-\tau)\mathbf{f}\left(t,\tau,\mathbf{x}^*(\tau)\right)d\tau \tag{2.19}$$

where $\mathbf{x}(t)$ is bounded, $m_1 < \|\mathbf{x}(t)\| < m_2$, and continuous on [0,t'] as it is the solution of a linear structural dynamics problem, and $\mathbf{f}\left(t,\tau,\mathbf{x}^*(\tau)\right)$ is assumed to be continuous (for now), and hence bounded on the rectangle $t \in [0,t']$, $\tau \in [0,t']$. We also assume that $\left\|\mathbf{F}\left(t,\tau,\mathbf{x}^*(\tau)\right)\right\| \le M$ for bounded $\mathbf{x}^*(\tau)$, i.e. $x_l < \|\mathbf{x}^*(t)\| < x_u$. We note that $\mathbf{x}^*(\tau)$ can be reasonably assumed bounded as it is the (unknown) solution to a locally nonlinear structural dynamics problem where the boundedness of the response is a goal of design. Note also that $\mathbf{f}\left(t,\tau,\mathbf{x}^*(\tau)\right)$ represents the action of some passive, semi-active, or active structural/mechanical element, and hence its boundedness is expected due to physical limitations.

Following [10], consider the mapping defined by the successive approximation (Picard Iteration) of Eq. (2.19),

$$\mathbf{x}^*_{n+1}(t) = \mathbf{x}(t) - \int_0^t \mathbf{H}(t-\tau)\mathbf{f}\left(t,\tau,\mathbf{x}^*_n(\tau),\dot{\mathbf{x}}^*_n(\tau)\right)d\tau \equiv T\left(\mathbf{x}^*_n(\tau),\dot{\mathbf{x}}^*_n(\tau)\right) \tag{2.20}$$

defined on the space of continuous functions, $C^{(n)}[0,t']$. We examine the following elements from a (bounded) sequence

$$\ldots \mathbf{x}^*_{n-1}(t), \mathbf{x}^*_n(t), \mathbf{x}^*_{n+1}(t)\ldots$$

along with the metric, Eq. (2.17). Our first goal is to show that the mapping of Eq. (2.20) is a contraction, and to this end, the metric is used to measure the "distance" between inputs $\mathbf{x}^*_{n-1}(t)$ and $\mathbf{x}^*_n(t)$, and the respective outputs. This distance is,

$$d\left(\mathbf{x}^*_{n+1}(t), \mathbf{x}^*_n(t)\right) = d\left(T\left(\mathbf{x}^*_n(t), \dot{\mathbf{x}}^*_n(t)\right), T\left(\mathbf{x}^*_{n-1}(t), \dot{\mathbf{x}}^*_{n-1}(t)\right)\right) = \ldots$$

$$\ldots = \left\| \mathbf{x}(t) - \int_0^t \mathbf{H}(t-\tau)\mathbf{f}\left(t,\tau,\mathbf{x}^*_n(\tau),\dot{\mathbf{x}}^*_n(\tau)\right)d\tau - \left[\mathbf{x}(t) - \int_0^t \mathbf{H}(t-\tau)\mathbf{f}\left(t,\tau,\mathbf{x}^*_{n-1}(\tau),\dot{\mathbf{x}}^*_{n-1}(\tau)\right)d\tau\right]\right\| \quad (2.21)$$

or,

$$d\left(\mathbf{x}^*_{n+1}(t), \mathbf{x}^*_n(t)\right) = \left\| \int_0^t \mathbf{H}(t-\tau)\left\{\mathbf{f}\left(t,\tau,\mathbf{x}^*_n(\tau),\dot{\mathbf{x}}^*_n(\tau)\right) - \mathbf{f}\left(t,\tau,\mathbf{x}^*_{n-1}(\tau),\dot{\mathbf{x}}^*_{n-1}(\tau)\right)\right\}d\tau \right\| \quad (2.22)$$

or, letting

$$\mathbf{f}^*_n(t,\tau) = \mathbf{f}\left(t,\tau,\mathbf{x}^*_n(\tau),\dot{\mathbf{x}}^*_n(\tau)\right) - \mathbf{f}\left(t,\tau,\mathbf{x}^*_{n-1}(\tau),\dot{\mathbf{x}}^*_{n-1}(\tau)\right), \quad (2.23)$$

yields the following form of Eq. (2.21)

$$\left\|\mathbf{x}^*_{n+1}(t) - \mathbf{x}^*_n(t)\right\| = d\left(\mathbf{x}^*_{n+1}(t), \mathbf{x}^*_n(t)\right) = \left\| \int_0^t \mathbf{H}(t-\tau)\mathbf{f}^*_n(t,\tau)d\tau \right\| \quad (2.24)$$

In order to be able to relate this metric of the outputs to the corresponding metric of the inputs (and demonstrate the contractive property of the mapping Eq. (2.20)), the quantity $\left\|\mathbf{x}^*_n(\tau) - \mathbf{x}^*_{n-1}(\tau)\right\|$ must be isolated, which is not possible in the form of Eq. (2.24), as this quantity is operated on by the function **f**. To this end, we impose that **f** satisfy the Lipschitz condition,

$$\left\|\mathbf{f}_n^*(t,\tau)\right\| \le L\left\|\mathbf{x}_n^*(\tau) - \mathbf{x}_{n-1}^*(\tau)\right\| \tag{2.25}$$

for $\left(t,\tau,\mathbf{x}_n^*(\tau),\dot{\mathbf{x}}_n^*(\tau)\right)$ and $\left(t,\tau,\mathbf{x}_{n-1}^*(\tau),\dot{\mathbf{x}}_{n-1}^*(\tau)\right)$ in the domain of **f**, and where L is a positive constant. Note that the contraction, Eq. (2.18) is a Lipschitz condition with $L = \alpha < 1$ [11]. Imposing the Lipschitz condition on the integral of Eq. (2.24),

$$\left\|\mathbf{x}_{n+1}^*(t) - \mathbf{x}_n^*(t)\right\| \le L\int_0^t \|\mathbf{H}(t-\tau)\| \cdot \left\|\mathbf{x}_n^*(\tau) - \mathbf{x}_{n-1}^*(\tau)\right\| d\tau \tag{2.26}$$

We can evaluate Eq. for two members of the sequence, $\mathbf{x}_1^*(t)$ and $\mathbf{x}_2^*(t)$,

$$\left\|\mathbf{x}_3^*(t) - \mathbf{x}_2^*(t)\right\| \le L\int_0^t \|\mathbf{H}(t-\tau)\| \cdot \left\|\mathbf{x}_2^*(\tau) - \mathbf{x}_1^*(\tau)\right\| d\tau. \tag{2.27}$$

Making use of the bounds on the members of the sequence (see paragraph following Eq. (2.14)), we find,

$$\left\|\mathbf{x}_3^*(t) - \mathbf{x}_2^*(t)\right\| \le Lt\left|x_u - x_l\right| \cdot \|\mathbf{H}(t)\|. \tag{2.28}$$

Continuing in sequence,

$$\left\|\mathbf{x}_4^*(t) - \mathbf{x}_3^*(t)\right\| \le L\int_0^t \|\mathbf{H}(t-\tau)\| \cdot \left\|\mathbf{x}_3^*(\tau) - \mathbf{x}_2^*(\tau)\right\| d\tau \tag{2.29}$$

which becomes,

$$\left\|\mathbf{x}_4^*(t) - \mathbf{x}_3^*(t)\right\| \le \frac{1}{2}\left(Lt\|\mathbf{H}(t)\|\right)^2\left|x_u - x_l\right| \tag{2.30}$$

and,

$$\left\|\mathbf{x}_5^*(t) - \mathbf{x}_4^*(t)\right\| \le L\int_0^t \|\mathbf{H}(t-\tau)\| \cdot \left\|\mathbf{x}_4^*(\tau) - \mathbf{x}_3^*(\tau)\right\| d\tau \tag{2.31}$$

which becomes,

$$\left\|\mathbf{x}_5^*(t) - \mathbf{x}_4^*(t)\right\| \le L^3 \frac{t^3}{3!}\|\mathbf{H}(t)\|^3 |x_u - x_l| \tag{2.32}$$

and by induction the following can be established,

$$\left\|\mathbf{x}_{n+1}^*(t) - \mathbf{x}_n^*(t)\right\| \le \frac{|x_u - x_l|}{(n-1)!}\left(Lt\|\mathbf{H}(t)\|\right)^{n-1} \tag{2.33}$$

which establishes the uniform convergence of the series,

$$\sum_{n=1}^{\infty}\left\|\mathbf{x}_{n+1}^*(t) - \mathbf{x}_n^*(t)\right\| \tag{2.34}$$

with no restriction on L, t, or ||**H**(t)||. To show the convergence of the sequence $\mathbf{x}_n^*(t)$ of Eq. (2.20) to the solution $\mathbf{x}^*(t)$ of Eq. (4.5), we write $\mathbf{x}_n^*(t)$ using a telescoping series,

$$\mathbf{x}_n^*(t) = \mathbf{x}_1^*(t) + \sum_{j=1}^{n-1}\left(\mathbf{x}_{j+1}^*(t) - \mathbf{x}_j^*(t)\right). \tag{2.35}$$

Given the convergence of Eq. (2.34), taking the limit of Eq. (2.35) as $n \to \infty$ reveals that $\lim_{n\to\infty} \mathbf{x}_n^*(t)$ exists for all $t \in [0, t']$. This sequence is identical to the sequence generated by Eq. (2.20). We assumed that $\mathbf{f}\left(t, \tau, \mathbf{x}_n^*(t), \dot{\mathbf{x}}_n^*(t)\right)$ is continuous, and hence taking the limit as $n \to \infty$ on both sides of Eq. (2.20) leads to the conclusion that $\lim_{n\to\infty} \mathbf{x}_n^*(t) = \mathbf{x}^*(t)$, the solution to Eq. (2.14).

Although we have shown that the convergence of the successive approximation is not dependent on ||**H**(t)||, it is worthwhile to evaluate ||**H**(t)|| at this point. Defining the pth modal residue matrix,

$$\mathbf{R}^{(p)} = \boldsymbol{\phi}^{(p)}\left(\boldsymbol{\phi}^{(p)}\right)^T$$

we have $\mathbf{R} = \sum_{p=1}^{n} \mathbf{R}^{(p)} = \mathbf{M}^{-1}$. The pth modal impulse response function is,

$$h_p(t) = \frac{e^{-\varsigma\omega_p t}\sin(\omega_{dp} t)}{\omega_{dp}} \tag{2.36}$$

and hence any element of **H**, Eq. (2.9), can be written as,

$$H_{ij}(t) = \sum_{p=1}^{r} R_{ij}^{p}\ t + \sum_{p=r+1}^{n} R_{ij}^{p}\ h_p(t) \tag{2.37}$$

where r is the number of rigid body modes and n is the total number of modes. An upper bound for the pth modal residue is

$$\left|R_{ij}^{p}\right| \le \max_{i,j}\left(\mathbf{M}^{-1}\right),\ p=1,2,..,n$$

and for the modal impulse response,

$$\max_{0\le t\le t'}\left|h_p(t)\right| \le \frac{1}{\omega_{dp}},$$

and hence the norm of any element H_{ij} can be written as

$$\max_{0\le t\le t'}\left|H_{ij}(t)\right| \le \left(rt' + \sum_{p=r+1}^{n} \omega_{d_p}^{-1} \right)\cdot\left|R_{ij}^{p}\right|. \tag{2.38}$$

Given this result, we find that

$$\max_{0\le t\le t'}\left\|\mathbf{H}(t)\right\| \le J\left(rt' + \sum_{p=r+1}^{n} \omega_{d_p}^{-1} \right)\cdot\left|R_{ij}^{p}\right|, \tag{2.39}$$

where J is the number of columns in **H**.

3. RECURSIVE ALGORITHM 3: RECURSIVE BLOCK-BY-BLOCK CONVOLUTION

The algorithms RA1 and RA2 (Section 4) are similar in that they are all modal-based. In these algorithms, the recursion is obtained from the use of a modal transition matrix. Furthermore, it is shown that these algorithms, which are inherently unstable in their explicit forms, can be stabilized by reformulation of the algorithms in implicit forms, specifically as predictor-corrector forms. The stability achieved through the use of a single corrector step is only marginally effective, and hence we develop a different recursive approach for the solution of the governing nonlinear integral equation, Eq. (2.14). The solution method to be developed will exploit the exponential convergence result of Section 2.6 as the solution method is iterative. The algorithm will improve greatly upon the performance of the basic iterative algorithm reported in [3], by the development of a recursive, block-by-block convolution solution. In fact, the block-by-block convolution is ideally suited for the calculation of structural response for long time records, as will be demonstrated.

3.1 Numerical Quadrature for Nonlinear Volterra Integral Equations

The numerical solution of Eq. (2.14) typically starts with a discretization of the equation using some quadrature rule. For the response at some time $t = i\Delta t \equiv t_i$, $(t = 0 = 0\Delta t)$, Eq. (2.14) becomes,

$$\mathbf{x}^*(i\Delta t) = \mathbf{x}(i\Delta t) - (\Delta t)^\alpha \sum_{j=0}^{i-\beta} w_j \mathbf{H}((i-j)\Delta t)\mathbf{f}^*(j\Delta t) \tag{3.1}$$

where we have abbreviated the general nonlinear force as $\mathbf{f}^*$, α and β are real scalar constants depending on the quadrature rule chosen, and the w_j are the quadrature weights. For example, if we consider the simplest of quadrature rules, the rectangular rule (for a purpose to be made clear below), $\alpha = 1$, $\beta = 1$, and $w_j=1$. For i = 0,1,2, Eq. (3.1) becomes ,

$$\mathbf{x}^*(0\Delta t) = \mathbf{x}(0\Delta t) \tag{3.2}$$

$$\mathbf{x}^*(1\Delta t) = \mathbf{x}(1\Delta t) - \Delta t\left[\mathbf{H}(1\Delta t)\mathbf{f}^*(0\Delta t) + \mathbf{H}(0\Delta t)\mathbf{f}^*(1\Delta t)\right] \tag{3.3}$$

$$\mathbf{x}^*(2\Delta t) = \mathbf{x}(2\Delta t) - \Delta t\left[\mathbf{H}(2\Delta t)\mathbf{f}^*(0\Delta t) + \mathbf{H}(1\Delta t)\mathbf{f}^*(1\Delta t) + \mathbf{H}(0\Delta t)\mathbf{f}^*(2\Delta t)\right] \tag{3.4}$$

and we note that **H**(t=0) = **0**, yielding the correct series for the rectangular rule. These equations can be rewritten using a simplified indexing scheme, i.e. **x**(iΔt) = **x**(i+1), which corresponds to the indexing required for computer programming, e.g. **x**(0Δt) = **x**(1). Using this indexing, Eqs.(3.2), (3.3), and (3.4) become, for i = 1,2,3,

$$\mathbf{x}*(1) = \mathbf{x}(1) \tag{3.5}$$

$$\mathbf{x}*(2) = \mathbf{x}(2) - \Delta t\left[\mathbf{H}(2)\mathbf{f}*(1) + \mathbf{H}(1)\mathbf{f}*(2)\right] \tag{3.6}$$

$$\mathbf{x}*(3) = \mathbf{x}(3) - \Delta t\left[\mathbf{H}(3)\mathbf{f}*(1) + \mathbf{H}(2)\mathbf{f}*(2) + \mathbf{H}(1)\mathbf{f}*(3)\right] \tag{3.7}$$

where again, **H**(1) = **0**. As will be seen below, the bracketed terms in Eqs.(3.5), (3.6), and (3.7) are available from the discrete convolution.

The trapezoid rule and Simpson's rule are more commonly used quadratures for this application [9,12]. The performance of the trapezoid rule in the solution of the linear synthesis problem was reported by Gordis [2].

3.2 *Discrete Convolution and Filter Matrices*

We define the basic convolution in order to establish a notation for the development of the block-by-block convolution which follows. The convolution of two vectors **x** and **y** is denoted as **x*y**. The discrete convolution of **x** and **y** is given by

$$\mathbf{x}*\mathbf{y}=\sum_{k} x(n-k)y(k) \tag{3.8}$$

If **x** and **y** are each (n x 1), e.g.

$$\mathbf{x}=\begin{pmatrix}x_1 & x_2 & \cdots & \cdots & x_{n-1} & x_n\end{pmatrix}^T$$
$$\mathbf{y}=\begin{pmatrix}y_1 & y_2 & \cdots & \cdots & y_{n-1} & y_n\end{pmatrix}^T$$

then the convolution **x*y** can be written as the following matrix-vector product, where the matrix is Toeplitz, constant diagonal, and is referred to as a filter matrix **h(x)** [13]:

$$\mathbf{z}=\mathbf{x}*\mathbf{y}=\mathbf{h}(\mathbf{x})\cdot\mathbf{y}=\begin{bmatrix} x_1 & 0 & \cdots & & \cdots & 0 \\ x_2 & x_1 & 0 & \cdots & \cdots & \vdots \\ \vdots & x_2 & x_1 & 0 & \cdots & \\ \vdots & \vdots & & \ddots & \ddots & \vdots \\ x_{n-1} & x_{n-2} & \cdots & \ddots & x_1 & 0 \\ x_n & x_{n-1} & x_{n-2} & \cdots & x_2 & x_1 \end{bmatrix}\begin{Bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{n-1} \\ y_n \end{Bmatrix} \tag{3.9}$$

For example, if **x** and **y** are both 3 * 1, we have

$$\mathbf{x}*\mathbf{y}=\begin{bmatrix} x_1 & 0 & 0 \\ x_2 & x_1 & 0 \\ x_3 & x_2 & x_1 \end{bmatrix}\begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}=\mathbf{h}(\mathbf{x})\cdot\mathbf{y}=\begin{Bmatrix} x_1y_1 \\ x_2y_1+x_1y_2 \\ x_3y_1+x_2y_2+x_1y_3 \end{Bmatrix} \tag{3.10}$$

where the filter matrix of the vector **x** is

$$\mathbf{h}(\mathbf{x})=\begin{bmatrix} x_1 & 0 & 0 \\ x_2 & x_1 & 0 \\ x_3 & x_2 & x_1 \end{bmatrix} \tag{3.11}$$

and the elements of **x** are referred to as filter weights [13]. Note that here we refer to **h** as an arbitrary filter matrix, which should not cause confusion with the use of the symbol **H** to refer to the impulse response function (IRF) matrix, as the IRF matrix is a filter matrix as well.

From a comparison of Eqs. (3.5), (3.6), (3.7) with Eq. (3.10), it is clear that the discrete convolution is equivalent to the use of the rectangular rule for numerically integrating a convolution integral.

We now define a delay matrix **D** [13] with the following structure:

$$\mathbf{D} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & 0 & 0 & 0 & \cdot \\ \cdot & 1 & 0 & 0 & 0 & \cdot \\ \cdot & 0 & 1 & 0 & 0 & \cdot \\ \cdot & 0 & 0 & 1 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \tag{3.12}$$

where the dimension of **D** is consistent with the length of the vector on which it operates. The matrix **D** produces a delay in time by one sample. For example, consider the 3 by 1 vector **x**,

$$\mathbf{D}\cdot\mathbf{x} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ x_1 \\ x_2 \end{Bmatrix} \tag{3.13}$$

where the product **Dx** is equivalent to the vector **x** shifted forward in time (delayed) by one sample. We can introduce delays of an arbitrary number of samples as $\mathbf{D}^k$. The product $\mathbf{D}^k\mathbf{x}$ produces a vector equivalent to the vector **x** but delayed by k samples.

The filter matrix **h** is equal to the summation of powers of the delay matrix multiplied by the filter weights, x_i. Alternatively, the columns of the filter matrix **h** are each products of powers of the delay matrix **D** and the vector **x**, i.e. the kth column of **h** is given by $\mathbf{D}^k\mathbf{x}$. The filter matrix of a vector **x** of length n is therefore,

$$\mathbf{h}(\mathbf{x}) = \sum_{k-0}^{n-1} x_k \cdot \mathbf{D}^k = \begin{bmatrix} \mathbf{D}^0\mathbf{x} & \mathbf{D}^1\mathbf{x} & \cdots & \mathbf{D}^{n-2}\mathbf{x} & \mathbf{D}^{n-1}\mathbf{x} \end{bmatrix} \tag{3.14}$$

3.3 *Block-by-Block Convolution*

We now develop the block-by-block (BBB) convolution of two vectors, **x** and **y**, i.e. **x*y**. We subdivide the entire time record of duration T seconds, consisting of N sample points (Δt = T/N) into a number of equally sized blocks, or subintervals, i.e. each subinterval contains the same number of sample points. We will subdivide the entire record into "K" blocks, where each block consists of J = N/K samples, and the duration of each block is JΔt seconds.

It is important to emphasize that there is a delay of J samples between blocks. For the purpose of developing the BBB algorithm, we will need to extract those rows of a vector corresponding to a particular block. To this end, we define the following row extraction matrix **r**:

$$\mathbf{r} = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & & & \\ \vdots & & & 0 & 1 & 0 & & & \\ & & & & 0 & 1 & 0 & & \\ \vdots & & & & & 0 & 1 & 0 & \\ 0 & \cdots & & & & & \cdots & 0 & 1 \end{bmatrix} \tag{3.15}$$

The product of the matrix **r** with a vector **x** is the subvector of **x** consisting of the rows (samples) of the K^th^ (i.e. last) block, i.e. $\mathbf{r} \cdot \mathbf{x} = \mathbf{x}_K$ where

$$\mathbf{x}_K = \begin{bmatrix} x_{J(K-1)+1} & \cdots & x_{n-1} & x_n \end{bmatrix}^T$$

Using the delay matrix **D**, we can define a matrix which extracts the rows of the k^th^ block, where k = 1,2,…,K.

$$\mathbf{r}_k = \mathbf{r} \cdot \mathbf{D}^k = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & & & & \cdots & 0 \\ \vdots & & & 0 & 1 & 0 & & & & & \vdots \\ & & & & 0 & 1 & 0 & & & & \\ \vdots & & & & & 0 & 1 & 0 & & & \vdots \\ 0 & \cdots & & & & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \tag{3.16}$$

The matrix equation, Eq. (3.9), can be written in a block-partitioned form as follows. We can write the kth subvector of **z**, i.e. $\mathbf{z}_k$, as,

$$\mathbf{z}_k = \sum_{m=1}^{k} \mathbf{r}_k \cdot \mathbf{h}(\mathbf{x}) \, \mathbf{r}_m^T \cdot \mathbf{y}_m \tag{3.17}$$

where $\mathbf{y}_m = \mathbf{r}_m \cdot \mathbf{y}$. It is important to note that the block filter matrix $\mathbf{r}_k \cdot \mathbf{h}(\mathbf{x}) \cdot \mathbf{r}_m^T$ need never be formed, as the following relations hold:

$$\mathbf{r}_k \cdot \mathbf{h}(\mathbf{x}) \cdot \mathbf{r}_m^T = \begin{cases} x(1{:}J) & \text{if } k = m \\ x\big(((k-m+1)J-(2J-2)){:}(k-m+1)J\big) & \text{if } k > m \end{cases} \tag{3.18}$$

where x(p:q) indicates the subvector of x consisting of elements p through q.

3.4 Performance Comparison - Standard and Block-by-Block Convolution

A traditional (single-block) convolution, for sufficiently long records of length n, is most efficiently computed using the FFT, yielding a total number of floating point operations (FLOPS) proportional to $n*\log_2(n)$. The computing language, MATLAB provides a built-in function for convolution which uses FIR filters for the calculation, and yields total FLOPS proportional to n^2. As we are here interested in comparing the performance of the BBB algorithm with the traditional single-block convolution, the use of the MATLAB function will provide much convenience with no loss in the ability to compare algorithms. Of course, if one were to optimize a convolution algorithm for large n, the FFT approach would be best.

Assuming one iteration for each diagonal block, the number of FLOPS for the BBB algorithm is given by:

$$\text{FLOPS} \approx K\left(2J^2 - J\right) + \frac{1}{2}\left(K^2 - K\right)\left(4J^2 - 4J + 1\right) \tag{3.19}$$

which yields an optimum number of blocks,

$$K_{opt} = \frac{1}{2} + 2N \tag{3.20}$$

which yields a solution which is clearly not realizable. The optimal number of blocks calculated is greater than the total number of samples N, and is a non-integer number of blocks. What is useful about this solution is that is indicates that the FLOPS required by a BBB convolution decreases monotonically with increasing block number. This is shown in Figure 2, which compares the FLOPS required by a standard convolution to the BBB convolution for varying total number of samples, N, and for different numbers of blocks.

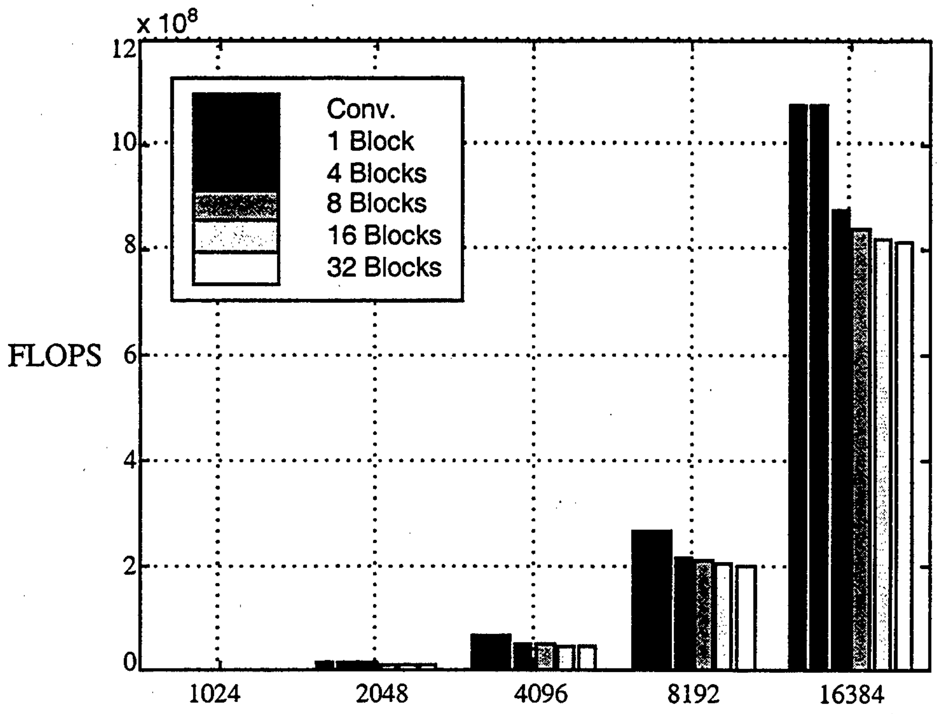

Figure 2. FLOPS required versus convolution vector for varying number of blocks

However, if we compare actual compute time (using MATLAB), we see that there is a point at which increasing the number of blocks results in increased compute times, as the computing "overhead" associated with increased block numbers outweighs the decrease in computing time due to the reduction in FLOPS required. This is shown in Figure 3.

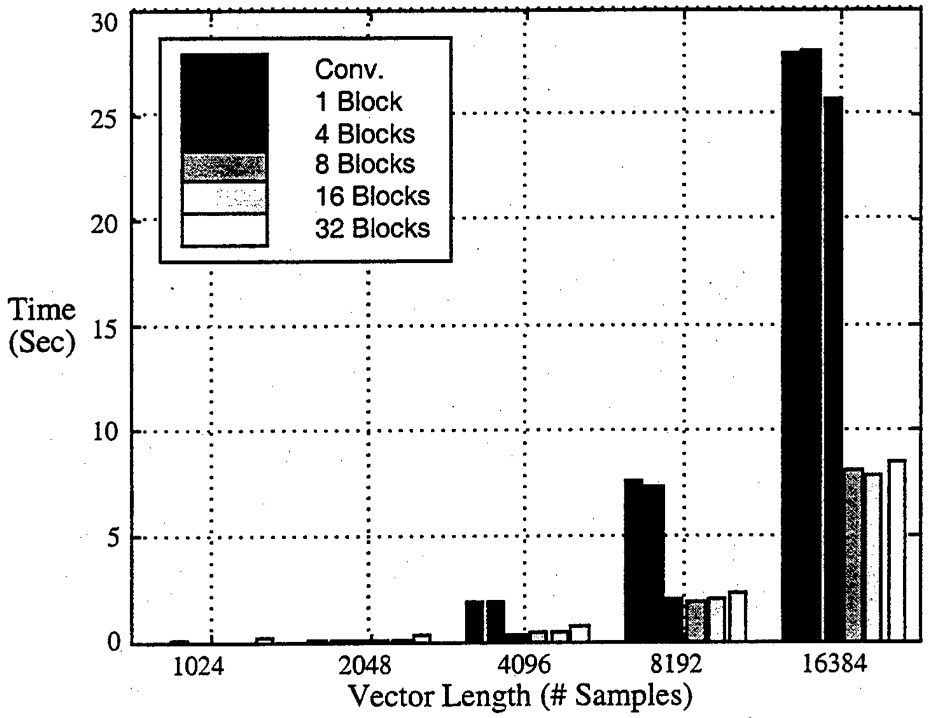

Figure 3.Computer time (sec) required versus convolution size for varying number of blocks

3.5 Recursive Block-by-Block Iteration Solution

Before discussing the recursive block-by-block iteration, we outline the basic iteration algorithm for the solution of Eq. (2.14). In the algorithm which follows, it is implied that the vector $\mathbf{x}$ is partitioned consistently with Eq. (2.16), with the alteration that the partition associated with the "i" coordinates has been deleted. Only those coordinates $\mathbf{x}^*$ directly involved in the synthesis, i.e. those coordinates subjected to forces of synthesis, are included in the iteration. The "i" set coordinate responses are calculated by a direct convolution of the associated IRF with the (converged) forces of synthesis, which result from the iteration. The coordinate set involved in the synthesis is the defined by the set union

$$s = m \cup c \cup b \tag{3.21}$$

where s denotes the synthesis set. The IRF matrix is therefore more fully denoted as $\mathbf{H}_{ss}(t)$.

For clarity of presentation, the time dependence and asterisk (•)* indicating a synthesized quantity will be dropped.

Basic Iteration :

- Initialize:
  $j \Leftarrow 1$
  $\mathbf{f}_s^j \Leftarrow \mathbf{1}$
- While $\mathbf{x}_s^{j+1} \neq \mathbf{x}_s^j$
  - $\mathbf{x}_s^{j+1} \Leftarrow \mathbf{x}_s - \mathbf{H}_{ss} * \mathbf{f}\left(\mathbf{x}_s^j, \dot{\mathbf{x}}_s^j, \mathbf{y}\right)$
  - $j \Leftarrow j+1$
- Converged forces of synthesis:
  $\mathbf{f}_s^* \Leftarrow \mathbf{f}\left(\mathbf{x}_s^j, \dot{\mathbf{x}}_s^j, \mathbf{y}\right)$
- Solution for i-set responses:
  $\mathbf{x}_i^* = \mathbf{x}_i - \mathbf{H}_{is} * \mathbf{f}_s^*$

We will now expand this algorithm to incorporate the recursive, block-by-block approach. The algorithm is recursive in that the iteration performed for block "k" makes use of the already converged forces of synthesis $\mathbf{f}^*$ for prior-time blocks k-1, k-2, etc, where for the sake of clarity, the "s" subscipt has been dropped. As will be described, only those forces of synthesis at the current block are included in the iteration, as prior block synthesis forces are converged. We will denote the responses and forces for the kth block, and at the jth iteration, as $\mathbf{x}_k^j$ and $\mathbf{f}_k^j$. The IRF filter matrix relating the kth response block and the mth input block is denoted as $\mathbf{H}_{km}$, and is given by

$$\mathbf{H}_{km} = \mathbf{r}_k \cdot \mathbf{H} \cdot \mathbf{r}_m^T \tag{3.22}$$

There are K blocks, k = 1,2,…,K, and each block is of length J (samples). We will make use of Eq. (3.22) to symbolically denote the IRF matrix blocks, while keeping in mind that in practice these matrices need never be formed. What is formed in practice are the partitioned vectors from which these IRF blocks are constructed, as given by Eq. (2.1). The iteration for the kth block is given by,

$$\mathbf{x}_k^{j+1} = \mathbf{x}_k - \sum_{m=1}^{k-1}\left(\mathbf{H}_{km}\cdot\mathbf{f}_m\right) - \mathbf{H}_{kk}\cdot\mathbf{f}_k^j \tag{3.23}$$

We can now construct the algorithm.

Recursive Block-by-Block Convolution Algorithm

```
• Initialize:
  $j \Leftarrow 1$
  $\mathbf{f}^j \Leftarrow 1$ (over all blocks)
• Do k = 1 : K
    • While $\mathbf{x}_k^{j+1} \neq \mathbf{x}_k^j$
        • $\mathbf{x}_k^{j+1} \Leftarrow \mathbf{x}_k - \sum_{m=1}^{k-1}\left(\mathbf{H}_{km}\cdot\mathbf{f}_m^*\right) - \mathbf{H}_{kk}\cdot\mathbf{f}_k^j$
        • $\mathbf{f}_k^{j+1} \Leftarrow \mathbf{f}\left(\mathbf{x}_k^{j+1}, \dot{\mathbf{x}}_k^{j+1}, \mathbf{y}\right)$
        • $j \Leftarrow j+1$
    • End While
    • Converged forces of synthesis:
      $\mathbf{f}_k \Leftarrow \mathbf{f}_k^j$
• End Do
• Solution for i-set responses:
  $\mathbf{x}_i^* = \mathbf{x}_i - \mathbf{H}_{is} * \mathbf{f}_s^*$
```

3.6 *Example of Block-by-Block Synthesis*

The following example is taken from [3], in order to demonstrate the improvement in performance of the current block-by-block algorithm relative to the basic iteration algorithm reported in [3].

This example demonstrates the use of the nonlinear synthesis procedure in a nonlinear base isolation problem subjected to a prescribed transient base displacement excitation. An idealized "deck" model is isolated by four nonlinear springs/dampers located at the four corner nodes, as shown in Figure 4. A piece of equipment is mounted elastically on the deck, modeled using a single lumped mass and linear spring. The deck is modeled using four-noded quadrilateral elements. The deck is a square steel plate, 10 inch on a side, and the plate thickness is 0.125 inch. The lumped mass is taken as 5.5% of the total plate mass. The linear spring stiffness is 1000 lbf/in.

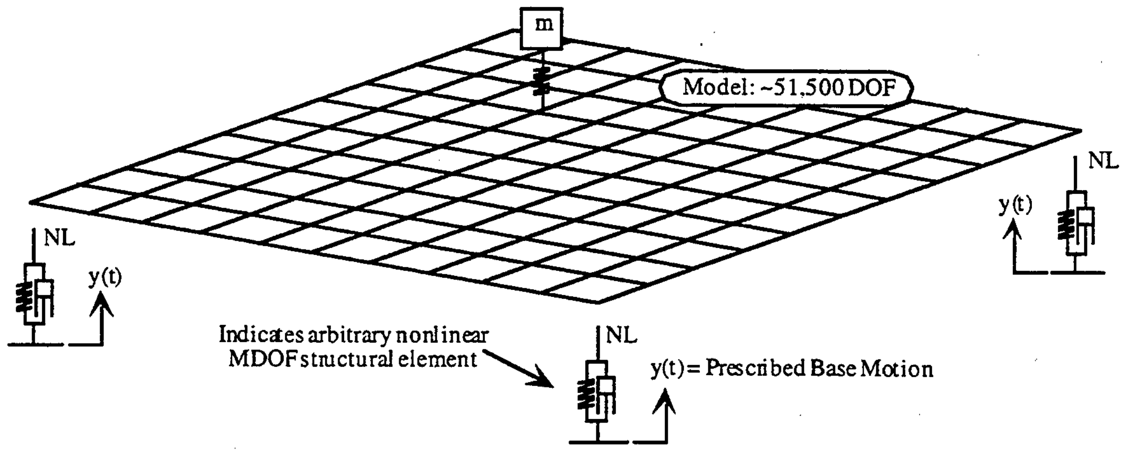
m
Model: ~51,500 DOF
NL
y(t)
NL
y(t)
Indicates arbitrary nonlinear MDOF structural element
NL
y(t) = Prescibed Base Motion

Figure 4. Isolated deck with equipment (lumped mass) mounted with linear spring

The excitation used is a prescribed "blast" base motion, shown in Figure 5.

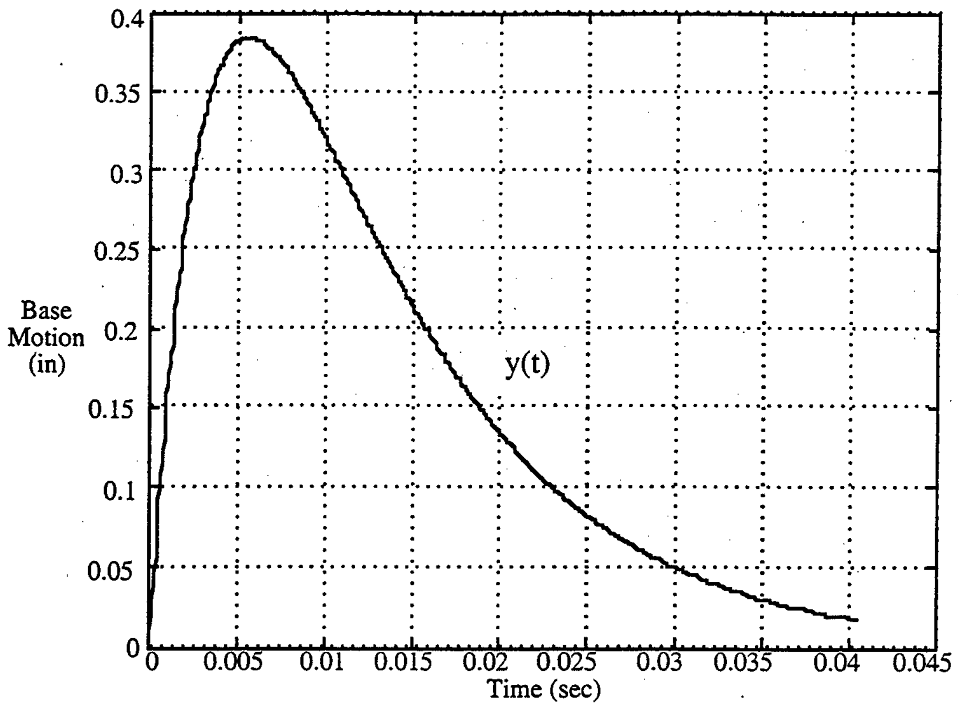

Figure 5. Transient base motion

We now present the results obtained from the current recursive block-by-block algorithm. In this example, all modes up to 12KHz were calculated (99 modes). The synthesis was performed using all modes up to 4KHz. The isolators used in this example are described by the following equation:

$$f(x-y,\dot{x}-\dot{y}) = k(x-y) + k_3(x-y)^3 + c_2(\dot{x}-\dot{y})^2 \tag{3.24}$$

where k is the linear stiffness coefficient, k_3 is the cubic stiffness coefficient, and c_2 is the quadratic damping coefficient. In the example problem, the isolator parameters have the following values:

k = 100 lbf/in	k_3 = 20 lbf/in^3	C_2 = 0.001 lbf-sec/in^2

The synthesis was performed using the following parameters:

Sample length:	5.0e-5 sec	Modes Retained:	38
Number of Subintervals:	8	Max mode frequency:	3,702.9 Hz
Samples/subinterval:	101	Modal Damping:	0.0
End time:	0.04 sec		

The Direct FE solution used the same sample length and end time as the synthesis.

In Figure 6, the vertical displacement response of a corner node of the plate model is shown, along with the corresponding response quantity as calculated using the commercial Direct FE solution. Keeping in mind the relative times required, 7 min 54 sec for the synthesis (including the solution for 99 modes up to 12KHz), versus 30 min 15 sec for the Direct FE solution, its clear that the synthesis provides a very accurate solution.

The corresponding comparison of the velocity for the same node is shown in Figure 7. While there is some disparity in the two solutions for velocity at early times, these differences do not compromise the solution after the early times. This is due to the fact that the convergence of the numerical method is relatively insensitive to errors in the starting values; the effect of the starting errors on convergence is attenuated by the factor Δt [9].

In Figure 8, the vertical displacement response of the lumped mass is compared. It should be noted that the synthesis solution for the response of the lumped mass is a straight convolution using the nonlinear forces which are the direct result of the synthesis. The isolated mass is a model DOF not subjected to forces of synthesis (i.e. nonlinear isolator forces), and hence this DOF is a member of the coordinate set "i" rather than the "c" coordinate set.

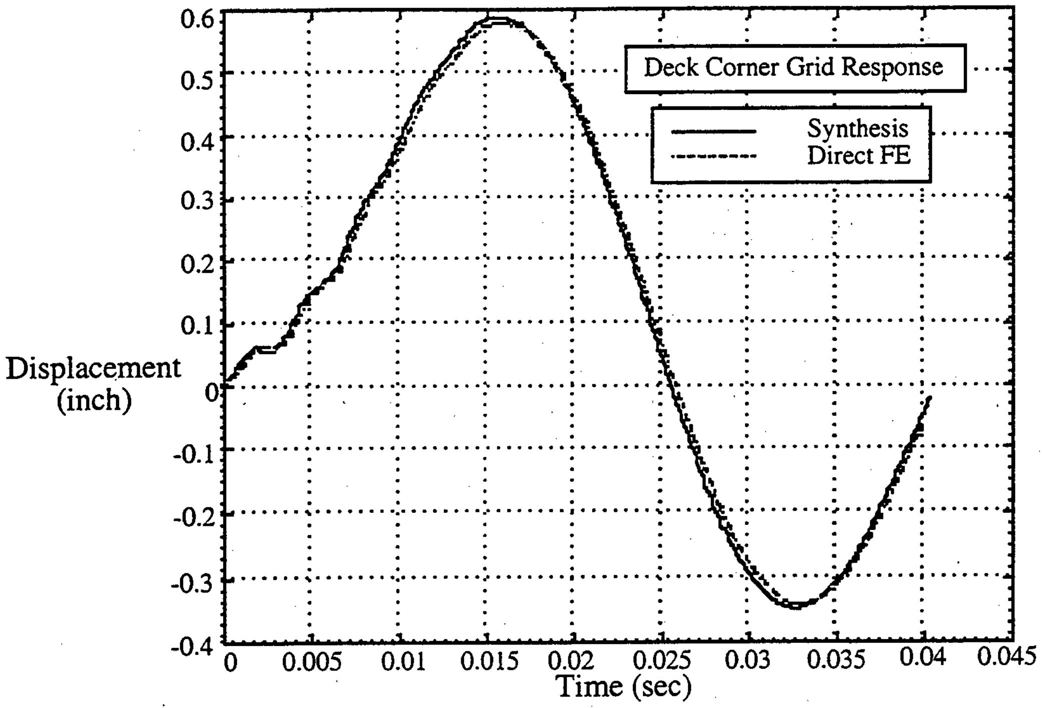

Figure 6. Deck corner vertical displacement response

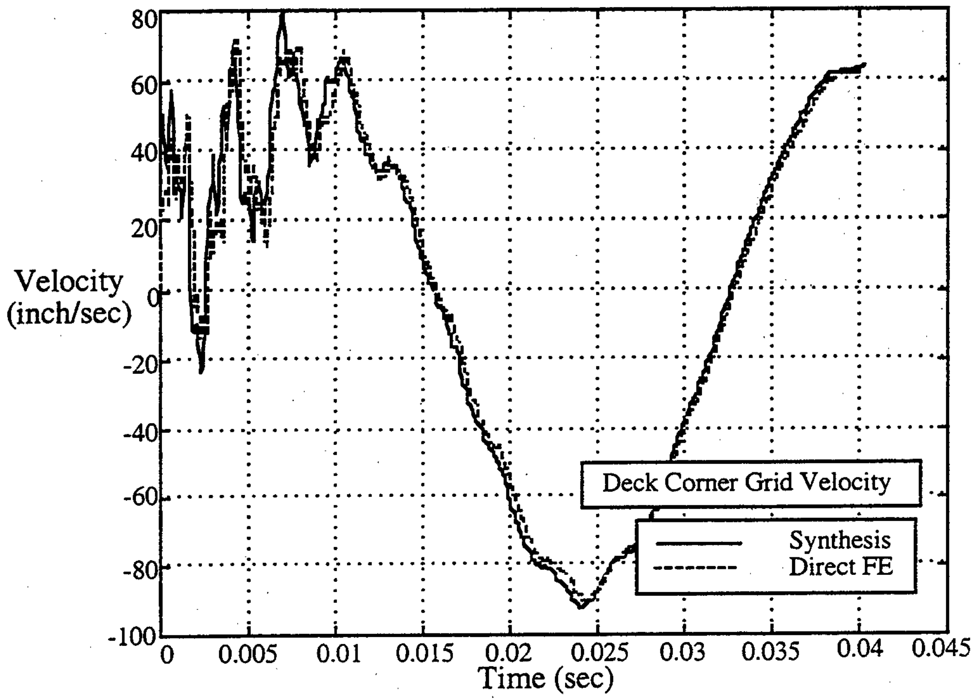

Figure 7. Deck corner vertical velocity response

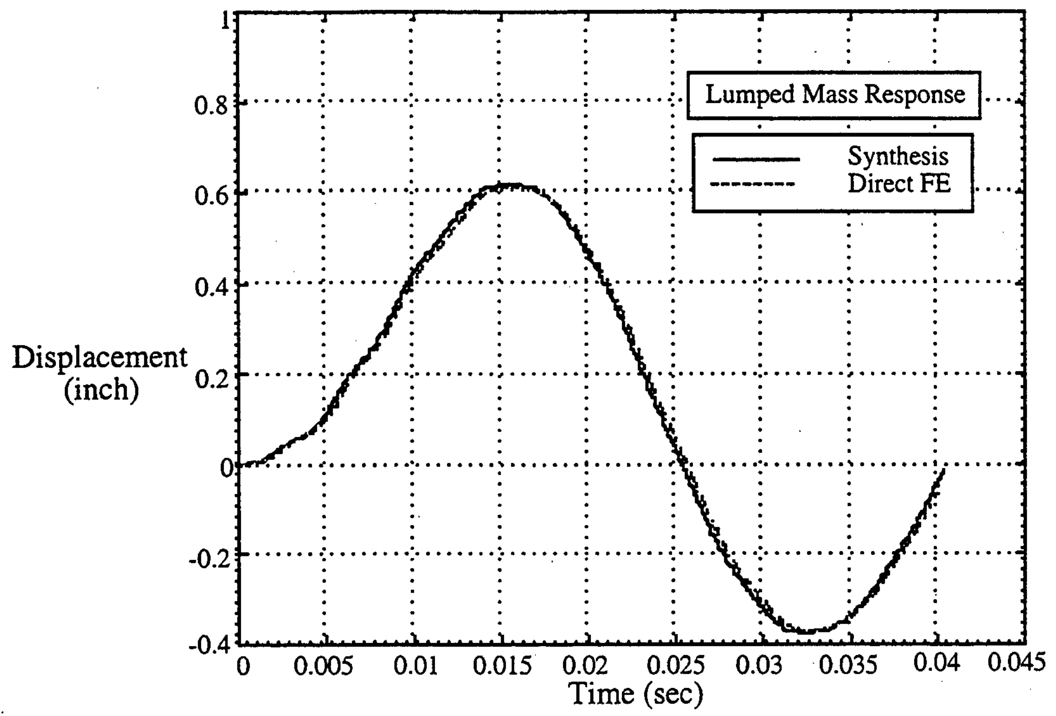

Figure 8. Isolated lumped mass vertical displacement response

3.6.1 Comparison of Computing Times Required: Synthesis vs. Direct FE

In the tables which follow, a comparison is made of the computing times required for a complete transient analysis as performed using the synthesis and using commercial nonlinear direct transient analysis. The direct transient analysis does not use the modes, and hence the total time for the direct FE solution does not include a modal solution. The synthesis does, however, require a modes database for each substructure, and this solution is performed using the commercial FE program. The time required for a single complete transient analysis using synthesis therefore includes the time for the modes solution and for the synthesis itself. As is seen from the tables, all subsequent analyses require only the synthesis itself, and hence an enormous savings in time is realized, as compared with the direct FE solution, which must be repeated in its entirety for each analysis. All of the calculation times reported in each table were generated using a common computer which ran both the commercial FE program and the previously reported synthesis algorithm, written in MATLAB.

Table 1. **Summary of Solution Times Required: Recursive Block-by-Block Algorithm**

	Computing Time Required	
Calculation Performed	**Using Synthesis**	***Using Direct FE***
Normal Modes: (All modes to 12KHz)	7 min 47 sec	*not required*
Transient Response:	0 min 7 sec	30 min 15 sec
Total Time Required - Single Analysis	7 min 54 sec	30 min 15 sec
Total Time Required - Subsequent Analyses	0 min 7 sec	30 min 15 sec

The following table, Table 2, taken from Reference [3], shows the comparison of the previously reported algorithm described in [3] with a commercial direct nonlinear transient FE solution. This table is included here to demonstrate the improved performance of the new recursive block-by-block algorithm as compared with the algorithm of Reference [3]. As can be seen from the times reported in Table 2, for a single analysis, the original algorithm required approximately 60% less computer time as compared with the commercial nonlinear direct transient analysis. The new algorithm requires approximately 74% less computer time than the commercial nonlinear direct transient analysis. However, the improvement in computing time for the synthesis alone is dramatic; the new algorithm requires approximately 7 seconds to perform each subsequent nonlinear transient analysis. This fast reanalysis capability will facilitate the use of numerical optimization techniques for nonlinear isolation design.

Table 2. Summary of Solution Times Required: Previously Reported Algorithm Ref. [3]

	Computing Time Required	
Calculation Performed	**Using Synthesis**	***Using Direct FE***
Normal Modes: (All modes to 12KHz)	13 min 24 sec	*not required*
Transient Response:	4 min 15 sec	43 min 41 sec
Total Time Required -Single Analysis	17 min 39 sec	43 min 41 sec
Total Time Required - Subsequent Analyses	4 min 15 sec	43 min 41 sec

3.7 Summary of Recursive Algorithm 3: Block-by-Block Convolution

The block-by-block convolution algorithm provides a dramatic decrease in computer time required for convolutions. As it was shown that the discrete convolution is identical to a numerical integration using the rectangular rule, the block-by-block algorithm provides the same decrease in computer time required for the solution of the governing nonlinear integral equation. While the time savings for a single analysis, relative to a direct nonlinear transient analysis is large (approximately 74% for the example presented), the time savings for each subsequent analysis is extremely large, as is summarized in Table 1. This extremely fast reanalysis will facilitate the use of numerical optimization for locally nonlinear structures.

Currently, each block is integrated using the convolution (i.e. rectangular rule). However, it is a straightforward matter to implement the Trapezoid rule, which will provide an increase in accuracy. The algorithm lends itself to parallel computation, and to variable step size/block size as well. In its simplest form, the step size for each block can be established independently of the other blocks. For example, a short initial block with a small step size can be used to minimize starting errors.

4. *RECURSIVE TRANSITION MATRIX ALGORITHMS*

The algorithms developed in this section, Recursive Algorithms 1 and 2 (RA1 and RA2), were developed prior to Recursive Algorithm 3 (RA3), which is presented in Section 3 of this report. Algorithms RA1 and RA2 are based on modal transition matrices, and hence are fundamentally different form RA3. As was discussed in Section 1.2, these algorithms did not meet the goal stated in Section 1.1, but are documented for completeness, and for the potential benefit of the developments in other applications, specifically with regard to RA2. We begin with some background.

Many approaches to the transient analysis of locally nonlinear problems are *direct*, in that they are based on the physical-coordinate (not modally transformed) second-order differential equations [14, Sec. 2.C.1],

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{f}_{nl}(\mathbf{x},\dot{\mathbf{x}}) = \mathbf{f}_e \tag{4.1}$$

where **x** is a vector of "N" nodal DOF, and the ($\dot{}$), ($\ddot{}$) indicate the first and second time derivatives. **M** is an N by N symmetric positive-definite mass matrix, $\mathbf{f}_e$ is a vector of externally applied loads, and $\mathbf{f}_{nl}$ represents the loading which is a nonlinear function of the displacement and velocity vectors. In the case where the damping and elastic forces have linear portions separable from the nonlinear portions, Eq. (4.1) can be written as

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{f}_e + \mathbf{f}_{nl}(\mathbf{x},\dot{\mathbf{x}}) \tag{4.2}$$

where **C** is a symmetric positive-semidefinite damping matrix, and **K** is a symmetric positive-semidefinite stiffness matrix. These three matrices, along with **M**, represent the linear portion of the model. In Eq. (4.2), the vector $\mathbf{f}_{nl}(\mathbf{x},\dot{\mathbf{x}})$ represents the forces imposed on the linear portion of the model by nonlinear elements in the model. It should again be noted that we are limiting our attention to localized nonlinear components which have no internal DOF. Such a component could not be represented by Eq. (4.2), but rather by the following system of equations,

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{f}_e + \mathbf{f}_{nl}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{z}, \dot{\mathbf{z}}) \tag{4.3}$$

$$\ddot{\mathbf{z}} = \mathbf{g}_{nl}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{z}, \dot{\mathbf{z}}) \tag{4.4}$$

where **z** is an n*1 vector of DOF internal to the nonlinear components, and $\mathbf{g}_{nl}$ is the nonlinear vector function representing the nonlinear components. Note that we have excluded nonlinear inertia terms for reasons of clarity, not necessity. As discussed in [14, Sec. 2.C.1], Eqs. (4.1) through (4.4) can describe spatially-discrete models with nonlinear elastic and/or visco-elastic elements, as well as geometrically nonlinear elements. However, phenomena such as plasticity and visco-plasticity are not addressed by these equations due to the history-dependence of the associated internal forces of these phenomena. A large variety of algorithms exist for the solution of Eqs. (4.1) through (4.4) and are well-documented. See Hughes [15] for a detailed summary of many current methods. A commonly used algorithm in structural dynamics is the average-acceleration, unconditionally stable Newmark method [16].

It is also possible to address these problems in first-order form, i.e.

$$\dot{\mathbf{y}} = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{f}_{nl}(\mathbf{y}, t) \tag{4.5}$$

where **y** is a (2N x 1) vector contain both displacements and velocities, **A** is the (2N x 2N) non-symmetric, positive semi-definite system matrix, **B** is the (2N x b) input matrix (for “b” inputs), and $\mathbf{f}_{nl}$(y,t) is the (b x 1) vector of nonlinear loads associated with this equation. More will be said later about the first-order form for locally-nonlinear transient structural dynamics.

A disadvantage of direct methods is the requirement to retain all DOF of the model, regardless of whether the response of these DOF are of interest. In contrast to the direct methods, the use of modal methods in the solution of the differential equations (4.2) and (4.5) can eliminate the need to retain all model DOF, but a disadvantage with such a modal approach is that, strictly speaking, the number of retained modes required can be assessed only after (two) solutions have been performed, thereby demonstrating modal convergence.

The direction of this work is the development of a solution method of Eq. (4.12) which is designed to have an implicit and exact model reduction, in that only those model DOF directly associated with externally applied and nonlinear forces need be retained in the analysis, and to be numerically robust with excellent convergence properties. This integral equation formulation for transient structural synthesis [2,3], has been shown to provide significant reductions in computer time as compared with direct integration using a well-known commercial finite element program [3], due to this implicit reduction.

We will describe two transition matrix algorithms developed. We discuss the motivation for these approaches, why these algorithms are not suitable for the purpose at hand, and what insights and results of value were gained by the development.

4.1 Recursive Solutions of First-Order Differential Equations

We begin with a brief review of recursive convolution solutions to first-order differential equations, due to the development of algorithms RA1, RA1PC, RA2, and RA2PC. The following brief review can be found in many texts in more detail; see for example, Meirovitch [17].

A recursive solution is attractive due to its reduced storage requirements, which are non-trivial when processing large FE models, and due to the reduction in compute time which can be obtained, depending both on the algorithm, and the structure representation employed.

The total solution of the linear first-order equation,

$$\dot{\mathbf{y}} = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{f}(t) \tag{4.6}$$

is given by,

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{f}(\tau)d\tau \tag{4.7}$$

Evaluating the solution, Eq. (4.7), at sample points k and k+1, gives respectively

$$\mathbf{x}(kT) = e^{\mathbf{A}kT}\mathbf{x}(0) + \int_0^{kT} e^{\mathbf{A}(kT-\tau)}\mathbf{B}\mathbf{f}(\tau)d\tau \tag{4.8}$$

$$\mathbf{x}(kT+T) = e^{\mathbf{A}(kT+T)}\mathbf{x}(0) + \int_0^{kT+T} e^{\mathbf{A}(kT+T-\tau)}\mathbf{B}\mathbf{f}(\tau)d\tau \tag{4.9}$$

A fundamental assumption required to produce an explicit recursive (discrete) form of Eq. (4.7) is that the force **f** is constant over the sample interval, i.e. $\mathbf{f}(\tau)=\mathbf{f}^{(kT)}$ for $kT \le \tau \le (k+1)T$. With this assumption, and making the change of variable, $\sigma = kT + T - \tau$, and defining $\Phi = e^{\mathbf{A}T}$ and $\Gamma = \int_0^T e^{\mathbf{A}\sigma}d\sigma\mathbf{B} = \mathbf{A}^{-1}\left(e^{\mathbf{A}T} - \mathbf{I}\right)\mathbf{B}$, the following discrete recursive form of Eq. (4.7) is found:

$$\mathbf{x}^{(k+1)} = \mathbf{\Phi}\mathbf{x}^{(k)} + \mathbf{\Gamma}\mathbf{f}^{(k)} \tag{4.10}$$

The advantage in the use of Eq. (4.10), as opposed to Eq. (4.7), is as follows. Considering a single-input, single-output response calculation over K sample points, the convolution integral in Eq. (4.7) requires on the order of k^2 flops, and at each sample point k, all prior values of **f** and **Φ** are required, k-1, k-2, k-3, etc. The recursive form, i.e. Eq. (4.10) requires only the data at sample point k-1 to calculate the response at sample point k. This constitutes a large saving in both compute time and data storage required. While these advantages are significant, the calculation of the matrix exponential **Φ** is required. As the system matrix **A** is 2n by 2n, the explicit computation of **Φ** is not practical for large models. Modal calculations of **Φ** are possible, but require the calculation of the left eigenvectors of **A**. This is not an attractive option either, given that most commercial finite element programs do not provide for this calculation, although it is possible in NASTRAN using the DMAP language. Our motivation is therefore to develop a recursive, transition-matrix solution algorithm solution which has an inherent implicit exact model reduction, and the above described advantages of a recursive procedure. We also stipulate that any such method not require the calculation of exp(AT), the matrix exponential of the 2N x

2N system matrix. Our intermediate goal here is the development of a recursive, transition matrix approach based on the second-order differential equation model, i.e. Eq. (2.1).

A recursive predictor-corrector algorithm for locally nonlinear earthquake isolation was developed by Inaudi and De La Llera, and implemented in the INADEL program [7]. This algorithm represents the structure with a state-space model, i.e. Eq. (4.5). The algorithm developed in [7] is rendered implicit by the use of non-constant approximations for the force **f**, which is defined quite generally. While this work contributes to the motivation for exploring a recursive transition matrix approach, as stated above, a goal of the current work is to avoid the calculation of the transition matrix (exponential of the system matrix), due to the attendant computational demand.

4.2 Recursive Algorithm 1: First-Order Modal Transition Matrix

The modal transformation, $\mathbf{x}=\mathbf{\Phi q}$, of Eq. (2.1), for proportional damping, produces a set of uncoupled modal differential equations of the form,

$$\ddot{\mathbf{q}} + \mathrm{diag}(2\varsigma_i\omega_i)\dot{\mathbf{q}} + \mathrm{diag}\left(\omega_i^2\right)\mathbf{q} = \tilde{\mathbf{f}} \qquad (4.11)$$

where the modal force is $\tilde{\mathbf{f}}(t) = \mathbf{\Phi}^T\mathbf{f}(t)$, $\mathbf{\Phi}$ containing mass-normalized mode shapes, and where (~) indicates a modal quantity. In the developments which follow, the force **f**(t) may be interpreted to be,

$$\mathbf{f}(t) \equiv \mathbf{f}(t, \mathbf{x}^*(t), \dot{\mathbf{x}}^*(t))$$

The set of (at most) N equations, Eq. (4.11), is comprised of two basic forms. The rigid body modes ($\omega_i = 0$) are described by

$$\ddot{q}_i = \tilde{f}_i \qquad i=1,2,..,r \qquad (4.12)$$

where r is the number of rigid body modes (RBM) possessed by the structure, and excluding mechanisms, $r \leq 6$. The solution to Eq. (4.12) is given by,

$$q_i(t) = \int_0^t \int_0^\tau \tilde{f}_i(\sigma) d\sigma d\tau + q_{i_0} + \dot{q}_{i_0} t = \int_0^t (t - \tau) \tilde{f}_i(\tau) d\tau + q_{i_0} + \dot{q}_{i_0} t \qquad (4.13)$$

The elastic modes ($\omega_i > 0$) are described by:

$$\ddot{q}_i + 2\varsigma_i \omega_i \dot{q}_i + \omega_i^2 q_i = \tilde{f}_i \qquad (r+1) \leq i \leq N \qquad (4.14)$$

where the number of elastic modes retained is typically much less than N, the total number of nodal DOF.

We may also write the solution to Eqs. (4.12) as a convolution integral,

$$q_i(t) = \int_0^t \hat{h}_i(t-\tau) \tilde{f}_i(\tau) d\tau + q_{i_0} + \dot{q}_{i_0} t, \qquad (4.15)$$

where q_{i_0} and $\dot{q}_{i_0}$ are the modal initial conditions, and where $\hat{h}_i(t)$ is the modal impulse response function (IRF). As seen from Eq. (4.13), the IRF for an RBM is,

$$\tilde{h}_i(t) = t, \qquad i=1,2,..,r, \qquad (4.16)$$

and for an elastic mode, the modal IRF is given by,

$$\tilde{h}_i(t) = \frac{1}{\omega_{di}} e^{-\varsigma_i \omega_i t} \sin(\omega_{di} t), \qquad (r+1) \leq i \leq N. \qquad (4.17)$$

We will construct the modal transition matrices for the rigid body and elastic modes, and the associated recursive algorithm, based on the physical (nodal) and modal state vectors defined as follows:

$$\hat{\mathbf{q}}_i \equiv \begin{Bmatrix} q_i \\ \dot{q}_i \end{Bmatrix} \qquad \hat{\mathbf{x}} \equiv \begin{Bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{Bmatrix} \qquad (4.18)$$

Based on Eqs. (4.15), (4.16), and (4.18), the solution to the rigid body mode equation, Eq. (4.12), can be written as

$$\hat{\mathbf{q}}_i(t) = \boldsymbol{\Psi}_{rb}(t)\hat{\mathbf{q}}_{i_0} + \int_0^t \boldsymbol{\Psi}_{rb}(t-\tau)\mathbf{B}\tilde{f}_i(\tau)d\tau \tag{4.19}$$

where $\mathbf{B} = (0,1)^T$ and where the matrix $\boldsymbol{\Psi}_{rb}(t)$ is defined as,

$$\boldsymbol{\Psi}_{rb}(t) = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}. \tag{4.20}$$

In order for a matrix $\boldsymbol{\Psi}$ to be a transition matrix, it must satisfy the following property:

$$\boldsymbol{\Psi}(t+\Delta t) = \boldsymbol{\Psi}(t)\boldsymbol{\Psi}(\Delta t) = \boldsymbol{\Psi}(\Delta t)\boldsymbol{\Psi}(t) \tag{4.21}$$

It is easily shown that $\boldsymbol{\Psi}_{rb}$ satisfies this property, and hence $\boldsymbol{\Psi}_{rb}$ is the RBM transition matrix, as will be shown here. We can write Eq. (4.19) for a time one sample later, i.e. at t+Δt,

$$\hat{\mathbf{q}}_i(t+\Delta t) = \boldsymbol{\Psi}_{rb}(t+\Delta t)\hat{\mathbf{q}}_{i_0} + \int_0^{t+\Delta t} \boldsymbol{\Psi}_{rb}(t+\Delta t-\tau)\mathbf{B}\tilde{f}_i(\tau)d\tau, \tag{4.22}$$

and as was done to obtain Eq. (4.10), in conjunction with the fact that $\boldsymbol{\Psi}_{rb}$ satisfies Eq. (4.21), we find the recursive form of Eq. (4.19),

$$\hat{\mathbf{q}}_i(t+\Delta t) = \boldsymbol{\Psi}_{rb}(\Delta t)\hat{\mathbf{q}}_i(t) + \boldsymbol{\Gamma}_{rb}(\Delta t)\tilde{f}_i(t) \tag{4.23}$$

where

$$\boldsymbol{\Gamma}_{rb}(\Delta t) = \int_0^{\Delta t} \boldsymbol{\Psi}_{rb}(\sigma)d\sigma\mathbf{B} = \Delta t\begin{bmatrix} 1 & .5\Delta t \\ 0 & 1 \end{bmatrix}\mathbf{B}. \tag{4.24}$$

If we define the kth time step $t_k = k\Delta t$, denote a quantity evaluated at the kth step as •(k), and recognize that $\boldsymbol{\Psi}_{rb}$ and $\boldsymbol{\Gamma}_{rb}$ are constants for a given sample length Δt, Eq. (4.23) can be written as,

$$\hat{\mathbf{q}}_i((k+1)T) = \boldsymbol{\Psi}_{rb}(\Delta t)\hat{\mathbf{q}}_i(kT) + \boldsymbol{\Gamma}_{rb}(\Delta t)\tilde{f}_i(kT) \qquad i=1,2,\ldots,r \tag{4.25}$$

Focusing on an elastic mode, the state-space equation of motion for an elastic mode is,

$$\dot{\hat{\mathbf{q}}}_i = \mathbf{A}_i\hat{\mathbf{q}}_i + \mathbf{B}\tilde{f}_i \tag{4.26}$$

and the recursive form of the solution to Eq. (4.26) is,

$$\hat{\mathbf{q}}_i^{(k+1)} = \boldsymbol{\Psi}_i\hat{\mathbf{q}}_i^{(k)} + \boldsymbol{\Gamma}_i\tilde{f}_i \qquad (r+1) \le i \le N. \tag{4.27}$$

and where $\boldsymbol{\Psi}_i = e^{\mathbf{A}_i\Delta t}$, $\boldsymbol{\Gamma}_i = \mathbf{A}_i^{-1}(\boldsymbol{\Psi}_i - \mathbf{I})\mathbf{B}$, $\mathbf{B}=[0\ 1]^T$, and the mode system matrix is,

$$\mathbf{A}_i = \begin{bmatrix} 0 & 1 \\ -\omega_i^2 & -2\varsigma_i\omega_i \end{bmatrix} \tag{4.28}$$

The explicit form of RA1, based on Eq. (4.25) is given here. The number of modes retained is indicated by "p" where typically, p<<N.

Recursive Algorithm #1 - Explicit Form

```
•   Evaluate Modal Transition Matrices $\boldsymbol{\Psi}$ and Input Matrices $\boldsymbol{\Gamma}$
•   $k := 0$
•   for $i = 1 : p$
•       $\tilde{f}_i^{(k)} := \boldsymbol{\phi}_i^T \mathbf{f}\left(kT, \mathbf{x}^{(k)^*}, \dot{\mathbf{x}}^{(k)^*}\right)$
•       if $i \le r$ then
•           $\hat{\mathbf{q}}_i^{(k+1)} = \boldsymbol{\Psi}_{rb}\hat{\mathbf{q}}_i^{(k)} + \boldsymbol{\Gamma}_{rb}\tilde{f}_i^{(k)}$
•       else
•           $\hat{\mathbf{q}}_i^{(k+1)} = \boldsymbol{\Psi}_i\hat{\mathbf{q}}_i^{(k)} + \boldsymbol{\Gamma}_i\tilde{f}_I^{(k)}$
•       end
•   end
•   $\hat{\mathbf{x}}^{(k+1)} = \hat{\boldsymbol{\Phi}}\hat{\mathbf{q}}_i^{(k+1)}$
•   $k := k + 1$
```

4.3 Recursive Algorithm 2: 2nd-Order Complex Modal Transition Matrix

This algorithm is based on a newly developed transition matrix constructed from the solution to the second-order elastic mode differential equation. The key to the development of such a transition matrix is to write the solution of the second-order equations with a leading matrix which satisfies the property of Eq. (4.21), i.e. a leading matrix whose matrix factors commute. For the rigid-body modes, we will use the transition matrix already developed in RA1. We begin by denoting the complex ($j = \sqrt{-1}$) eigenvalues of the ith mode as

$$\lambda_i^+ = -\varsigma_i\omega_i + j\omega_{di}$$
$$\lambda_i^- = -\varsigma_i\omega_i - j\omega_{di}$$

and the homogeneous solution to Eq. (4.14) written in complex form is

$$q_i(t) = C_i^+ e^{\lambda_i^+ t} + C_i^- e^{\lambda_i^- t}. \tag{4.29}$$

where C_i^+ and C_i^- are constants of integration for the ith mode. The initial condition vector consistent with the modal state-vector of Eq. (51) is,

$$\hat{\mathbf{q}}_i(t=0) = \hat{\mathbf{q}}_i^0 = \begin{Bmatrix} q_i^0 \\ \dot{q}_i^0 \end{Bmatrix}$$

and hence the solution, in a form consistent with the modal state vector of Eq. (4.18), becomes,

$$\hat{\mathbf{q}}_i(t) = \frac{1}{2\omega_{di}} \begin{Bmatrix} e^{\lambda_i^- t}\left(j\dot{q}_i^0 + q_i^0(\omega_{di} + j\varsigma_i\omega_i)\right) - je^{\lambda_i^+ t}\left(\dot{q}_i^0 + q_i^0(j\omega_{di} + \varsigma_i\omega_i)\right) \\ \lambda_i^- e^{\lambda_i^- t}\left(j\dot{q}_i^0 + q_i^0(\omega_{di} + j\varsigma_i\omega_i)\right) - j\lambda_i^+ e^{\lambda_i^+ t}\left(\dot{q}_i^0 + q_i^0(j\omega_{di} + \varsigma_i\omega_i)\right) \end{Bmatrix} \tag{4.30}$$

It can be shown that the modal impulse response $\tilde{H}_i$ is available from the homogeneous solution given in Eq. (4.30) using the following initial conditions:

$$q_i(t) = \tilde{H}_i(t) \qquad \text{if} \qquad \hat{\mathbf{q}}_i^0 = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$$

$$\dot{q}_i(t) = \tilde{H}_i(t) \qquad \text{if} \qquad \hat{\mathbf{q}}_i^0 = \begin{Bmatrix} -\omega_i^{-2} \\ 1 \end{Bmatrix}$$

Due to its importance in what follows, we define the following modal state initial condition:

$$\hat{\mathbf{q}}_h = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}.$$

We now define the following matrices:

$$\mathbf{P}_{di} = \frac{1}{2}\begin{bmatrix} 1 - j\dfrac{\varsigma_i \omega_i}{\omega_{di}} & -j\dfrac{1}{\omega_{di}} \\ 1 + j\dfrac{\varsigma_i \omega_i}{\omega_{di}} & j\dfrac{1}{\omega_{di}} \end{bmatrix} \qquad \mathbf{\Lambda}_i = \mathrm{diag}\left(\lambda_i^+, \lambda_i^-\right) \qquad \mathbf{P}_{vi} = \mathbf{\Lambda}_i \mathbf{P}_{di} \tag{4.31}$$

Given these definitions, it can be shown that the modal displacement and velocity are,

$$q_i(t) = \mathbf{v}^T e^{\mathbf{\Lambda}_i t} \mathbf{P}_{di} \hat{\mathbf{q}}_i^0 \tag{4.32}$$

$$\dot{q}_i(t) = \mathbf{v}^T e^{\mathbf{\Lambda}_i t} \mathbf{P}_{vi} \hat{\mathbf{q}}_i^0 \tag{4.33}$$

where $\mathbf{v}^T = [1 \ 1]$. In addition to the relation of Eqs. (4.31), the matrices $\mathbf{P}_d$ and $\mathbf{P}_v$ have the additional relation,

$$\mathbf{P}_{vi} = \mathbf{P}_{di} \mathbf{A}_i \tag{4.34}$$

revealing that $\mathbf{A}_i = \mathbf{\Lambda}_i^{-1}$.

Using these definitions, we can write the total modal transient response as:

$$q_i(t) = \mathbf{v}^T e^{\mathbf{\Lambda}_i t} \mathbf{P}_{di} \hat{\mathbf{q}}_i^0 + \int_0^t \mathbf{v}^T e^{\mathbf{\Lambda}_i (t-\tau)} \mathbf{P}_{di} \hat{\mathbf{q}}_h \boldsymbol{\phi}_i^T \mathbf{f}_i(\tau) d\tau \tag{4.35}$$

$$\dot{q}_i(t) = \mathbf{v}^T e^{\mathbf{\Lambda}_i t} \mathbf{P}_{vi} \hat{\mathbf{q}}_i^0 + \int_0^t \mathbf{v}^T e^{\mathbf{\Lambda}_i (t-\tau)} \mathbf{P}_{vi} \hat{\mathbf{q}}_h \boldsymbol{\phi}_i^T \mathbf{f}_i(\tau) d\tau \tag{4.36}$$

It is of interest to note that $\hat{\mathbf{q}}_h = \mathbf{B}$, where $\mathbf{B}$ was defined in RA1 above. However, it should be emphasized that $\hat{\mathbf{q}}_h$ is an initial condition vector which produces the impulse response function from the homogeneous solution to the damped modal oscillator.

We now define a *complex modal state vector* as follows:

$$\bar{\mathbf{q}}(t) = \begin{Bmatrix} q_i^+(t) \\ q_i^-(t) \end{Bmatrix} \tag{4.37}$$

Using this state vector, Eqs. (4.32) and (4.33) can be written

$$q_i = \mathbf{v}^T \bar{\mathbf{q}}_i \tag{4.38}$$

$$\dot{q}_i = \mathbf{v}^T \dot{\bar{\mathbf{q}}}_i \tag{4.39}$$

where we have dropped the time-dependence notation for simplicity.

We now construct the recursion using the above relations. The total modal response, Eqs. (4.35) and (4.36) can be written in terms of the complex modal state vector,

$$\bar{\mathbf{q}}_i(t) = e^{\mathbf{\Lambda}_i t}\mathbf{P}_{di}\hat{\mathbf{q}}_i^0 + \int_0^t e^{\mathbf{\Lambda}_i (t-\tau)}\mathbf{P}_{di}\hat{\mathbf{q}}_h \boldsymbol{\phi}_i^T \mathbf{f}(\tau)d\tau \tag{4.40}$$

$$\dot{\bar{\mathbf{q}}}_i(t) = e^{\mathbf{\Lambda}_i t}\mathbf{P}_{vi}\hat{\mathbf{q}}_i^0 + \int_0^t e^{\mathbf{\Lambda}_i (t-\tau)}\mathbf{P}_{vi}\hat{\mathbf{q}}_h \boldsymbol{\phi}_i^T \mathbf{f}(\tau)d\tau \tag{4.41}$$

Equations (4.40) and (4.41) can be written at a time one sample Δt later,

$$\bar{\mathbf{q}}_i(t+\Delta t) = e^{\mathbf{\Lambda}_i (t+\Delta t)}\mathbf{P}_{di}\hat{\mathbf{q}}_i^0 + \int_0^{t+\Delta t} e^{\mathbf{\Lambda}_i (t+\Delta t-\tau)}\mathbf{P}_{di}\hat{\mathbf{q}}_h \boldsymbol{\phi}_i^T \mathbf{f}(\tau)d\tau \tag{4.42}$$

$$\dot{\bar{\mathbf{q}}}_i(t+\Delta t) = e^{\mathbf{\Lambda}_i (t+\Delta t)}\mathbf{P}_{vi}\hat{\mathbf{q}}_i^0 + \int_0^{t+\Delta t} e^{\mathbf{\Lambda}_i (t+\Delta t-\tau)}\mathbf{P}_{vi}\hat{\mathbf{q}}_h \boldsymbol{\phi}_i^T \mathbf{f}(\tau)d\tau \tag{4.43}$$

Using the fact that $e^{\mathbf{\Lambda}_i t}$ satisfies the property of Eq. (4.21), Eqs. (4.42) and (4.43) become,

$$\bar{\mathbf{q}}_i(t+\Delta t) = e^{\Lambda_i \Delta t}\left(e^{\Lambda_i t}\mathbf{P}_{di}\hat{\mathbf{q}}_i^0 + \int_0^t e^{\Lambda_i (t-\tau)}\mathbf{P}_{di}\hat{\mathbf{q}}_h \boldsymbol{\phi}_h^T \mathbf{f}(\tau)d\tau \right) + \ldots$$

$$\ldots + \int_t^{t+\Delta t} e^{\Lambda_i (t+\Delta t-\tau)}\mathbf{P}_{di}\hat{\mathbf{q}}_h \boldsymbol{\phi}_h^T \mathbf{f}(\tau)d\tau \qquad (4.44)$$

$$\dot{\bar{\mathbf{q}}}_i(t+\Delta t) = e^{\Lambda_i \Delta t}\left(e^{\Lambda_i t}\mathbf{P}_{vi}\hat{\mathbf{q}}_i^0 + \int_0^t e^{\Lambda_i (t-\tau)}\mathbf{P}_{vi}\hat{\mathbf{q}}_h \boldsymbol{\phi}_h^T \mathbf{f}(\tau)d\tau \right) + \ldots$$

$$\ldots + \int_t^{t+\Delta t} e^{\Lambda_i (t+\Delta t-\tau)}\mathbf{P}_{vi}\hat{\mathbf{q}}_h \boldsymbol{\phi}_h^T \mathbf{f}(\tau)d\tau \qquad (4.45)$$

If we again assume that the sampling period is sufficiently small such that **f** does not vary appreciably over Δt, and with the change of variable $\sigma = t + \Delta t - \tau$, Eqs. (4.44) and (4.45) become

$$\bar{\mathbf{q}}_i(t+\Delta t) = e^{\Lambda_i \Delta t}\left(e^{\Lambda_i t}\mathbf{P}_{di}\hat{\mathbf{q}}_i^0 + \int_0^t e^{\Lambda_i (t-\tau)}\mathbf{P}_{di}\hat{\mathbf{q}}_h \boldsymbol{\phi}_h^T \mathbf{f}(\tau)d\tau \right) + \ldots$$

$$\ldots + \int_0^{\Delta t} e^{\Lambda_i (\sigma)}d\sigma \mathbf{P}_{di}\hat{\mathbf{q}}_h \boldsymbol{\phi}_h^T \mathbf{f}(t) \qquad (4.46)$$

$$\dot{\bar{\mathbf{q}}}_i(t+\Delta t) = e^{\Lambda_i \Delta t}\left(e^{\Lambda_i t}\mathbf{P}_{vi}\hat{\mathbf{q}}_i^0 + \int_0^t e^{\Lambda_i (t-\tau)}\mathbf{P}_{vi}\hat{\mathbf{q}}_h \boldsymbol{\phi}_h^T \mathbf{f}(\tau)d\tau \right) + \ldots$$

$$\ldots + \int_0^{\Delta t} e^{\Lambda_i (\sigma)}d\sigma \mathbf{P}_{vi}\hat{\mathbf{q}}_h \boldsymbol{\phi}_h^T \mathbf{f}(t) \qquad (4.47)$$

Recognizing the states at time t in Eqs. (4.46) and (4.47) allows the following simplification,

$$\bar{\mathbf{q}}_i(t+\Delta t) = e^{\Lambda_i \Delta t}\bar{\mathbf{q}}_i(t) + \int_0^{\Delta t} e^{\Lambda_i (\sigma)}d\sigma \mathbf{P}_{di}\hat{\mathbf{q}}_h \boldsymbol{\phi}_h^T \mathbf{f}(t) \qquad (4.48)$$

$$\dot{\bar{\mathbf{q}}}_i(t+\Delta t) = e^{\Lambda_i \Delta t}\dot{\bar{\mathbf{q}}}_i(t) + \int_0^{\Delta t} e^{\Lambda_i (\sigma)}d\sigma \mathbf{P}_{vi}\hat{\mathbf{q}}_h \boldsymbol{\phi}_h^T \mathbf{f}(t) \qquad (4.49)$$

and the integral in Eqs. (4.48) and (4.49) can be evaluated as,

$$\mathbf{S}_i(\Delta t) = \int_0^{\Delta t} e^{\Lambda_i (\sigma)}d\sigma = \mathrm{diag}\left(\frac{e^{\lambda_i^+ \Delta t} - 1}{\lambda_i^+}, \frac{e^{\lambda_i^- \Delta t} - 1}{\lambda_i^-} \right)$$

which yields the final recursions for the complex modal displacement and velocities

$$\bar{\mathbf{q}}_i(t+\Delta t) = \boldsymbol{\Psi}_i \bar{\mathbf{q}}_i(t) + \boldsymbol{\Gamma}_{di} \tilde{f}_i(t) \tag{4.50}$$

$$\dot{\bar{\mathbf{q}}}_i(t+\Delta t) = \boldsymbol{\Psi}_i \dot{\bar{\mathbf{q}}}_i(t) + \boldsymbol{\Gamma}_{vi} \tilde{f}_i(t) \tag{4.51}$$

where $\tilde{f}_i = \boldsymbol{\phi}_i^T \mathbf{f}$ and in which the following quantities are defined:

$$\boldsymbol{\Psi}_i = e^{\boldsymbol{\Lambda}_i \Delta t} \tag{4.52}$$

$$\boldsymbol{\Gamma}_{di} = \mathbf{S}_i \mathbf{P}_{di} \hat{\mathbf{q}}_h \tag{4.53}$$

$$\boldsymbol{\Gamma}_{vi} = \mathbf{S}_i \mathbf{P}_{vi} \hat{\mathbf{q}}_h \tag{4.54}$$

The explicit form of RA2, based on Eqs. (4.25), (4.50), and (4.51) is given here. For clarity, we again indicate the various quantities at time kT using a superscript notation, e.g. $\mathbf{q}(kT) = \mathbf{q}^k$. The number of modes retained is indicated by "p" where typically, p<<N.

Recursive Algorithm #2 - Explicit Form

```
•   Evaluate Modal Transition Matrices $\boldsymbol{\Psi}$ and Input Matrices $\boldsymbol{\Gamma}_d, \boldsymbol{\Gamma}_v$
•   Initialize k := 0
•   for i = 1 : p
•       $\tilde{f}_i^{(k)} := \boldsymbol{\phi}_i^T \mathbf{f}\left(kT, \mathbf{x}^{(k)^*}, \dot{\mathbf{x}}^{(k)^*}\right)$
•       if i ≤ r then
•           $\hat{\mathbf{q}}_i^{(k+1)} = \boldsymbol{\Psi}_{rb} \hat{\mathbf{q}}_i^{(k)} + \boldsymbol{\Gamma}_{rb} \tilde{f}_i^{(k)}$
•       else
•           $\bar{\mathbf{q}}_i^{(k+1)} = \boldsymbol{\Psi}_i \bar{\mathbf{q}}_i^{(k)} + \boldsymbol{\Gamma}_{di} f_i^{(k)}$
            $\dot{\bar{\mathbf{q}}}_i^{(k+1)} = \boldsymbol{\Psi}_i \dot{\bar{\mathbf{q}}}_i^{(k)} + \boldsymbol{\Gamma}_{vi} f_i^{(k)}$
•       end
•   end
•   $\mathbf{x}^{(k+1)} = \sum_{i=1}^{p} \boldsymbol{\phi}_i^T \mathbf{v}^T \bar{\mathbf{q}}_i^{(k+1)}$
    $\dot{\mathbf{x}}^{(k+1)} = \sum_{i=1}^{p} \boldsymbol{\phi}_i^T \mathbf{v}^T \dot{\bar{\mathbf{q}}}_i^{(k+1)}$
•   k := k + 1
```

4.4 Stability Analysis - Recursive Algorithm 1 - Explicit Form

The stability analysis for the explicit form of RA1 is given. The result of this analysis reveals a fundamental limitation of a modal transition matrix recursive solution to Eq. (2.19), in that the spectral radius of the integration matrix operator has a limiting value of 1.0, approached from above, as the sample length is made smaller.

For the stability analysis, the structural modifications (e.g. isolators) are assumed to be linear elastic and hence represented by

$$\mathbf{f}^{(k)} = -\mathbf{K}^{*}\mathbf{x}^{*(k)}. \tag{4.55}$$

The linear matrix operator representing RA1 is therefore,

$$\hat{\mathbf{q}}^{(k+1)} = \left[\boldsymbol{\Psi} - \boldsymbol{\Gamma}\boldsymbol{\Phi}^{T}\mathbf{K}^{*}\boldsymbol{\Phi}\mathbf{T}_{b}\right]\hat{\mathbf{q}}^{(k)} = \mathbf{T}_{1}\hat{\mathbf{q}}^{(k)} \tag{4.56}$$

where the matrices $\boldsymbol{\Psi}$ and $\boldsymbol{\Gamma}$, defined for RA1, are of size 2p by 2p, and 2p by p respectively, and contain contributions from all "p" retained modes, and $\mathbf{T}_b$ (2p by 2p) is a Boolean matrix which rearranges the elements of $\hat{\mathbf{q}}^{(k)}$ (2p by 1) to be consistent with form of Eq. (4.55). The stability of the algorithm is measured by the spectral radius of $\mathbf{T}_1$, i.e. stability requires that

$$\rho(\mathbf{T}_1) \leq 1.$$

We will postpone the example calculations until the stability analysis of RA2 is presented. We should also note that this condition on the spectral radius is theoretical; the degree to which ρ must be less than unity is dependent on numerical considerations.

4.5 Stability Analysis - Recursive Algorithm 2 - Explicit Form

We define a combined complex modal state vector, of dimensions 4x1, as

$$\overline{\mathbf{Q}}_{i}^{(k)} \equiv \begin{Bmatrix} \overline{\mathbf{q}}_{i} \\ \dot{\overline{\mathbf{q}}}_{i} \end{Bmatrix} \tag{4.57}$$

and along with Eq. (4.55), allows the elastic mode recursion of RA2 to be written as the following linear matrix operator,

$$\overline{\mathbf{Q}}^{(k+1)} = \left[\mathbf{\Psi} - \mathbf{\Gamma}\mathbf{\Phi}^{T}\mathbf{K}^{*}\mathbf{\Phi}\mathbf{V}^{T}\right]\overline{\mathbf{Q}}^{(k)} = \mathbf{T}_2\overline{\mathbf{Q}}^{(k)} \tag{4.58}$$

where the matrices $\mathbf{\Psi}$ and $\mathbf{\Gamma}$, defined for RA2, are 4p by 4p, and 4p by p, respectively, and $\mathbf{V}^T$ is p by 4p. Each of these matrices is block quasi-diagonal where each block is associated with a retained mode, and the definition of which can be found in the above section outlining the development of RA2. The stability of the algorithm is measured by the spectral radius of $\mathbf{T}_2$, i.e. stability requires that $\rho(\mathbf{T}_2) \leq 1$.

4.6 Example Calculations - Stability of Explicit Recursive Algorithms

We will make use of a simple 4 DOF spring-mass model to calculate algorithm stability. The system is shown in Figure 9. In this example, the synthesis will install the stiffness modification k*, and calculate the transient response of mass #1, m_1. The example system has the following parameter values,

$$\mathbf{M} = \text{diag}(1.7513 \quad 2.1016 \quad 0.3503 \quad 1.7513)\ \text{kg},$$
$$k^* = 700\ \text{N/m} \quad k_1 = k_2 = k_3 = 175.13\ \text{N/m},$$

and has natural frequencies in Hertz, $f_1 = 0.6261$, $f_2 = 1.5421$, $f_3 = 2.5750$ $f_4 = 5.2678$ (Hz).

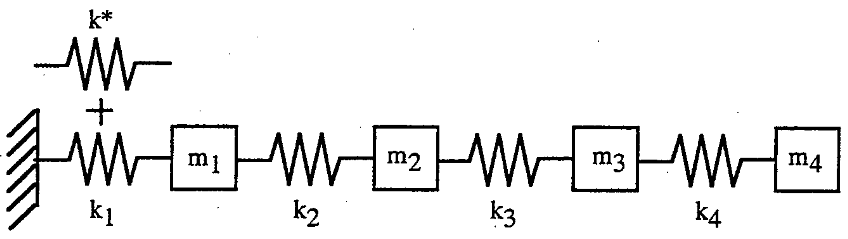

Figure 9. A Lumped 4-DOF System with a stiffness modification

As the purpose of this example is to determine stability of the algorithms RA1 and RA2, we limit the example to a linear elastic modification.

Stability of RA1: We first calculate the spectral radius of the explicit operator for RA1 (Eq. (4.56)), as a function of sample length, Δt, for a fixed value of $k^* = 700$ N/m, with $\zeta = 0.0$, $\zeta = 0.05$, $\zeta = 0.2$, and $\zeta = 0.3$. This is shown in Figure 10. The range of sample lengths is from 0.0001 s to 0.01 s, which is consistent with the range of natural periods for the model which is 0.19 s to 1.6 s, with respect to the requirements on a sufficiently small time step.

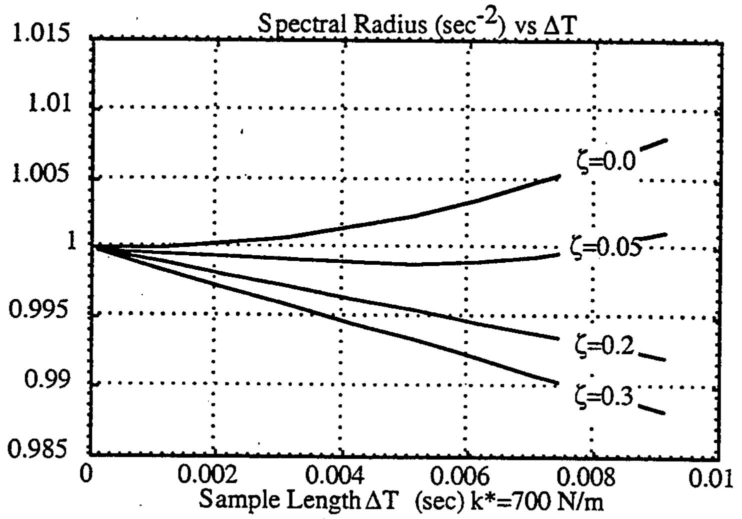

Figure 10. Spectral radius of $\mathbf{T}_1$ versus sample length Δt for various ζ

From this plot, it is clear that the simple modal-based recursion is ineffective for the solution of Eq. (2.14). The algorithm achieves marginal stability, as the step size is made small, for large values of modal damping. This result is attributed to the use of the modal transition matrix, $e^{A\Delta t}$, which clearly has a limiting spectral radius of 1, for $\Delta t \rightarrow 0$. By increasing the modal damping, the region of neutral stability is extended to longer sample lengths, but this does not offer any practical value.

We can also examine the dependence of the stability on the stiffness value k^*, for fixed sample length of $\Delta t = 0.01$s, and for various values of ζ, shown in Figure 11. Again, unrealistically high values of modal damping are required to stabilize the algorithm.

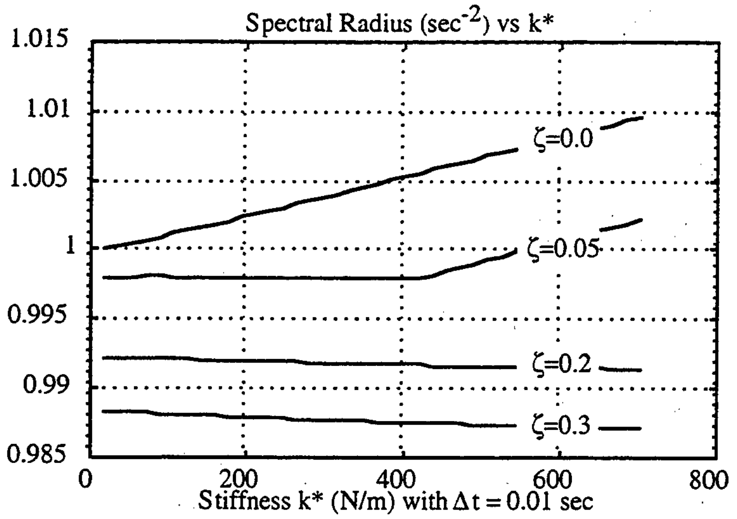

Figure 11. Spectral radius of $\mathbf{T}_1$ versus stiffness k* for various ζ

Stability of RA2: We calculate the spectral radius of the operator $\mathbf{T}_2$ (Eq. (4.58)) for explicit-RA2, as a function of sample length, Δt, for k* = 700 N/m, and for various values of ζ. The results of this calculation are shown in Figure 12. These results are in fact identical to the corresponding results for RA1, indicating that the use of the modal transition matrix based on the second-order equations does not have any effect on the stability, as might be expected, as the difference between the algorithms is essentially a change of coordinates, and both algorithms are explicit. It is seen that RA2 cannot be stabilized by increasing the modal damping.

The dependence of stability of RA2 with varying k* is shown in Figure 13, and again the results of Figure 13 are identical to those calculated using RA1, while varying k*.

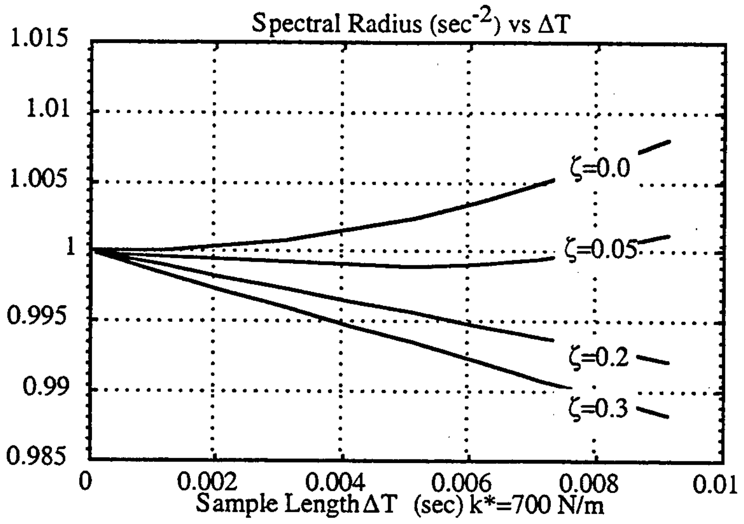

Figure 12. Spectral radius of $\mathbf{T}_2$ versus sample length Δt for various ζ

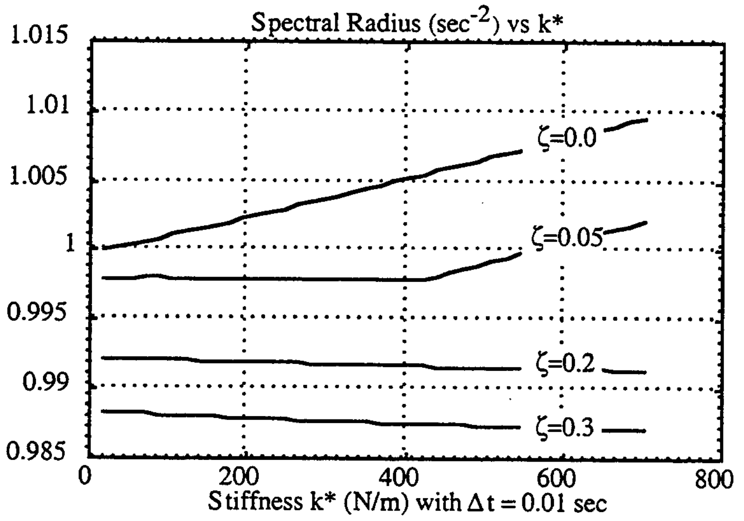

Figure 13. Spectral radius of $\mathbf{T}_2$ versus stiffness k* for various ζ

The algorithms RA1 and RA2 are attractive in that they satisfy some of the requirements set forth above regarding the development of a solution method for Eq. (2.14), specifically in that these algorithms achieve a model reduction using modal superposition. The results shown immediately above indicate that these algorithms are not viable methods, in their explicit forms. We will now develop implicit, predictor-corrector versions of these algorithms, to be referred to as RA1PC, and RA2PC, to see if the instability of the explicit forms can be overcome.

4.7 Recursive Algorithm 1: Predictor-Corrector Form

We begin with a recursive form of the solution to Eq. (4.14)

$$\hat{\mathbf{q}}_i^{(k+1)} = e^{\mathbf{A}_i \Delta t}\hat{\mathbf{q}}_i^{(k)} + \int_t^{t+\Delta t} e^{\mathbf{A}_i(t+\Delta t-\tau)}\mathbf{B}\tilde{\mathbf{f}}_i d\tau, \tag{4.59}$$

and letting $\sigma = t + \Delta t - \tau$, then $d\sigma = -d\tau$ and Eq. (4.59) becomes,

$$\hat{\mathbf{q}}_i^{(k+1)} = e^{\mathbf{A}_i \Delta t}\hat{\mathbf{q}}_i^{(k)} + \int_0^{\Delta t} e^{\mathbf{A}_i \sigma}\mathbf{B}\tilde{\mathbf{f}}_i(t + \Delta t - \tau)d\sigma. \tag{4.60}$$

We employ the following interpolation for the modal force,

$$\tilde{f}_i = \alpha\tilde{f}_i(t) + (1-\alpha)\tilde{f}_i(t+\Delta t) \tag{4.61}$$

where $0 \le \alpha \le 1$. Using $\alpha = \sigma/\Delta t$, we have,

$$\hat{\mathbf{q}}_i^{(k+1)} = e^{\mathbf{A}_i \Delta t}\hat{\mathbf{q}}_i^{(k)} + \int_0^{\Delta t} e^{\mathbf{A}_i \sigma}\mathbf{B}\left\{\frac{1}{\Delta t}\left[\sigma\tilde{f}_i^{(k)} + (\Delta t - \sigma)\tilde{f}_i^{(k+1)}\right]\right\}d\sigma, \tag{4.62}$$

or,

$$\hat{\mathbf{q}}_i^{(k+1)} = e^{\mathbf{A}_i \Delta t}\hat{\mathbf{q}}_i^{(k)} + \frac{1}{\Delta t}\left(\int_0^{\Delta t} e^{\mathbf{A}_i \sigma}\sigma d\sigma\right)\mathbf{B}\left(\tilde{f}_i^{(k)} - \tilde{f}_i^{(k+1)}\right) + \left(\int_0^{\Delta t} e^{\mathbf{A}_i \sigma}d\sigma\right)\mathbf{B}\tilde{f}_i^{(k+1)}. \tag{4.63}$$

The integrals in Eq. (4.63) can now be evaluated,

$$\int_0^{\Delta t} e^{\mathbf{A}_i \sigma}d\sigma = \mathbf{A}_i^{-1}\left(e^{\mathbf{A}_i \Delta t} - \mathbf{I}\right),$$

$$\int_0^{\Delta t} e^{\mathbf{A}_i \sigma}\sigma d\sigma = \Delta t\mathbf{A}_i^{-1}e^{\mathbf{A}_i \Delta t} - \mathbf{A}_i^{-2}\left(e^{\mathbf{A}_i \Delta t} - \mathbf{I}\right).$$

Letting $\mathbf{\Psi} = e^{\mathbf{A}_i \Delta t}$ and $\mathbf{\Gamma} = \int_0^{\Delta t} e^{\mathbf{A}_i \sigma}d\sigma = \mathbf{A}_i^{-1}\left(e^{\mathbf{A}_i \Delta t} - \mathbf{I}\right)\mathbf{B}$, then,

$$\hat{\mathbf{q}}_i^{(k+1)} = \mathbf{\Psi}_i\hat{\mathbf{q}}_i^{(k)} + \mathbf{\Gamma}_i\tilde{f}_i^{(k+1)} + \mathbf{A}_i^{-1}\mathbf{\Psi}_i\mathbf{B}\left(\tilde{f}_i^{(k)} - \tilde{f}_i^{(k+1)}\right) - \frac{1}{\Delta t}\mathbf{A}_i^{-1}\mathbf{\Gamma}_i\left(\tilde{f}_i^{(k)} - \tilde{f}_i^{(k+1)}\right) \tag{4.64}$$

or finally,

$$\hat{\mathbf{q}}_i^{(k+1)} = \mathbf{\Psi}_i \hat{\mathbf{q}}_i^{(k)} + \mathbf{A}_i^{-1}\left(\mathbf{\Psi}_i \mathbf{B} - \frac{1}{\Delta t}\mathbf{\Gamma}_i\right)\left(\tilde{\mathbf{f}}_i^{(k)} - \tilde{\mathbf{f}}_i^{(k+1)}\right) + \mathbf{\Gamma}_i \tilde{\mathbf{f}}_i^{(k+1)} \tag{4.65}$$

The predictor-corrector algorithm, RA1PC is now outlined.

Recursive Algorithm #1 - Predictor-Corrector

```
•   $k := 0$

Predictor:

•   for i = 1 : p
•       $\hat{\mathbf{q}}_i^{(k+1)^p} = \mathbf{\Psi}_i \hat{\mathbf{q}}_i^{(k)} + \mathbf{\Gamma}_i \tilde{\mathbf{f}}_i^{(k)}$
•   end

•   $\mathbf{x}^{(k+1)^p} = \mathbf{\Phi}\mathbf{q}^{(k+1)^p}$
•   $\dot{\mathbf{x}}^{(k+1)^p} = \mathbf{\Phi}\dot{\mathbf{q}}^{(k+1)^p}$

•   for i = 1 : p
•       $\tilde{\mathbf{f}}_i^{(k+1)^p} = \mathrm{f}\left(\mathbf{x}^{(k+1)^p}, \dot{\mathbf{x}}^{(k+1)^p}\right)$
•   end

Corrector:

•   for i = 1 : p
•       $\hat{\mathbf{q}}_i^{(k+1)} = \mathbf{\Psi}_i \hat{\mathbf{q}}_i^{(k)} + \mathbf{A}_i^{-1}\left(\mathbf{\Psi}_i \mathbf{B} - \frac{1}{\Delta t}\mathbf{\Gamma}_i\right)\left(\tilde{\mathbf{f}}_i^{(k)} - \tilde{\mathbf{f}}_i^{(k+1)^p}\right) + \mathbf{\Gamma}_i \tilde{\mathbf{f}}_i^{(k+1)^p}$
•   end

•   $\mathbf{x}^{(k+1)} = \mathbf{\Phi}\mathbf{q}^{(k+1)}$
•   $\dot{\mathbf{x}}^{(k+1)} = \mathbf{\Phi}\dot{\mathbf{q}}^{(k+1)}$

•   for i = 1 : p
•       $\tilde{\mathbf{f}}_i^{(k+1)} = \mathrm{f}\left(\mathbf{x}^{(k+1)}, \dot{\mathbf{x}}^{(k+1)}\right)$
•   end
•   $k := k + 1$
```

4.8 Stability Analysis - Recursive Algorithm 1 Predictor-Corrector Form

For the linear stability analysis, we will again assume that

$$\mathbf{f}^{(k)} = -\mathbf{K}^{*}\mathbf{x}^{*(k)}. \tag{4.66}$$

Combining the equations above yields the following,

$$\hat{\mathbf{q}}^{(k+1)} = \left[(\mathbf{I}-\mathbf{\Gamma Z})\mathbf{\Psi} + (\mathbf{\Gamma Z})^2 + \mathbf{\theta Z}(\mathbf{\Psi} - \mathbf{\Gamma Z} - \mathbf{I})\right]\hat{\mathbf{q}}^{(k)} \tag{4.67}$$

where $\mathbf{Z} = \mathbf{\Phi}^T\mathbf{K}^*\mathbf{\Phi T}_q$, and $\mathbf{T}_q$ is a Boolean matrix which extracts displacement entries in the $\mathbf{q}$ vector. Equation (4.67) can be abbreviated as

$$\hat{\mathbf{q}}_i^{(k+1)} = \mathbf{T}_I^{pc}\hat{\mathbf{q}}_i^{(k)}, \tag{4.68}$$

and the condition for stability is that the spectral radius of $\mathbf{T}_I^{pc}$ does not exceed unity, i.e. $\rho(\mathbf{T}_I^{pc}) \leq 1$.

4.9 Recursive Algorithm 2: Predictor-Corrector Form

We begin with the following equations for the complex modal state responses,

$$\bar{\mathbf{q}}_i^{(k+1)} = e^{\Lambda_i \Delta t}\bar{\mathbf{q}}_i^{(k)} + \int_t^{t+\Delta t} e^{\Lambda_i(t+\Delta t-\tau)}\mathbf{P}_{di}\hat{\mathbf{q}}_h\tilde{\mathrm{f}}_i(\tau)d\tau \tag{4.69}$$

$$\dot{\bar{\mathbf{q}}}_i^{(k+1)} = e^{\Lambda_i \Delta t}\dot{\bar{\mathbf{q}}}_i^{(k)} + \int_t^{t+\Delta t} e^{\Lambda_i(t+\Delta t-\tau)}\mathbf{P}_{vi}\hat{\mathbf{q}}_h\tilde{\mathrm{f}}_i(\tau)d\tau \tag{4.70}$$

Making use of Eq. (4.61), and following the same procedure as used in RA1PC, we arrive at the following recursions for the ith complex modal state responses,

$$\bar{\mathbf{q}}_i^{(k+1)} = \mathbf{\Psi}_i\bar{\mathbf{q}}_i^{(k)} + \mathbf{\Theta}_{di}\left(\tilde{\mathrm{f}}_i^{(k)} - \tilde{\mathrm{f}}_i^{(k+1)^p}\right) + \mathbf{\Gamma}_{di}\tilde{\mathrm{f}}_i^{(k+1)^p} \tag{4.71}$$

$$\dot{\bar{\mathbf{q}}}_i^{(k+1)} = \mathbf{\Psi}_i\dot{\bar{\mathbf{q}}}_i^{(k)} + \mathbf{\Theta}_{vi}\left(\tilde{\mathrm{f}}_i^{(k)} - \tilde{\mathrm{f}}_i^{(k+1)^p}\right) + \mathbf{\Gamma}_{vi}\tilde{\mathrm{f}}_i^{(k+1)^p} \tag{4.72}$$

where the following quantities are defined:

$$\mathbf{\Psi}_i = e^{\Lambda_i \Delta t} \tag{4.73}$$

$$\mathbf{\Gamma}_{di} = \mathbf{\Lambda}_i^{-1}(\mathbf{\Psi}_i - \mathbf{I})\mathbf{P}_{di}\hat{\mathbf{q}}_h \tag{4.74}$$

$$\mathbf{\Gamma}_{vi} = \mathbf{\Lambda}_i^{-1}(\mathbf{\Psi}_i - \mathbf{I})\mathbf{P}_{vi}\hat{\mathbf{q}}_h \tag{4.75}$$

$$\mathbf{\Theta}_{di} = \mathbf{\Lambda}_i^{-1}\left(\mathbf{\Psi}_i\mathbf{P}_{di}\hat{\mathbf{q}}_h - \frac{1}{\Delta t}\mathbf{\Gamma}_{di}\right) \tag{4.76}$$

$$\mathbf{\Theta}_{vi} = \mathbf{\Lambda}_i^{-1}\left(\mathbf{\Psi}_i\mathbf{P}_{vi}\hat{\mathbf{q}}_h - \frac{1}{\Delta t}\mathbf{\Gamma}_{vi}\right) \tag{4.77}$$

The predictor-corrector algorithm, RA2PC is now outlined. In this algorithm, the relevant matrices have been expanded to include all retained modes; that is, we eliminate the "for-end" loops over the modes in favor of a matrix notation, where the various matrices are block-diagonal.

Recursive Algorithm #2 - Predictor-Corrector

```
•   $k := 0$
Predictor:
•   $\overline{\mathbf{Q}}^{(k+1)^p} = \mathbf{\Psi}\overline{\mathbf{Q}}^{(k)} + \mathbf{\Gamma}_d\tilde{\mathbf{f}}^{(k+1)}$
•   $\dot{\overline{\mathbf{Q}}}^{(k+1)^p} = \mathbf{\Psi}\dot{\overline{\mathbf{Q}}}^{(k)} + \mathbf{\Gamma}_{di}\tilde{\mathbf{f}}^{(k+1)}$
•   $\mathbf{x}^{(k+1)^p} = \mathbf{\Phi}\mathbf{V}_d^T\overline{\mathbf{Q}}^{(k+1)^p}$
•   $\dot{\mathbf{x}}^{(k+1)^p} = \mathbf{\Phi}\mathbf{V}_v^T\overline{\mathbf{Q}}^{(k+1)^p}$
•   for $i = 1 : p$
•       $\tilde{f}_i^{(k+1)^p} = f\left(\mathbf{x}^{(k+1)^p}, \dot{\mathbf{x}}^{(k+1)^p}\right)$
•   end
Corrector:
•   $\overline{\mathbf{Q}}^{(k+1)} = \mathbf{\Psi}\overline{\mathbf{Q}}^{(k)} + \mathbf{\Theta}_d\left(\tilde{\mathbf{f}}^{(k)} - \tilde{\mathbf{f}}^{(k+1)^p}\right) + \mathbf{\Gamma}_d\tilde{\mathbf{f}}^{(k+1)^p}$
•   $\dot{\overline{\mathbf{Q}}}^{(k+1)} = \mathbf{\Psi}\dot{\overline{\mathbf{Q}}}^{(k)} + \mathbf{\Theta}_v\left(\tilde{\mathbf{f}}^{(k)} - \tilde{\mathbf{f}}^{(k+1)^p}\right) + \mathbf{\Gamma}_v\tilde{\mathbf{f}}^{(k+1)^p}$
•   $\mathbf{x}^{(k+1)^p} = \mathbf{\Phi}\mathbf{V}_d^T\overline{\mathbf{Q}}^{(k+1)^p}$
•   $\dot{\mathbf{x}}^{(k+1)^p} = \mathbf{\Phi}\mathbf{V}_v^T\overline{\mathbf{Q}}^{(k+1)^p}$
•   for $i = 1 : p$
•       $\tilde{f}_i^{(k+1)} = f\left(\mathbf{x}^{(k+1)}, \dot{\mathbf{x}}^{(k+1)}\right)$
•   end
•   $k := k + 1$
```

4.10 Stability Analysis - Recursive Algorithm 2 Predictor-Corrector Form

For the linear stability analysis, we will assume that

$$\mathbf{f}^{(k)} = -\mathbf{K}^{*}\mathbf{x}^{*(k)} \tag{4.78}$$

Combining the equations above yields the following,

$$\overline{\mathbf{Q}}^{(k+1)} = \left[(\mathbf{I}-\mathbf{\Gamma Z})\mathbf{\Psi}+(\mathbf{\Gamma Z})^2+\mathbf{\Theta Z}(\mathbf{\Psi}-\mathbf{\Gamma Z}-\mathbf{I})\right]\overline{\mathbf{Q}}^{(k)} \tag{4.79}$$

where $\mathbf{Z} = \mathbf{\Phi}^T\mathbf{K}^*\mathbf{\Phi}\mathbf{V}^T$, and $\mathbf{V}$ is a Boolean matrix which effects the addition of the elements of the complex modal responses. Equation (4.79) can be written as

$$\overline{\mathbf{Q}}^{(k+1)} = \mathbf{T}_2^{pc}\overline{\mathbf{Q}}^{(k)} \tag{4.80}$$

The theoretical condition for stability is that the spectral radius of $\mathbf{T}_2^{pc}$ does not exceed unity, i.e $\rho(\mathbf{T}_2^{pc}) \le 1$.

4.11 Example Calculations - Stability of Predictor-Corrector Algorithms

The calculations will make use of the system of Figure 9, with identical parameters for this example.

Stability of RA1PC: We first calculate the spectral radius of the RA1PC operator, (Eq. (4.68)), as a function of sample length, Δt, for a fixed value of k* = 700 N/m, with ζ = 0.0, ζ = 0.05, ζ = 0.2, and ζ = 0.3. This is shown in Figure 14. The range of sample lengths is from 0.0001 s to 0.01 sec. The range of natural periods for the model is 0.19 s to 1.6 s.

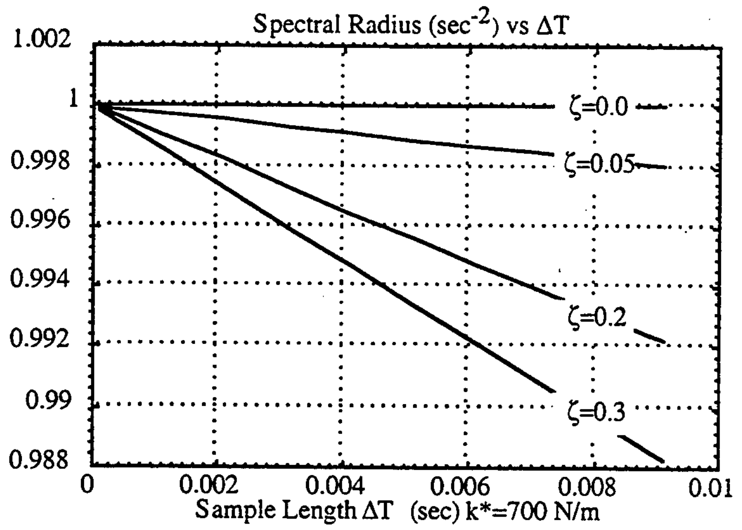

Figure 14. Spectral radius of $\mathbf{T}_I^{pc}$ versus sample length Δt for various ζ

From Figure 14, it is seen that, as compared with the explicit algorithm RA1 (Figure 10), the reformulation of the basic algorithm as an implicit predictor-corrector, using a single corrector step, has provided a substantial improvement in the fundamental stability of the algorithm. The spectral radius decreases with increasing step size, for non-zero system modal damping. The dependence of stability of RA1PC with varying k* is shown in Figure 15, for $\Delta t = 0.01$s.

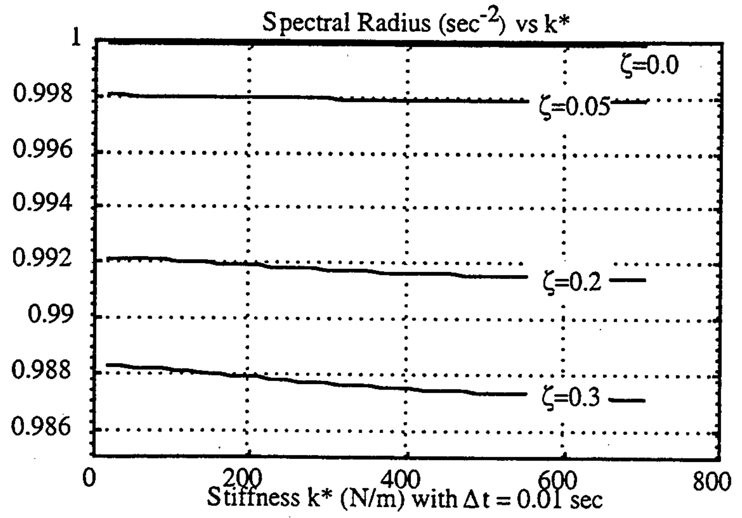

Figure 15. Spectral radius of T_I^{pc} versus sample length K^* for various ζ

Stability of RA2PC: We calculate the spectral radius of the operator for RA2PC as a function of sample length, Δt, for a fixed value of k* = 700 N/m, with $\zeta = 0.0$, $\zeta = 0.05$, $\zeta = 0.2$, and $\zeta = 0.3$. This is shown in Figure 16. The range of sample lengths is from 0.0001 s to 0.01 sec. The range of natural periods for the model is 0.19 s to 1.6 s. The reformulation of RA2 as an implicit algorithm has again provided a substantial improvement in the stability. As seen from Figure 16, the calculation produces results identical to those from RA1PC, again revealing the fundamental similarity of the algorithms, the difference being the use of the complex modal state vector in RA2 versus the traditional modal state vector in RA1. The dependence of stability of RA2PC with varying k* is shown in Figure 17, for Δt = 0.01s, which is again, identical to the analogous calculation for RA1PC.

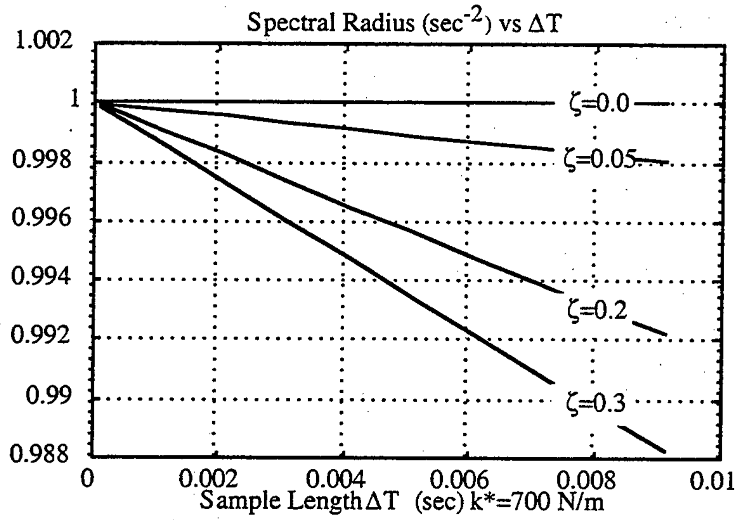

Figure 16. Spectral radius of T_2^{pc} versus sample length Δt for various ζ

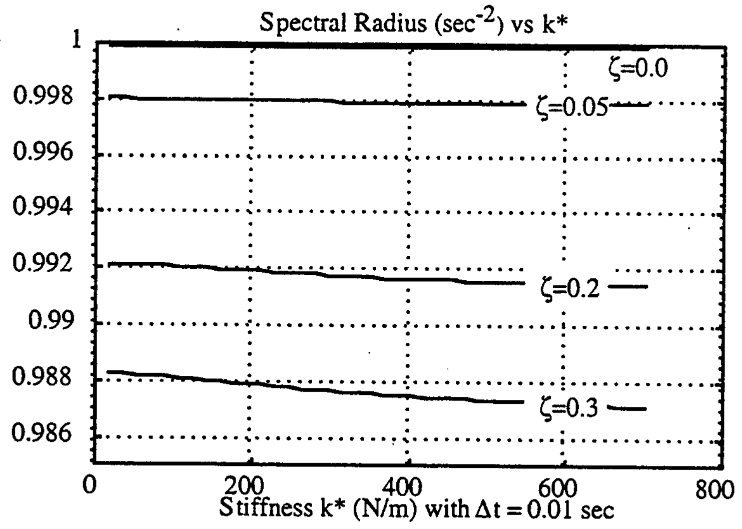

Figure 17. Spectral radius of $\mathbf{T}_2^{pc}$ versus sample length k^* for various

4.12 Transient Synthesis Example - Linear Elastic Modification

The algorithms RA1PC and RA2PC will be used to synthesize the transient response of the system shown in Figure 18. The synthesis will be used to simultaneously replace spring k_1 with k* and calculate the response of mass m_1 to the ground displacement input y(t). The ground motion y(t) is taken as a unit step input.

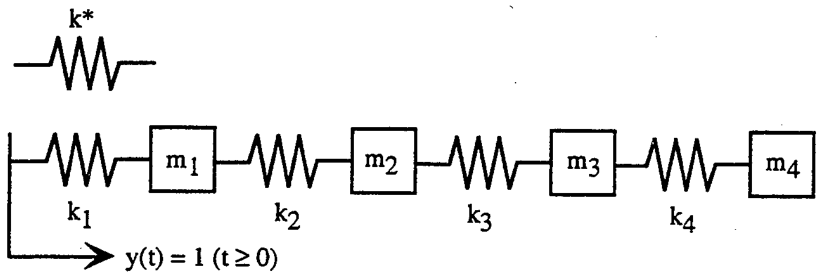

Figure 18. System for synthesis to transient base displacement response with spring replacement

The purpose of this example is two-fold. The first purpose is to demonstrate that a single corrector step taken in algorithms RA1PC and RA2PC, while providing a spectral radius less than unity, does not provide a sufficiently small spectral radius to stabilize the algorithms for a range of stiffness modification magnitudes. The recursive algorithm solutions will be compared with a direct integration (variable time step Runge-Kutte) of the system equations for the modified system, i.e. $k_1 = k^*$. In these calculations, the synthesis is first performed using a value of $k^* = k_2 = k_3 = k_4 = 175.13$ N/m, with $\zeta = 0.0$. The results from RA1PC are shown in Figure 19, and from RA2PC in Figure 20, where both algorithms provide accurate (and identical) results. The next calculations use $k^* = 1.5k_2 = 262.70$ N/m, and the instability is evident in the results from both algorithms, shown in Figures 21 and 22. These calculations with $k^* = 1.5k_2$ are then repeated using $\zeta = 0.05$, and the system modal damping stabilizes the algorithm, as shown in Figures 23 and 24.

The second purpose of this example is to demonstrate the substantial decrease in computer time required by the algorithms based on the complex modal state vector (RA2/RA2PC). Here, algorithms RA1 and RA2 will be compared to a standard modal approach to the solution of this system with k* already installed, as well as to a direct integration of the equations of motion. In other words, we will be comparing the computer time required for a structural modification calculation with that required for a standard analysis, where the structural modification has already been installed during the model assembly phase, i.e. $k_1 = k^*$. As the intended use of these algorithms is for the solution of nonlinear structural synthesis problems, the comparison includes the direct integration solution time. The modal solution is included as a reference time for a standard linear solution. The analysis corresponds to that analysis whose results are shown in Figure 24, but using the explicit form, RA1 and RA2. From the timing results from Table 3, it is seen that the time required for the *synthesis* algorithm RA2 is of the same order of magnitude as the standard linear modal transient analysis. The algorithm RA1 requires a time comparable to the direct integration. Of course, as the model size increases, the direct integration time increases, while the times for RA1 and RA2 are independent of model size, once a modal database has been calculated.

Table 3. Comparison of Times Required for Algorithms

Algorithm:	Direct Integ.	Modal	RA1	RA2
Time (sec)	1.130	0.103	1.019	0.175

Stability Comparison: Transient Base Displacement Response Δt = 0.01 sec

Direct Integration — —   Synthesis ——

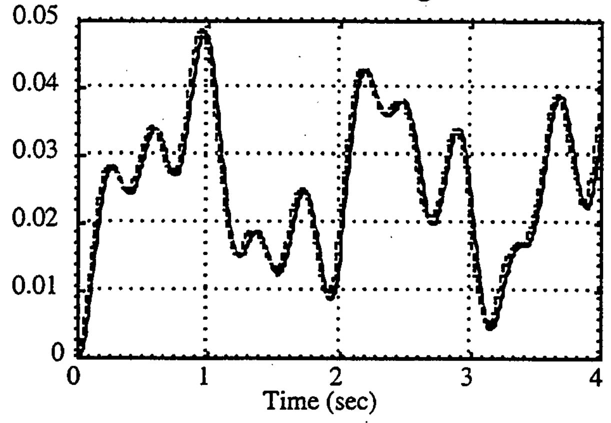

Figure 19. RA1PC: k* = 175.13 N/m, ζ = 0.0

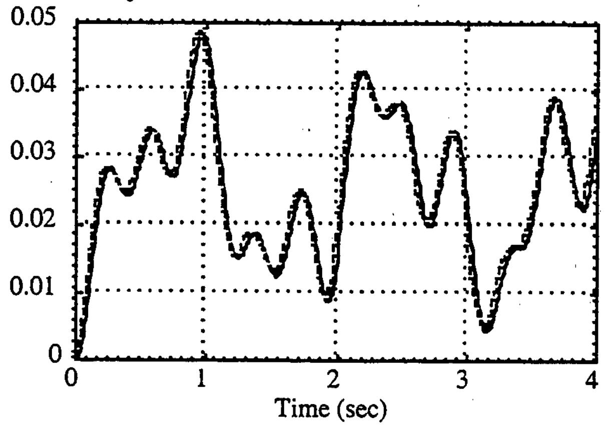

Figure 20. RA2PC: k* = 175.13 N/m, ζ = 0.0

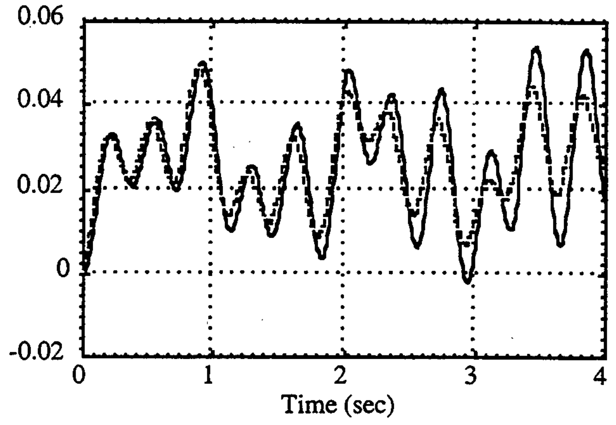

Figure 21. RA1PC: k* = 262.70 N/m, ζ = 0.0

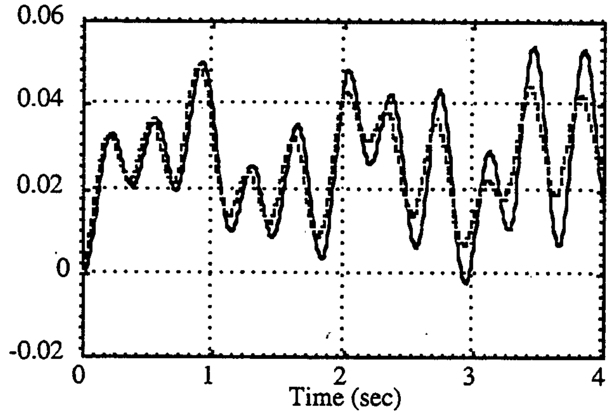

Figure 22. RA2PC: k* = 262.70 N/m, ζ = 0.0

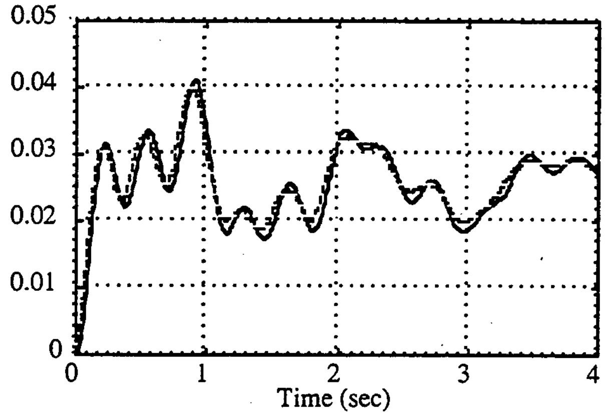

Figure 23. RA1PC: k* = 262.70 N/m, ζ = 0.05

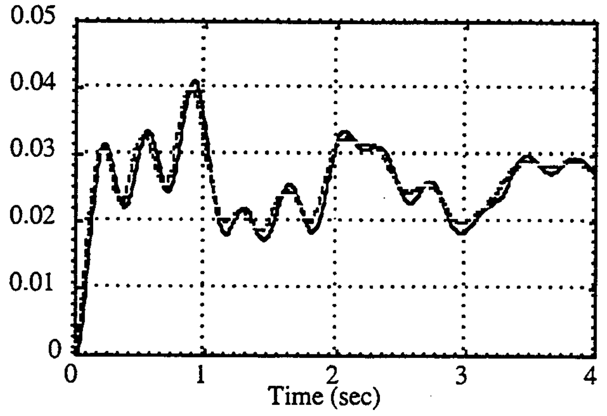

Figure 24. RA2PC: k* = 262.70 N/m, ζ = 0.05

4.13 Summary of Results for Transition Matrix-Based Recursive Algorithms: Explicit and Predictor-Corrector

The goal of this work is the development of an efficient and robust algorithm for the solution of the governing integral equation for locally nonlinear transient structural synthesis. The new algorithm must preserve the existing features of the existing formulation. The features include,

• a solution time which is independent of model size; an exact model reduction which is essentially unrestricted in the dimension of the reduction is implicit in the formulation,

• the ability to perform local structural modifications of general nonlinear characteristics

• accurate and fast solutions facilitating optimal design

Two general algorithms have developed and analyzed for stability. Both algorithms are based on modal transition matrices, facilitating the recursive forms of these algorithms. The first algorithm, referred to as Recursive Algorithm #1 (RA1), is based on the standard modal state vector and the associated transition matrix, $e^{A\Delta t}$. The second algorithm is based on a newly defined *complex modal state* vector, and also defines a new transition matrix formulation based exclusively on the second-order modal differential equations. This algorithm is referred to as Recursive Algorithm #2 (RA2). It was shown that while the formulations of these algorithms are different, RA1 being a real-arithmetic algorithm and RA2 being complex, the algorithms produce identical numerical results. However, RA2 allows a greater level of "diagonalization" relative to RA1, and hence yields computing times of an order of magnitude less than that required by RA1 (see Table 1).

The inherent instability of algorithms RA1 and RA2 was addressed by their reformulation as predictor-corrector algorithms, using a linear interpolation for the modal force, and a single corrector step. This reformulation was successful in that it demonstrated that the limiting value of unity for the spectral radii of RA1 and RA2 could be removed, and that spectral radii less than unity are achievable. However, the study performed indicated that the degree of stability provided by this basic reformulation is not sufficient to provide a robust algorithm. Therefore, rather than continue with the development of these recursive predictor-corrector algorithms, this work will continue with the development of a recursive algorithm which is not based on a transition matrix, but rather on a block-by-block convolution algorithm. It will be shown in the next section that this block-by-block convolution algorithm preserves all the features of the existing synthesis formulation, and provides a dramatic decrease in the computing time required. This algorithm is considered a non-modal formulation, as it is based on the use of physical impulse response to represent the (linear) substructures, which are subjected to structural modification, substructure coupling, and ground-motion input through isolation. This algorithm will be shown to be exponentially convergent, regardless of the linear substructure properties, and for a general and broad class of nonlinear modifications.

REFERENCES

[1] Macneal, R. H., 1972. The NASTRAN Theoretical Manual. Macneal-Schwendler Corporation. Ch. 11.

[2] Gordis, J. H. 1995. "*Integral Equation Formulation for Transient Structural Synthesis*" AIAA Journal, Vol. 33, No. 2, pp. 320-324.

[3] Gordis, J. H. and Radwick, J. L. "Efficient Transient Analysis for Large Locally Nonlinear Structures", Shock and Vibration, Vol. 6, No. 1, 1999.

[4] Gordis, J. H. 1994. "*Structural Synthesis in the Frequency Domain: A General Formulation*" Shock and Vibration, Volume 1, Issue 5, pp. 461-471.

[5] Gordis, J. H. and Flannelly, W. G. 1994. "*Analysis of Stress due to Fastener Tolerance in Assembled Components*" AIAA Journal, Vol. 32, No. 12, pp. 2440-2445.

[6] Gordis, J. H. Bielawa, R. L. Flannelly, W. G. 1991. "*A General Theory for Frequency Domain Structural Synthesis*" Journal of Sound and Vibration 150(1), pp. 139-158.

[7] Inaudi, J. A. and De La Llera, J. C. 1992. 'Dynamic Analysis of Nonlinear Structures Using State-Space Formulation and Partitioned Integration Schemes." University of California-Berkeley Earthquake Engineering Research Center Report No. UCB/EERC-92/18.

[8] Blakely, K. Basic Dynamic Analysis User's Guide. Macneal-Schwendler Corp. 1993.

[9] Linz, P. 1985. Analytical and Numerical Methods for Volterra Equations. Society for Industrial and Applied Mathematics.

[10] Kreyszig, E. Introductory Functional Analysis with Applications. John Wiley & Sons,

[11] Milne, R. D., Applied Functional Analysis. Pitman Advanced Publishing Co. 1980.

[12] A. J. Jerri, Introduction to Integral Equations with Applications, Marcel Dekker, Inc., New York, 1985

[13] Strang, G. and Nguyen, T., Wavelets and Filter Banks. Wellesley-Cambridge Press. 1996.

[14] Belytschko, T. and Hughes, T.J.R., 1983. Computational Methods for Transient Analysis. Volume 1 of Series, Computational Methods in Mechanics, North-Holland.

[15] Hughes, T.J.R., 1987. The Finite Element Method, Ch. 9.Prentice-Hall.

[16] Newmark, N. M., 1959 "*A Method of Computation for Structural Dynamics*," Journal of the Engineering Mechanics Division, Proceedings of the ASCE, Vol. 85, EM3, 1959, pp. 67-94.

[17] Meirovitch, L., 1997. Principles and Techniques of Vibrations. Prentice-Hall, pp. 40-42. 1978.

[18] Fromme, J. A. and Golberg, M. A., 1997. "*Exponential Convergence of Picard Iteration for Integrating Linear and Nonlinear Modally Coupled Equations of Motion.*" Proceedings of AIAA/ASME/ASCE/AHS/ACS 38th Structures, Structural Dynamics, and Materials Conference, Kissimmee, FL, 1997.

DISTRIBUTION LIST

1.	Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library, Code 013 Naval Postgraduate School Monterey, CA 93943-5100	2
3.	Research Office, Code 09 Naval Postgraduate School Monterey, CA 93943-5138	1
4.	The National Science Foundation Earthquake Hazard Mitigation Program Dr. S. C. Liu, Program Director 4201 Wilson Blvd. Room P60 Arlington VA 22230	2
5.	Dr. Joshua H. Gordis Associate Professor of Mechanical Engineering Code ME/Go Naval Postgraduate School Monterey, CA 93943-5146	5
6.	Dr. Beny Neta Professor of Mathematics Code MA/Nd Naval Postgraduate School Monterey, CA 93943-5100	1
7.	Mr. Dana R. Johansen Ship Survivability and Structures Technology Support Division - SEA 03P4 Building NC4 Naval Sea Systems Command 2531 Jefferson Davis Highway Arlington, VA 22242-5160	1
8.	Dr. William Gottwald NSWC - Carderock Division Code 671 Bethesda, MD 20084-5000	1