Fast Transient Analysis for Locally Nonlinear Structures by Recursive Block Convolution

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 93943-5146

ABSTRACT

The transient analysis of large structural systems with localized nonlinearities is a computationally demanding process, inhibiting dynamic redesign and optimization. A previously developed integral equation formulation for transient structural synthesis has demonstrated the ability to solve large locally nonlinear transient problems in a fraction of the time required by traditional direct integration methods, with equivalent or better accuracy. A recursive block-by-block convolution algorithm is developed for the solution of the governing integral equations which further reduces the solution times required. Examples using realistically-sized finite element models are presented, demonstrating the performance of the formulation.

NOMENCLATURE

D	delay matrix
f	excitations
h	filter matrix
H	impulse response function matrix
L	Lipschitz constant
R	boolean coupling matrix
r	row extraction matrix
t,τ	time
W	quadrature weights
x	displacement response
Subscripts/Superscripts:	
b,c,i,m	coordinate sets
e	external

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."

The approach in this work, originally reported in [1,2], 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 a broader category of physical coordinate *structural synthesis* methods [1-5], which includes substructure coupling, base excitation through generalized elements, 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.

While structural synthesis treats the nonlinear element responses as applied loads, in a manner similar to other methods for local nonlinear transient analysis, 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,
- general nonlinearities can be treated,
- the linear portion of the model is solved once,
- very fast solution times are obtained, an intrinsic property of the formulation.

The governing equation for transient structural synthesis is a nonlinear Volterra integral equation, involving a convolution-type kernel [1,2]. The convolution-type kernel suggests a recursive transition-matrix approach to the solution of first-order ordinary differential equations (e.g. [6]) as a potential improvement over the (non-recursive) iteration solution presented by Gordis and Radwick [2], in which an order-of-magnitude reduction in computing time required was demonstrated, relative to direct integration. However, in [6], 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. Furthermore, as is shown in [7], recursive modal transition matrix approaches, while providing a model reduction, are inherently unstable in explicit forms, and are not easily stabilized in implicit forms. We must therefore consider such an approach to be of limited value for large structural models.

The current recursive algorithm developed differs from previously developed recursive algorithms in that no transition matrix is employed. The current algorithm preserves the physical coordinate formulation originally developed by Gordis [1] and Gordis and Radwick [2], and hence preserves the implicit and unrestricted exact model reduction, concomitant with the formulation. The algorithm is exponentially convergent, for a general class of nonlinearities [7].

2. Coordinate Sets and Impulse Response Functions

We provide highlights in the relevant theory. The reader is referred to [1,2,7] for the complete development.

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

$$\mathbf{x}(t) = \mathbf{x}_h(t) + \int_0^t \mathbf{H}(t-\tau)\mathbf{f}(\tau)d\tau, \qquad (1)$$

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

$$\mathbf{x}(t) = \left[\mathbf{x}_i(t)^T \quad \mathbf{x}_c(t)^T \quad \mathbf{x}_m(t)^T \quad \mathbf{x}_b(t)^T\right]^T \qquad (2)$$

$$\mathbf{f}(t) = \left[\mathbf{f}_i(t)^T \quad \mathbf{f}_c(t)^T \quad \mathbf{f}_m(t)^T \quad \mathbf{f}_b(t)^T\right]^T \qquad (3)$$

Note that the above coordinate sets are each comprised for coordinates from any number of substructures, as shown in Figure 1.

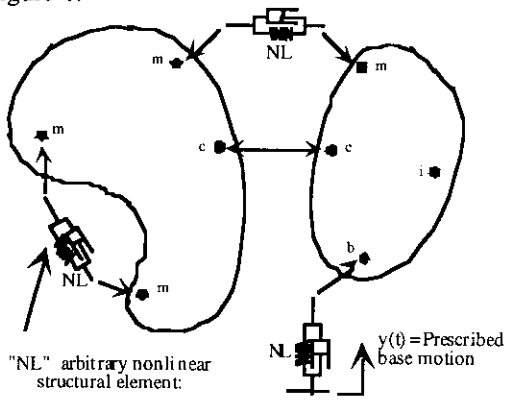

Figure 1. System for synthesis comprised of two substructures

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. The DOF sets are:

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

The synthesis provides a transient analysis that is independent of model size, in that only those structural DOF of interest need be included. 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. 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.

Each substructure is described by impulse response functions (IRF) calculated at the coordinates subjected to forces of synthesis (m-set, b-set, c-set), at other DOF for which synthesized nonlinear transient response is required (i-set), and where external loads are applied. 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.

The matrix **H** is the impulse response function (IRF) matrix, any element of which can be written as,

$$H_{ij}(t) = \sum_{p=1}^{r} \phi_i^p \phi_j^p t + \sum_{p=r+1}^{n} \frac{\phi_i^p \phi_j^p}{\omega_{dp}} e^{-\zeta_p \omega_p t} \sin\left(\omega_{dp} t\right) \qquad (4)$$

where ϕ_i^p is the i^{th} element of the p^{th} mass-normalized eigenvector of the substructure prior to synthesis, ω_p and ω_{dp} are the p^{th} undamped and damped natural frequencies, respectively, ζ_p is the p^{th} modal damping ratio, r is the number of rigid body modes, and $n \le N$ 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.

3. Governing Equation of Nonlinear Transient Synthesis

The governing equation for structural synthesis is

$$\mathbf{x}^*(t) = \mathbf{x}(t) - \int_0^t \mathbf{H}(t-\tau)\mathbf{f}^*\left(t, \tau, \mathbf{x}^*(\tau), \dot{\mathbf{x}}^*(\tau)\right)d\tau \qquad (5)$$

where $\mathbf{x}(t)$ contains both the initial displacement and response due to externally applied excitations,

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \mathbf{H}(t-\tau)\mathbf{f}^e(\tau)d\tau \qquad (6)$$

and $\mathbf{f}^*(t)$ are the synthesized reactions acting on all DOF sets,

$$\mathbf{f}^*(t) = \left[0^T \quad \left(\mathbf{R}\tilde{\mathbf{f}}_c^*(t)\right)^T \quad \mathbf{f}_m^*(t)^T \quad \mathbf{f}_b^*(t)^T\right]^T \qquad (7)$$

where **R** is a boolean matrix reflecting the equilibrium which exists between the coupled DOF

Equation (5) 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 yielding exponential convergence properties [7].

4. Iterative Solution: Uniqueness and Convergence Results

The following results are excerpted from [7]. The recursive block-by-block convolution algorithm is iterative, and hence we are concerned with the boundedness of a sequence of solutions,

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

We require that the forces of synthesis satisfy a Lipschitz condition,

$$\left\|\mathbf{f}_n^*(t,\tau)\right\| \le L\left\|\mathbf{x}_n^*(\tau) - \mathbf{x}_{n-1}^*(\tau)\right\| \qquad (8)$$

where L is a positive constant. Using an inductive argument the following is established

$$\left\|\mathbf{x}_{n+1}^*(t) - \mathbf{x}_n^*(t)\right\| \le \frac{\left|x_u - x_l\right|}{(n-1)!}\left(Lt\left\|\mathbf{H}(t)\right\|\right)^{n-1} \qquad (9)$$

where x_u and x_l are upper and lower bounds on the system response, which establishes the uniform exponential convergence of the series,

$$\sum_{n=1}^{\infty} \left\|\mathbf{x}_{n+1}^*(t) - \mathbf{x}_n^*(t)\right\| \qquad (10)$$

with no restriction on L, t, or $\|\mathbf{H}(t)\|$.

5. Numerical Quadrature for Nonlinear Volterra Integral Equations

The numerical solution of Eq. (5) 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. (5) becomes,

$$\mathbf{x}^*(i\Delta t) = \mathbf{x}(i\Delta t) - (\Delta t)^{\alpha} \sum_{j=0}^{i-\beta} w_j \mathbf{H}\left((i-j)\Delta t\right)\mathbf{f}^*(j\Delta t) \qquad (11)$$

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. (11) becomes ,

$$\mathbf{x}^*(0\Delta t) = \mathbf{x}(0\Delta t) \qquad (12)$$

$$\mathbf{x}^*(1\Delta t) = \mathbf{x}(1\Delta t) - \Delta t\left[\mathbf{H}(1\Delta t)\mathbf{f}^*(0\Delta t) + \mathbf{H}(0\Delta t)\mathbf{f}^*(1\Delta t)\right] \tag{13}$$

$$\mathbf{x}^*(2\Delta t) = \mathbf{x}(2\Delta t) - \ldots$$
$$\ldots\Delta t\left[\mathbf{H}(2\Delta t)\mathbf{f}^*(0\Delta t) + \mathbf{H}(1\Delta t)\mathbf{f}^*(1\Delta t) + \mathbf{H}(0\Delta t)\mathbf{f}^*(2\Delta t)\right] \tag{14}$$

and we note that $\mathbf{H}(t=0) = 0$, yielding the correct series for the rectangular rule. It is important to recognize that the bracketed terms in Eqs. (12), (13), and (14) are equivalent to those produced by the discrete convolution.

The trapezoid rule and Simpson's rule are commonly used quadratures for this application [9,10]. The performance of the trapezoid rule in the solution of the linear synthesis problem is reported in [2].

6. 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

$$\mathbf{x} * \mathbf{y} = \sum_k x(n-k)y(k) \tag{15}$$

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) [11]:

$$\mathbf{z} = \mathbf{x} * \mathbf{y} = \mathbf{h}(\mathbf{x}) \cdot \mathbf{y} = \ldots$$
$$\ldots = \begin{bmatrix} x_1 & 0 & \cdots & & \cdots & 0 \\ x_2 & x_1 & 0 & \cdots & \cdots & \vdots \\ \vdots & x_2 & x_1 & 0 & \cdots & \\ \vdots & \vdots & & \ddots & \ddots & \vdots \\ x_{n-1} & x_{n-2} & \cdots & \ddots & x_1 & 0 \\ x_n & x_{n-1} & x_{n-2} & \cdots & x_2 & x_1 \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{n-1} \\ y_n \end{Bmatrix} \tag{16}$$

and the elements of x are referred to as filter weights [11]. 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.

We now define a delay matrix **D** [11] with the following structure:

$$\mathbf{D} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & 0 & 0 & 0 & \cdot \\ \cdot & 1 & 0 & 0 & 0 & \cdot \\ \cdot & 0 & 1 & 0 & 0 & \cdot \\ \cdot & 0 & 0 & 1 & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \tag{17}$$

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**,

$$\mathbf{D} \cdot \mathbf{x} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ x_1 \\ x_2 \end{Bmatrix} \tag{18}$$

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

The filter matrix **h** is equal to the summation of powers of the delay matrix multiplied by the filter weights, x_j. 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 j^{th} column of **h** is given by $\mathbf{D}^j\mathbf{x}$. The filter matrix of a vector **x** of length n is therefore,

$$\mathbf{h}(\mathbf{x}) = \sum_{j=0}^{n-1} x_j \cdot \mathbf{D}^k = \left[\mathbf{D}^0\mathbf{x} \quad \mathbf{D}^1\mathbf{x} \quad \cdots \quad \mathbf{D}^{n-2}\mathbf{x} \quad \mathbf{D}^{n-1}\mathbf{x}\right] \tag{19}$$

7. 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 $J\Delta t$ 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**:

$$\mathbf{r} = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & & \\ \vdots & & & 0 & 1 & 0 & & \\ & & & & 0 & 1 & 0 & \\ \vdots & & & & & 0 & 1 & 0 \\ 0 & \cdots & & & & \cdots & 0 & 1 \end{bmatrix} \tag{20}$$

The product of the matrix $\mathbf{r}$ with a vector $\mathbf{x}$ is the subvector of $\mathbf{x}$ consisting of the rows (samples) of the K^{th} 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 $\mathbf{D}$, we can define a matrix which extracts the rows of the k^{th} block, where k = 1,2,...,K.

$$\mathbf{r}_k = \mathbf{r} \cdot \mathbf{D}^k = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & & & \cdots & 0 \\ \vdots & & & 0 & 1 & 0 & & & & \vdots \\ & & & & 0 & 1 & 0 & & & \\ \vdots & & & & & 0 & 1 & 0 & & \vdots \\ 0 & \cdots & & & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \tag{21}$$

The matrix equation, Eq. (16), can be written in a block-partitioned form as follows. We can write the k^{th} subvector of z, i.e. z_k, as,

$$\mathbf{z}_k = \sum_{m=1}^{k} \mathbf{r}_k \cdot \mathbf{h}(\mathbf{x}) \cdot \mathbf{r}_m^T \cdot \mathbf{y}_m \tag{22}$$

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:

$$\mathbf{r}_k \cdot \mathbf{h}(\mathbf{x}) \cdot \mathbf{r}_m^T = \begin{cases} x(1:J) & \text{if } k = m \\ x\big((k-m-1)J+2:(k-m+1)J\big) & \text{if } k > m \end{cases} \tag{23}$$

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

8. 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. The number of FLOPS for the BBB algorithm is given by:

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

which yields an optimum number of blocks 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 1, 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.

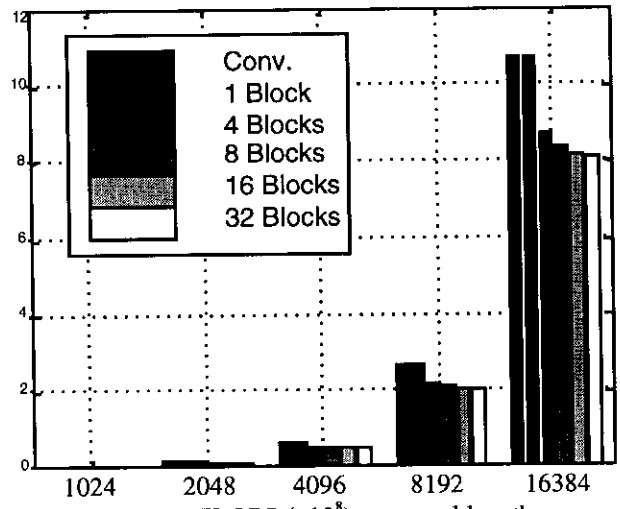

Figure 1. FLOPS ($x10^8$) vs record length

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 number outweighs the decrease in computing time due to the reduction in FLOPS required. This is shown in Figure 2, and 8 blocks is the minimum.

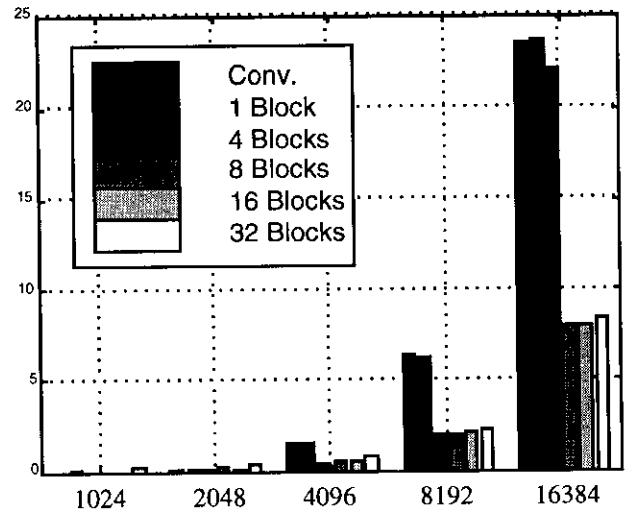

Figure 2. Compute time (sec) vs. record length

9. 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. (5). In the algorithm which follows, it is implied that the vector $\mathbf{x}$ is partitioned consistently with Eq. (2), 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 $s = m \cup c \cup b$ 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: $j \Leftarrow 1 \quad \mathbf{f}_s^j \Leftarrow \mathbf{1}$

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" subscript 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 k^{th} block, and at the j^{th} iteration, as $\mathbf{x}_k^j$ and $\mathbf{f}_k^j$. The IRF filter matrix for the k^{th} block is denoted as $\mathbf{H}_k$, and is given by

$$\mathbf{H}_{km} = \mathbf{r}_k \cdot \mathbf{H} \cdot \mathbf{r}_m^T \qquad (24)$$

There are K blocks, k = 1,2,...,K, and each block is of length J (samples). We will make use of Eq. (24) 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.(23). The iteration for the k^{th} block is given by,

$$\mathbf{x}_k^{j+1} = \mathbf{x}_k - \sum_{m=1}^{k-1} \left(\mathbf{H}_{km} \cdot \mathbf{f}_m\right) - \mathbf{H}_{kk} \cdot \mathbf{f}_k^j \qquad (25)$$

Recursive Block-by-Block Algorithm

Initialize: $j \Leftarrow 1$, $\mathbf{f}^j \Leftarrow \mathbf{1}$ (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: $\mathbf{x}_i^* = \mathbf{x}_i - \mathbf{H}_{is} * \mathbf{f}_s^*$

10. Performance of Block Algorithm

The algorithm will be applied to the nonlinear base isolation problem shown in Figure 3. In this problem, a deck model of approximately 51,500 DOF supports a piece of equipment (lumped mass). The ground motion time history is shown in Figure 4.

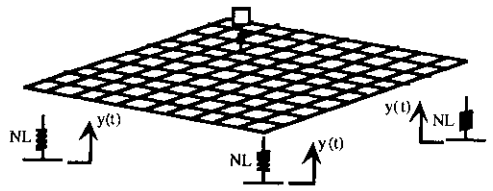

Figure 3. Isolated Equipment on Deck

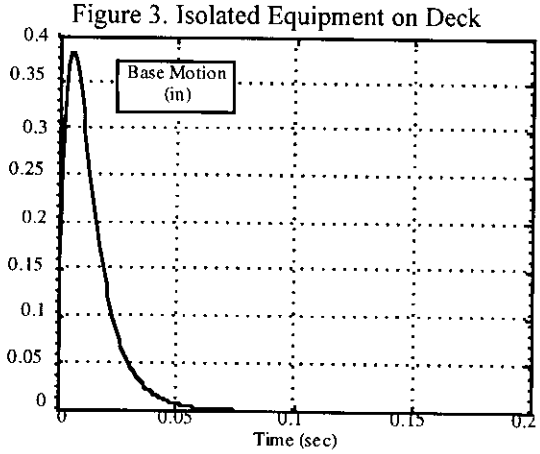

Figure 4. Ground motion time history

The isolators are comprised of a cubically hardening spring, i.e. $f(z) = kz + k_3 z^3$, where z is the relative displacement x – y, at the four corners of the deck. The block-by-block synthesis solution will be compared to a direct integration using a widely used commercial finite element program. The FE solution will be referred to as the "Direct FE" solution. We will compare the time histories for one of the corner points, and for the supported equipment, as calculated using the synthesis and Direct FE.

An eigensolution of the free-free (linear) deck model was performed to generate a modes database from which IRF are calculated for the synthesis. All modes under 12,000 Hz were calculated (99 modes). This modes solution took 7 minutes 47 seconds. These modes are not used by the Direct FE solution.

The actual nonlinear direct transient analysis of the isolated deck model (cubically hardening springs) was performed in 40 minutes 46 seconds.

As can be seen from Figures 5 and 6, the block by block synthesis provides a very accurate solution, taking the Direct FE as the reference.

We now tabulate the solution times for the block-by-block algorithm, for different numbers of blocks. This is shown in Table 1. Note that the time to load the modes database is not included in the synthesis times tabulated.

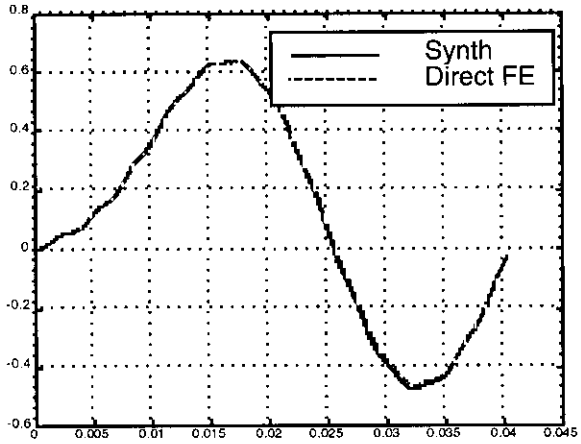

Figure 5. Corner vertical displacement vs time
Displacement (in) Time (sec)

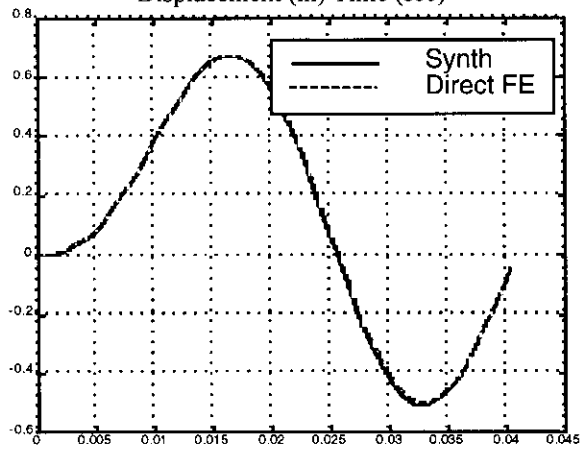

Figure 6. Mass vertical displacement vs time
Displacement (in) Time (sec)

Table 1. Synthesis Times Vs. Number of Blocks

# Blocks	1	2	4	8	16
Modes Min:Sec	7:47	7:47	7:47	7:47	7:47
Synth. (seconds)	6.71	5.87	2.21	2.03	3.04
Total Min:Sec	7:54	7:53	7:49	7:49	7:50

Total Time for Direct FE: 40 minutes 46 seconds

Clearly, the Block-by-Block algorithm is extremely fast compared with Direct FE.

Conclusions

A new recursive block-by-block convolution algorithm has been developed for the solution of the governing nonlinear Volterra integral equation for locally nonlinear structural synthesis. The new algorithm is extremely fast, as compared with direct integration, and is also must faster than the previously reported algorithm [2]. The algorithm lends itself for use in nonlinear structural dynamic optimization.

Acknowledgements

This work is dedicated to Joseph Robert Gordis, in honor of his second year. The authors gratefully acknowledge the support of the National Science Foundation, #9713481.

References

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

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

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

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

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

[6] Inaudi, J. A. and De La Llera, J. C. *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, 1992.

[7] Gordis, J. H. and Neta, B. *Efficient Nonlinear Transient Dynamic Analysis for Structural Optimization Using an Exact Integral Equation Formulation,* Naval Postgraduate School Technical Report. 1999.

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

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