A SIMPLE LMS-BASED APPROACH TO THE STRUCTURAL HEALTH MONITORING BENCHMARK PROBLEM

A thesis submitted in partial fulfilment of the requirements for the

Degree

of Master of Mechanical Engineering

in the University of Canterbury

by Kyu-Suk Leo Hwang

University of Canterbury

2003

ABSTRACT

A civil structure's health or level of damage can be monitored by identifying changes in structural or modal parameters. However, modal parameters can be less sensitive to (localised) damage than directly identifying the changes in physical parameters of a structure. This research directly identifies changes in structural stiffness due to modelling error or damage, when the structural state and a reasonable estimate of the input force are available, such as earthquake or roof loading of a sensored steel frame structure. This thesis presents the development and implementation of a health monitoring method based on Adaptive Least Mean Square (LMS) filtering theory. The focus in developing these methods is on simplicity to enable real-time implementation with minimal computation. Several adaptive LMS filtering approaches are used to analyse the data from the International Association for Structural Control and American Society of Civil Engineers Structural Health Monitoring (SHM) Task Group Benchmark problem. Results are compared with those from the Task Group and other published results. The proposed methods are shown to be very effective, accurately identifying damage to within 1%, with convergence times of 0.4 – 13.0 seconds for the twelve different 4 and 12 degrees-of-freedom SHM Benchmark problems. The resulting modal parameters match to within 1% those from the SHM Benchmark problem definition. Finally, the method presented is computationally extremely simple, requiring no more than 1.4 Mega-cycles of computation, and therefore could easily be implemented in real-time.

ACKNOWLEDGMENTS

I would like to truthfully thank Dr. Geoff Chase, my supervisor, for his guidance and support at the every stage of not only this work, but also my university career. This research may never have been productive without his willingness to share his experience and knowledge.

I would like to thank my fellow post-grads, Mark Carey, Zhu Hui Lam, Joo-Young Lee, and Andrew Shaw, for their willingness to help, encouragement and suggestions during the development of this work. I also wish to thank my friends, Sam, Yoon-Shik, Sang-Sub, Sang-Min and Dong-Hoon, whose unconditional friendship made living life more special and memorable in Christchurch.

Finally, I would like to thank to Somi, my beloved girl friend, her trust, encouragement and patience from the very beginning of my university life. I would also like to thank my family, Mom, Dad and my brother, Kyu Won for their support and guidance throughout my University period.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathbf{C}$	damping matrix
c_i	damping coefficient of i^{th} story
e_k	error between measured noisy signal and modelled value at time k
$\mathbf{F}$	input force
i	integer; story number
j	integer; tap number
k	integer; time index
$\mathbf{K}$	stiffness matrix
k_i	stiffness of i^{th} story
$\mathbf{M}$	mass matrix
m	integer; number of taps or prior time steps
m_i	mass of i^{th} story
n	integer; number of input signals or degree-of-freedom
v	displacement of the baseline model (undamaged)
$\dot{v}$	velocity of the baseline model (undamaged)
$\ddot{v}$	acceleration of the baseline model (undamaged)
$\bar{v}$	displacement of the damaged model
$\dot{\bar{v}}$	velocity of the damaged model
$\ddot{\bar{v}}$	acceleration of the damaged model
w_k	adjustable coefficient matrix
$w_0(i)$	initial weight for i^{th} story
$w_k(i)$	adjustable weight for i^{th} story at time k

W_k	adjustable filter coefficient vector at time k
$\ddot{x}_g$	ground acceleration
X_k	input vector to a filter at time k
y_k	measured vector noisy signal at time k
$\hat{y}_k$	measured scalar noisy signal at time k
$\bar{y}_k$	filter approximation to y_k at time k
α_i	stiffness coefficient of ΔK_i
α_k	vector of coefficients of α_i at time k
α_{ij}	i^{th} row and j^{th} column element of w_k
Δk_i	change in stiffness of i^{th} story
$\Delta\mathbf{K}$	change in stiffness matrix
ΔK_i	i^{th} sub-matrix of $\Delta\mathbf{K}$
μ	adaptive LMS coefficient
∇MSE	gradient of mean square error
$\nabla MSE_{\alpha_{ij}}$	gradient of mean square error with respect to α_{ij}

1. INTRODUCTION

1.1. Motivation

Structural Health Monitoring (SHM) is the process of examining the current state of a structure's condition and determining the existence, location, and degree of damage that may exist, particularly after a damaging input, such as an earthquake or other large environmental load. Current SHM methods are based on the idea of vibration-based damage detection where changes in modal parameters, such as frequencies, mode shapes and modal damping, are a result of changes in the physical mass, damping and stiffness properties of the structure (Doebling et al, 1996). SHM can simplify typical procedures of visual or localized experimental methods, such as acoustic or ultrasonic methods, magnetic field methods, radiography, eddy-current methods or thermal field methods (Doherty, 1997), as it does not require visual inspection of the structure and its connections or components. Doebling et al (1996a) has an excellent review of the numerous different approaches for vibration-based damage detection methods. However, the various studies apply different methods to different structures, rendering side-by-side comparison difficult.

In 1999, under the auspices of the International Association for Structural Control (IASC) and the Dynamics Committee of the American Society of Civil Engineers (ASCE) Engineering Mechanics Division, the SHM Task Group was formed and charged with studying the efficacy of various SHM methods. The IASC-ASCE SHM Task Group developed a series of Benchmark SHM problems and established a set of specific

Benchmark results for a specially designed test structure in the Earthquake Engineering Research Laboratory at the University of British Columbia (Johnson et al, 2000). After the Benchmark problem was established, SHM research for civil structures was concentrated on applying different techniques to the Benchmark problem to examine the relative and absolute effectiveness of different algorithms.

SHM in civil structures is very useful for determining damage state of a structure. In particular, the ability to assess damage in real-time or immediately after an earthquake would allow Civil Defence authorities to determine which structures were safe. Current methods are more applicable to steel frame or bridge structures where vibration response may be more linear under ambient vibrations. These problems typically have known, or reasonably estimated, input loads. However, the insensitivity of modal parameters to (localised) damage in some cases can be a major limitation for the larger number of methods that rely on identifying these parameters to assess and locate damage. This research uses adaptive filtering to assess the damage directly without using modal parameters. This approach promises a computationally efficient sample-to-sample method of directly identifying damage.

1.2. Objective

The primary objective of this research is to develop simpler and more efficient algorithms than existing methods for continuously monitoring structural status. This information is especially important during or after earthquakes, or any other form of damaging environmental excitation. A specific goal is an algorithm simple enough to be implemented in real-time, adaptive and LMS filtering theory is employed as an appropriate algorithm to achieve the goal. To achieve the primary objective of this research, the following intermediate stages are undertaken:

- To study the literature on SHM and adaptive LMS filtering
- To develop an analytical model of the SHM Benchmark structure
- To develop an adaptive LMS filter based method for SHM
- To verify the effectiveness of the algorithm via simulation
- To compare the simulation results with the Task Group Benchmark results and other published results
- To investigate the ability to implement the developed method in real time with the current technology

1.3. Literature Survey

Most of the modern developments in vibration-based damage detection stem from studies performed in the 1970s and early 1980s by the offshore oil industry (Doebling et al, 1996a). However, most of the early proposed techniques were less successful. Research in vibration-based damage identification has been rapidly expanding over the last few years. The literature study was concentrated on recently developed methods, particularly those corresponding to the SHM Task Group Benchmark problem.

The most common method for identification of civil structural model parameters is the Eigensystem Realization Algorithm (ERA). The ERA method is based on knowledge of the time domain free response data. In ERA, a discrete Hankel matrix is formed, and the state and output matrices for the resulting discrete matrix are determined. These matrices are transformed to the corresponding continuous time system. The natural frequencies are found by determining the eigenvalues of the continuous time system. Dyke et al (2000) use cross correlation functions in conjunction with the ERA method for identification of the modal parameters, which are used to identify frequency and damping parameters. Caicedo et al (2000) introduces SHM methods based on changes in the component transfer functions of the structure, or transfer functions between the floors of a structure, and use the ERA to identify the natural frequencies of each component transfer function. Lus and Betti (2000) also proposed a damage identification method based on ERA with a Data Correlation and Observer/Kalman Identification algorithm. Bernal and Gunes (2000) also used the ERA with Observer/Kalman Identification for identifying modal characteristics when the input is known, and used a Subspace Identification algorithm when the input cannot be measured.

Wavelet analysis approaches for SHM and damage detection may be found in Corbin et al (2000) and Hou et al (2000). Damage, and the moment when the damage occurs, can be detected by a spike or an impulse in the plots of higher resolution details from wavelet decomposition of the acceleration response data. Wavelets offer the advantage of determining not only the extent of the damage but also the time of its occurrence, which can be correlated to the input record for greater understanding of what occurred.

The major drawback of all of these approaches is their inability to be implemented in real-time, as the event occurs. More specifically, the wavelet and ERA methods require the entire measured response to process and identify damage. Further, their reliance on modal properties, which can be subject to noise, has potential problems. In addition, modal properties have been shown in some cases, to be non-robust in the presence of strong noise and insensitive to small amounts of damage (Hou et al, 2000).

Adaptive identification methods were employed to identifying modal parameters by Sato and Qi (1998) and Loh et al (2000). Loh et al (2000) used the adaptive fading Kalman filter technique, and Sato and Qi (1998) an Adaptive H_∞ Filter, to achieve real-time capable or near real-time capable results. What these approaches provide in real-time identification of modal parameters comes with significant computational cost and complexity. These methods can be sensitive to noise but typically account for it directly in the formulation. Hence, this research looks for the simplest possible algorithm while sustaining real-time capability.

1.4. Overview

This thesis presents the development of a much simpler and efficient algorithm than existing methods for continuously monitoring the status of a steel frame structure. Adaptive LMS filtering is employed for its overall simplicity and resultant ability to be implemented in real-time. A procedure is presented for developing adaptive LMS filtering based methods for SHM. This task is accomplished by taking advantage of this filter's ability to adaptively model noisy signals to identify changes in structural parameters in comparison to a base structural model. The algorithms developed consist of a series of coupled adaptive LMS filters and are tested on the Benchmark problem test cases. Results are also compared with those presented in the literature to further verify the method.

2. STRUCTURAL HEALTH MONITORING

2.1. Definition

A seismically excited structure can be modelled using standard linear equations of motion, or a more complex computational model. Consider a three story shear building, as a simple example, which can be approximated as shown in Figure 1.

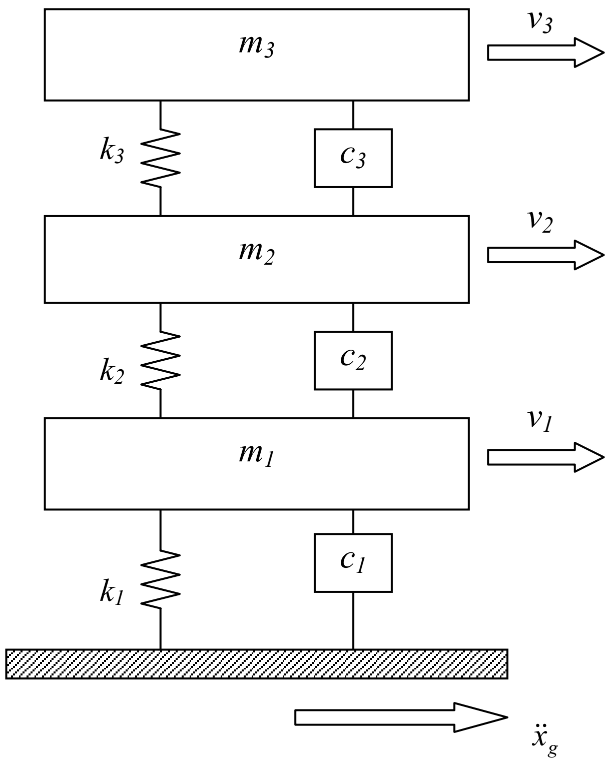


Figure 1 Spring-Mass-Damper model for three story building under seismically excited load

where v_i is displacement of the i^{th} floor relative to the ground, m_i, k_i and c_i are the mass, stiffness and damping coefficient respectively, of the i^{th} floor, and $\ddot{x}_g$ is the ground motion acceleration due to an earthquake. Hence, the absolute displacements of the 1st, 2nd and 3rd floors are $(x_g + v_1)$, $(x_g + v_2)$ and $(x_g + v_3)$ respectively.

The equation of motion for each floor is derived:

For 3rd floor:

$$m_3\left(\ddot{v}_3 + \ddot{x}_g\right) = -c_3\left(\dot{v}_3 - \dot{v}_2\right) - k_3\left(v_3 - v_2\right) \quad \textbf{(1)}$$

For 2nd floor:

$$m_2\left(\ddot{v}_2 + \ddot{x}_g\right) = -c_2\left(\dot{v}_2 - \dot{v}_1\right) - k_2\left(v_2 - v_1\right) + c_3\left(\dot{v}_3 - \dot{v}_2\right) + k_3\left(v_3 - v_2\right) \quad \textbf{(2)}$$

For 1st floor:

$$m_1\left(\ddot{v}_1 + \ddot{x}_g\right) = -c_1\left(\dot{v}_1\right) - k_1\left(v_1\right) + c_2\left(\dot{v}_2 - \dot{v}_1\right) + k_2\left(v_2 - v_1\right) \quad \textbf{(3)}$$

Equations (1) – (3) can be rearranged and combined into matrix form:

$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix}\begin{bmatrix} \ddot{v}_1 \\ \ddot{v}_2 \\ \ddot{v}_3 \end{bmatrix} + \begin{bmatrix} c_1 + c_2 & -c_2 & 0 \\ -c_2 & c_2 + c_3 & -c_3 \\ 0 & -c_3 & c_3 \end{bmatrix}\begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \\ \dot{v}_3 \end{bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 & 0 \\ -k_2 & k_2 + k_3 & -k_3 \\ 0 & -k_3 & k_3 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = -\begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}\left\{\ddot{x}_g\right\}$$

(4)

where m_i, c_i and k_i are the mass, damping and stiffness coefficient for i^{th} story respectively, and v_i is relative displacement of the i^{th} story to the ground. It also can be expressed as the matrix equation:

$$\mathbf{M}\cdot\{\ddot{v}\}+\mathbf{C}\cdot\{\dot{v}\}+\mathbf{K}\cdot\{v\}=-\underline{\mathbf{M}}\cdot\underline{\ddot{x}}_g \quad \textbf{(5)}$$

where **M**, **C** and **K** are the mass, damping and stiffness matrices of the model, respectively, and $\{v\}$, $\{\dot{v}\}$ and $\{\ddot{v}\}$ are the displacement, velocity and acceleration vectors, respectively. This system represents a linear and undamaged baseline model. If damage occurred in the structure from an earthquake, or any other form of damaging excitation, structural properties such as natural frequency and stiffness will change. These changes can be time varying or result without an input from simple modelling error. For the damaged, or mis-modelled, structure, the equations of motion can be re-defined:

$$\mathbf{M}\cdot\{\ddot{\bar{v}}\}+\mathbf{C}\cdot\{\dot{\bar{v}}\}+(\mathbf{K}+\Delta\mathbf{K})\cdot\{\bar{v}\}=-\underline{\mathbf{M}}\cdot\ddot{x}_g \quad \textbf{(6)}$$

where $\ddot{\bar{v}}$, $\dot{\bar{v}}$ and $\bar{v}$ are the responses of the damaged structure, and **ΔK** contains changes in the stiffness of the structure and can be a function of time. By tracking the changes in the stiffness matrix via the **ΔK** term, the structure's condition can be directly monitored without having to identify modal parameters or mode shapes first. Therefore, the development of this heath monitoring algorithm is concentrated on developing a method to directly determine these changes.

Damage is detected by identifying changes in the physical properties of the structure, particularly changes in stiffness, because it is the most likely to change for a steel frame structure as damage occurs. Per the Benchmark problem definition this research examines changes in stiffness properties. Damping changes, **ΔC**, could also be identified and can occur due to hysteresis. However, hysteretic damping could also be seen as oscillations in **ΔK** rather than absolute changes, and identified that way. Change in mass, **ΔM**, is not likely to be significant even in the most damaging cases, hence it is ignored. Finally, the approach can be generalized to more detailed or complex models of the system and variations, as required.

To determine **ΔK** using adaptive LMS, a new form of **ΔK** is defined with time varying scalar parameters, α_i, to be determined using the LMS filter to identify damage. For the three story example of Equations (1) – (4), the **ΔK** matrix is sub-divided into three matrices with entries of 1, -1 and 0 to allow independent identification of changes in k_1, k_2 and k_3, the story stiffnesses, from the α_i coefficients. These matrices have a 1 or -1 wherever that story stiffness appeared in Equation (4).

$$\Delta\mathbf{K} = \alpha_1 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \alpha_3 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_1 + \alpha_2 & -\alpha_2 & 0 \\ -\alpha_2 & \alpha_2 + \alpha_3 & -\alpha_3 \\ 0 & -\alpha_3 & \alpha_3 \end{bmatrix} \quad \textbf{(7)}$$

where

$$\alpha_1 = \Delta k_1, \qquad \alpha_2 = \Delta k_2, \qquad \alpha_3 = \Delta k_3 \quad \textbf{(8)}$$

hence,

$$\mathbf{\Delta K} = \begin{bmatrix} \Delta k_1 + \Delta k_2 & -\Delta k_2 & 0 \\ -\Delta k_2 & \Delta k_2 + \Delta k_3 & -\Delta k_3 \\ 0 & -\Delta k_3 & \Delta k_3 \end{bmatrix} = \sum_{i=1}^{3} \alpha_i \Delta K_i \quad \textbf{(9)}$$

The basic idea of using α_i coefficients and sub-matrices is to enable the identification of changes in story stiffness values, k_i. Therefore, the stiffness of the damaged structure, or effective stiffness changes due to non-linear behaviour such as yielding or hysteresis could be determined by identifying the **ΔK** matrix at every discrete time step. Rewriting Equation (6) using Equations (7) – (9) yields:

$$\mathbf{M}\cdot\{\ddot{\bar{v}}\}+\mathbf{C}\cdot\{\dot{\bar{v}}\}+\mathbf{K}\cdot\{\bar{v}\}+\sum_{i=1}^{n}\alpha_i\Delta K_i v=\mathbf{F} \quad \textbf{(10)}$$

where n is the number of degree-of-freedom of the model and **F** is the known, or estimated, input load vector. Note that n is the maximum number of coefficients to identify to determine changes in each story stiffness. A lesser number can be used if some stories are assumed not to suffer damage. Similarly, a greater number could be used for a more complex structural model with more DOF per story to obtain greater resolution on the changes in structural parameters.

The varying stiffness term is simply the error between the, in this case, linear model and real measurements when actual measured values ($\bar{v}$, $\dot{\bar{v}}$ and $\ddot{\bar{v}}$) are put in for $\{v\}$, $\{\dot{v}\}$ and $\{\ddot{v}\}$ in Equation (10). Hence, $\Delta\mathbf{K}\bar{v}$ is the error in the linear model.

$$\sum_{i=1}^{n}\alpha_i\Delta K_i\bar{v}=\mathbf{F}-\mathbf{M}\ddot{\bar{v}}-\mathbf{C}\dot{\bar{v}}-\mathbf{K}\bar{v} \quad \textbf{(11)}$$

where $\bar{v}$, $\dot{\bar{v}}$ and $\ddot{\bar{v}}$ are measured values of the structural displacement, velocity and acceleration that are obtained either directly and/or from a dynamic state estimator, such as a Kalman filter. Equation (11) is only valid at any point in time if the α_i have the correct values. At any discrete time, k, the difference between the linear model and actual measurements can be defined:

$$y_k = \mathbf{F}_k - \mathbf{M}\ddot{\bar{v}}_k - \mathbf{C}\dot{\bar{v}}_k - \mathbf{K}\bar{v}_k \tag{12}$$

where $\mathbf{F}_k$ is the input at time k, and $\bar{v}_k$, $\dot{\bar{v}}_k$ and $\ddot{\bar{v}}_k$ are the measured displacement, velocity and acceleration at time k. The individual elements of the vector signal y_k can be readily modelled in real-time using individual adaptive LMS filters so that the coefficients α_i can be readily determined from the reduced noise modelled signal.

$$y_k = \sum_{i=1}^{n} \alpha_i \Delta K_i \bar{v}_k \tag{13}$$

More specifically, if each element of the vector signal vector, y_k is modelled using an adaptive filter then the α_i can be determined directly using the linear system of equations defined in Equation (13) at each time step.

2.2. Definition and Classification of Damage

Damage is defined as changes introduced into a system, either intentional or unintentional, which adversely affect the current or future performance of that system. The effects of damage on a structure can be classified as linear or nonlinear. A linear damage situation is defined as the case when the initially linear-elastic structure remains linear-elastic after damage. The changes in modal properties are a result of changes in the geometry and/or the material properties of the structure, but the structural response can still be modelled using linear equations of motion. Linear methods can be further classified as model-based and non-model-based. Model-based methods assume that the monitored structure responds in some predetermined manner that can be accurately discretised by finite element analysis, such as the response described by Euler-Bernoulli beam theory.

Nonlinear damage is defined as the case when the initially linear-elastic structure behaves in a nonlinear manner after the damage has been introduced. One example of nonlinear damage is the formation of a fatigue crack that subsequently opens and closes under the normal operating vibration environment. Other examples include loose connections that rattle and nonlinear material behaviour such as that exhibited by polymers. The majority of the studies reported in the technical literature address only the problem of linear damage detection. However, damage in civil structure under large environmental loads is inherently permanent and non-linear. Another classification system for damage-identification methods defines four levels of damage identification, as follows (Rytter, 1993):

- Level 1: Determination that damage is present in the structure
- Level 2: Level 1 plus determination of the geometric location of the damage
- Level 3: Level 2 plus quantification of the severity of the damage
- Level 4: Level 3 plus prediction of the remaining service life of the structure

Current vibration-based damage identification methods that do not make use of some structural model primarily provide Level 1 and Level 2 damage identification. When vibration-based methods are coupled with a structural model, Level 3 damage identification can be obtained in some cases. Level 4 prediction is generally associated with the fields of fracture mechanics, fatigue-life analysis, or structural design assessment.

In this thesis, we are considering Level 3 damage identification for non-linear damage. Once the change in stiffness matrix is identified, we are able to detect a presence, the geometric location and the severity of the damage by examining the elements of the resultant stiffness matrix. Note that once damage is identified, structural design codes provide metrics and methods to determine the remaining service life. Hence, Level 4 identification is possible for civil structure using the results from the method presented.

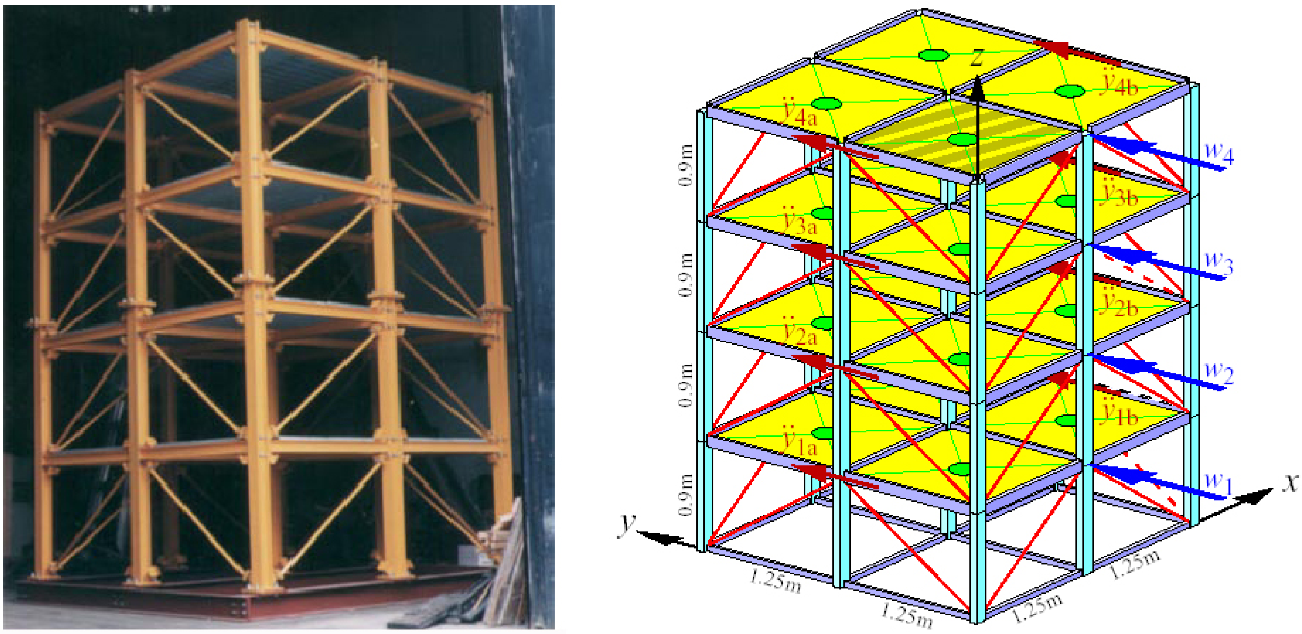
3. SHM Task Group Benchmark Problem

The IASC-ASCE Task Group on SHM was established in 1999 and the group developed a series of benchmark SHM problems (Johnson et al, 2000). The Task Group decided that the use of simulated data from an analytical structural model based on an existing structure would allow for future comparisons with data taken on the actual structure. Starting with simulated data allows participants to better understand the sensitivities of their methods to various aspects of the problem, such as differences between the identification model and the true model, incomplete sensor information, and the presence of noise in measurement signals.

The structure (Black and Ventura, 1998), shown in Figure 2 (a) and (b), is a 4-story, 2-bay by 2-bay steel-frame scale-model structure in the Earthquake Engineering Research Laboratory at the University of British Columbia (UBC). It has a 2.5 m × 2.5 m plan and is 3.6 m tall. The members are hot rolled grade 300W steel (nominal yield stress 300 MPa). The sections are unusual, designed for a scale model, with properties as given in Table 1. There is one floor slab per bay per floor: four 800 kg slabs at the first level, four 600 kg slabs at each of the second and third levels, and, on the fourth floor, either four 400 kg slabs or three 400 kg slabs and one 550 kg slab to create some asymmetry.

Two finite element models based on this structure were developed to generate the simulated data. The first is a 12 DOF shear-building model that constrains all motion except two horizontal translations and one rotation per floor. The second is a 120 DOF model that only requires floor nodes to have the same horizontal translation and in-plane rotation. The columns and floor beams are modelled as Euler-Bernoulli beams in both finite element models. The braces are bars with no bending stiffness. A diagram of the analytical model is shown in Figure 2 (b).

The SHM Task Group classified the analytical model into six different cases according to the number of DOF, whether it is symmetric or asymmetric and the type of loads applied. Cases 1, 3 and 4 are defined using the 12 DOF model whereas cases 2, 5 and 6 use the 120 DOF model. In cases 1 and 2, loads are applied on all floors, while the load is applied only on the top floor in cases 3, 4, 5 and 6. The analytical model was asymmetric in cases 4, 5 and 6. Table 2 shows all cases with specifications defined by the SHM Task Group.



(a) (b)

Figure 2 Benchmark Structure at the University of British Columbia (a) Steel-frame scale structure and (b) Diagram of analytical model (the w_i are excitations and $\ddot{y}_i$ are accelerometer measurements in y-direction

Table 1 Mechanical Properties of structural members

Property		Columns	Floor Beams	Braces
section type		B100×9	S75×11	L25×25×3
cross-sectional area	A [m^2]	1.133×10^{-3}	1.43×10^{-3}	0.141×10^{-3}
moment of inertia (strong direction)	I_y [m^4]	1.97×10^{-6}	1.22×10^{-6}	0
moment of inertia (weak direction)	I_z [m^4]	0.664×10^{-6}	0.249×10^{-6}	0
St. Venant torsion constant	J [m^4]	8.01×10^{-9}	38.2×10^{-9}	0
Young's Modulus	E [Pa]	2×10^{11}	2×10^{11}	2×10^{11}
Mass per unit length	ρ [kg/m]	8.89	11.0	1.11

The finite element models, by removing the stiffness of various elements, can simulate damage to the structure. Five damage patterns are defined for the structure in Table 3. Damage pattern 1 is where all of the first floor braces are removed and pattern 2 is where all of the first and third floors braces removed. Damage pattern 3 is defined as one brace removed in the first floor (drawn as dashed line in Figure 2 (b)) and pattern 4 is when one brace is removed in each of the first and third floors (drawn as dashed lines in Figure 2 (b)). Finally, damage pattern 5 is described as damage pattern 4 but with the floor beam partially unscrewed from the column (drawn as dashed line along the floor beam in Figure 2 (b)). As a result, the unscrewed beam-column connection can only transmit forces and cannot sustain any bending moments.

Table 2 Model cases of SHM Task Group Benchmark problem

Case	**Model**	**Applied Loads**
Case 1	12 DOF and symmetric	Ambient loads on all floors
Case 2	120 DOF and symmetric	Ambient loads on all floors
Case 3	12 DOF and symmetric	load at the roof
Case 4	12 DOF and asymmetric	load at the roof
Case 5	120 DOF and asymmetric	load at the roof
Case 6	120 DOF and asymmetric	load at the roof

Table 3 Damage patterns of SHM Task Group Benchmark problem

Damage Pattern	Descriptions
Damage Pattern 1	All braces in 1st story are removed.
Damage Pattern 2	All braces in 1st and 3rd stories are removed.
Damage Pattern 3	One brace in 1st story is removed.
Damage Pattern 4	One brace in each of the 1st and 3rd stories are removed.
Damage Pattern 5	Damage Pattern 4 and loosen floor beam at 1st level.

4. ADAPTIVE FILTERING

Adaptive filters are digital filters with coefficients that can change over time. The general idea is to assess how well the existing coefficients are performing in modelling a noisy signal, and then adapt the coefficient values to improve performance. Because of their self-adjusting performance and built-in flexibility, adaptive filters have found use modelling signals in many real-time applications, particularly in advanced telecommunications such as cell phones. The Least Mean Squares (LMS) algorithm is one of the most widely used of all the adaptive filtering algorithms and is relatively simple to implement. It is an approximation of the Steepest Descent Method using an estimator of the gradient instead of its actual value, considerably simplifying the calculations and can be readily performed in real-time applications. The initial goal in this case is to model the individual, scalar elements of the signal y_k in Equation (13) using the adaptive LMS filter.

In adaptive LMS filtering, the coefficients are adjusted from sample-to-sample to minimize the Mean Square Error (MSE), between a measured noisy scalar signal and its modelled value from the filter. The scalar error at time k is defined:

$$e_k = \hat{y}_k - W_k^T X_k = \hat{y}_k - \sum_{i=o}^{m-1} w_k(i) x_{k-i} = \hat{y}_k - \hat{n}_k \qquad \textbf{(14)}$$

where W_k is the adjustable filter coefficient vector or weight vector at time k, $\hat{y}_k$ is the noisy measured signal to be modelled or approximated, X_k is vector the input to the filter model of current and previous filter outputs, x_{k-i}, so $W_k^T X_k$ is the vector dot product output from the filter to model a scalar signal $\hat{y}_k$, and m is the number of taps or prior time steps considered. The Widrow-Hopf LMS algorithm for updating the weights to minimise the error, e_k is defined (Ifeachor and Jervis, 1993):

$$W_{k+1} = W_k + 2\mu \cdot e_k \cdot X_k \tag{15}$$

where μ is a positive scalar that controls the stability and rate of convergence.

Therefore, to find the α_i coefficients in Equation (13) using adaptive LMS the filter output modelling the m elements of the vector y_k, a linear system must be solved.

$$[\hat{y}_k]^N = [W_k^T X_k]^N = \sum_{i=o}^{m-1} (\alpha_{1i}\Delta K_1 + \alpha_{2i}\Delta K_2 + \cdots + \alpha_{ni}\Delta K_n)\overline{v}_{k-i} \tag{16}$$

where $\overline{v}_{k-i}$ is the measured displacement response i steps prior and N is the element, $N = 1$, …, n, of the vector $\hat{y}_k$ being modelled. Hence, these α_i, and thus the changes in stiffness can be found directly using adaptive LMS at any point in time, as data is gathered. As more data is obtained the error, e_k, goes to zero and α_i approach the correct damage identified values.

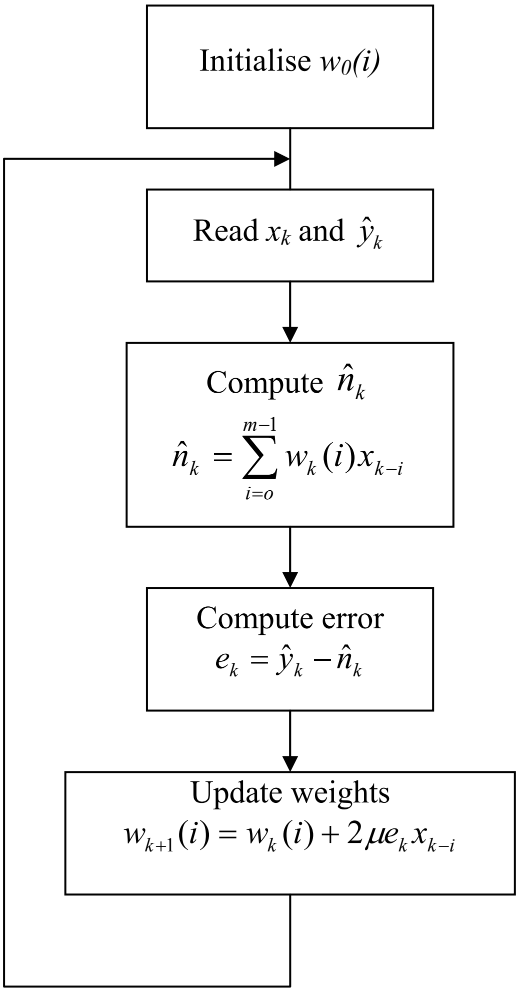


Figure 3 Flowchart for the standard adaptive LMS filter

The general computational procedure for the basic adaptive LMS algorithm is summarized in Figure 3. (Ifeachor and Jervis, 1993). By determining changes in α_i values using adaptive LMS filtering, the changes in the stiffness matrix due to the damage in the structure can be obtained. The first step of the method is the initialisation of weights, w_0, usually to zero. In the next step, x_k and $\hat{y}_k$ are measured, where x_k is the displacement response and $\hat{y}_k$ is the noisy signal from the structure. Third, the filter output $\hat{n}_k$ is calculated at time k. The computation of the filter output depends on a number of input signals and therefore the number of filters used (n), and number of taps, or prior time steps, (m) used for each weight in the calculation of each step filter. For instance, the three-story building model with one degree-of-freedom (DOF) per floor has 3 elements in y_k, so the number of input signals is equivalent to the total DOF of the system. The number of taps affects the rate of convergence. More taps makes the algorithm converge faster and improves the accuracy, however it takes more computation. Before the weight is updated for the next time step, the error estimate is computed by Equation (14), subtracting the filter output, $\hat{n}_k$ from the noisy measured signal, $\hat{y}_k$. The weights are then updated and the procedure continues. Two different methods for updating the weights were developed, one is called the One Step method and the other is the Two Step method. The following section has the derivation of these two methods.

5. Adaptive LMS Based Approaches for SHM

The One Step method presented in this section updates all of the weights in one operation as a whole matrix. Using the One Step method, the weights are the α_i values, and are determined directly via a modified adaptive LMS filter. Therefore, the *n* LMS filters are coupled. As a result, convergence in modelling the signal y_k can be slower. The Two Step method in which each element of the vector y_k is modelled by its own adaptive LMS filter can also be used. However, in this case, the α_i are solved after each time step independently of the filter weight updating process in a "second" step. Because the filters are decoupled convergence can be faster. Derivations of adaptive LMS based approaches for identifying the change in stiffness matrix: the One and the Two Step methods are presented.

5.1. *Two Step Method*

A noisy signal vector, y_k, for the linear model error is obtained from a simulation of the non-linear model per Equation (12). The vector y_k can be modelled using *n* adaptive LMS filters.

$$y_k = \begin{bmatrix} (\hat{y}_k)^1 \\ \vdots \\ (\hat{y}_k)^n \end{bmatrix} = \begin{bmatrix} \left(W_k^T X_k\right)^1 \\ \vdots \\ \left(W_k^T X_k\right)^n \end{bmatrix} \tag{17}$$

where each W_k^T is updated individually for n different input signals and $\left(W_k^T X_k\right)^i$ is the output for the i^{th} individual adaptive LMS filter. In the Two Step method, adaptive LMS filters approximate the noisy signal, $\hat{y}_k \approx y_k$ for each step, where $\hat{y}_k$ is the estimate of y_k with dimension $n \times 1$. Hence, from Equations (16) and (17), the filter approximation, $\hat{y}_k$ is defined:

$$\hat{y}_k = \sum_{i=1}^{n} \alpha_i \Delta K_i \bar{v}_k = \left[\Delta K_1 \bar{v}_k \quad \cdots \quad \Delta K_n \bar{v}_k\right] \alpha_k = A \alpha_k \qquad \textbf{(18)}$$

where dimensions of matrix A are $n \times n$ and α_k is a $n \times 1$ vector of coefficients α_i at time k. Therefore, the α_i values can be determined analytically by solving Equation (18) as long as the matrix is full rank.

The Two Step method is a fast and simple approach, and is also robust to noise because of its direct use of LMS filters. However, the computation required is more intense than desired due to the matrix solutions required at each time step, particularly as the number of DOF rises. Hence, as the complexity of the model increases the computational time also increases significantly. What is required is a method that combines the robustness and adaptive characteristics of LMS filtering to directly determine the α_i coefficients without the matrix solution.

5.2. *One Step Method*

The One Step method is developed to simplify and combine the steps of noisy signal modelling and filter approximations. The linear model error, estimated between the measured noisy signal and its modelled value from the filter, defined in Equation (14) can be expressed:

$$e_k = y_k - \sum_{j=0}^{m-1} \sum_{i=1}^{n} \alpha_{ij} \Delta K_i \bar{v}_k = y_k - Q_k \quad \textbf{(19)}$$

$$w_k = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,m} \\ \alpha_{2,1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \alpha_{n,1} & \cdots & \cdots & \alpha_{n,m} \end{bmatrix}_k \quad \textbf{(20)}$$

where $\bar{v}_k$ and y_k are noisy signals, Q_k is a $n \times 1$ vector and α_{ij} is the ith row and jth column element of the weight matrix of dimension $n \times m$, Equation (20). Hence, the change in k_i will be the sum over j of α_{ij}. This averaged approach essentially low-pass filters the signal $\bar{v}_k$ and reduces the impact of noise. An exact unfiltered solution would simply use $m = 1$. Note that there are no prior time steps involved when estimating the error at time k, because y_k is not stationary and adaptive LMS based algorithms are not effective in the presence of non-stationary signals (Ifeachor and Jervis, 1993). In addition, the error, e_k, in Equation (19) is the error at this time step and is a function of the response at time k only.

Hence, the mean square error (MSE) can be determined:

$$e_k^T e_k = (y_k - Q_k)^T (y_k - Q_k)$$

$$= y_k^T y_k + Q_k^T Q_k - y_k^T Q_k - Q_k^T y_k \qquad \textbf{(21)}$$

$$= y_k^T y_k + Q_k^T Q_k - 2 y_k^T Q_k$$

Since, adaptive LMS minimises the MSE with respect to the weights α_{ij}, the optimum solution occurs when the gradient of MSE is zero. The gradient of MSE is defined:

$$\nabla MSE_{\alpha_{ij}} = \nabla_{\alpha_{ij}} \left(y_k^T y_k + Q_k^T Q_k - 2 y_k^T Q_k \right)$$

$$= 0 + 2 Q_k^T \frac{\partial Q_k}{\partial \alpha_{ij}} - 2 y_k^T \frac{\partial Q_k}{\partial \alpha_{ij}}$$

$$= 2 \left(Q_k^T - y_k^T \right) \frac{\partial Q_k}{\partial \alpha_{ij}} \qquad \textbf{(22)}$$

$$= -2 e_k^T \frac{\partial Q_k}{\partial \alpha_{ij}}$$

where $\nabla MSE_{\alpha_{ij}}$ is one element of an $n \times m$ matrix ∇MSE. The term $\frac{\partial Q_k}{\partial \alpha_{ij}}$ is an $n \times 1$ vector defined:

$$\frac{\partial Q_k}{\partial \alpha_{ij}} = \frac{\partial}{\partial \alpha_{ij}} \left(\sum_{j=0}^{m-1} \sum_{i=1}^{n} \alpha_{ij} \Delta K_i \bar{v}_k \right) = \Delta K_i \bar{v}_k \qquad \textbf{(23)}$$

Therefore,

$$\nabla MSE = -2 \left[e_k^T \Delta K_i \bar{v}_k \right] \qquad \textbf{(24)}$$

where ∇MSE is an $n \times m$ matrix and $\left[e_k^T \Delta K_i \bar{v}_k\right]$ is the same across an entire row for all i = 1, …, n rows. The weight matrix of dimension $n \times m$ can then be updated.

$$\begin{aligned} w_{k+1} &= w_k - \mu \nabla MSE \\ &= w_k + 2\mu\left[e_k^T \Delta K_i \bar{v}_k\right] \end{aligned} \tag{25}$$

where the term $\left[e_k^T \Delta K_i \bar{v}_k\right]$ is the same for all m elements in the i^{th} row, continuing low-pass filtering from the definition in Equation (19).

Variations of this algorithm for updating the weight matrix include using prior time steps in error estimation to get a gradient over time, or determining the gradient without the coupling in the ΔK_i matrices and examining only the diagonal elements.

Using prior time step data to obtain a potentially improved gradient results in the following updated formula:

$$w_{k+1} = w_k + 2\mu\left[e_k^T \Delta K_i \bar{v}_{k-j}\right] \tag{26}$$

where the term $\left[e_k^T \Delta K_i \bar{v}_{k-j}\right]$ is the $(i,j)^{th}$ element of the $n \times m$ gradient matrix. Equation (26) is the term that would result from the derivation presented if prior time steps were used to calculate the error, e_k, in Equation (19).

Decoupling the gradient estimation by approximating ΔK_i as a zero matrix with a 1.0 for the (i,i) element results in the following weight update formula:

$$w_{k+1} = w_k + 2\mu\left[e_k^T(i)\overline{v}_k(i)\right] \tag{27}$$

where the term $\left[e_k^T(i)\overline{v}_k(i)\right]$ is the same for all m elements in row i. Without these coupling terms the gradient is calculated based only on changes in diagonal elements of the stiffness matrix, decreasing the likelihood of coupling terms reducing the gradient near zero error values. Note that the error calculation in Equation (19) still uses the ΔK_i as originally defined with coupling, and Equation (27) only modifies the means by which the weights are updated. Note that Equatoin (27) has significantly fewer computations than Equation (25). The following section presents testing of the adaptive LMS filtering algorithm on the Benchmark Problems for both the Two Step and One Step methods.

Throughout a series of tests, the weights converge slightly faster with the Two Step method. In terms of computational time, with small numbers of n and m (< 3), both the Two and One Step methods take almost the same amount of time to converge. However, for a more complex model with higher numbers of DOF and taps, the One Step method will require much less computational time based on counting the number of fundamental computational steps required.

6. ADAPTIVE LMS FOR SHM BENCHMARK PROBLEM

6.1. Cases and Damage Patterns Considered

The SHM Task Group analysed the structure using two models, one of 12 DOF and a second of 120 DOF. In the 12 DOF model, the structure is assumed to act as a shear building with three DOF per floor: translation in the x- and y-direction and rotation. Without constraints on the horizontal translation and rotation of floor nodes, the more complex 120 DOF model is formed. In this paper, only the 12 DOF model and the simpler, one direction, 4 DOF model are considered. However, all the methods presented are readily generalized to this more complex case. Per the algorithms presented, the displacement, velocity and acceleration at each DOF are assumed to be measured or estimated and include noise as defined in the Benchmark problem. The input loads in the Benchmark problems studied are assumed known in their definition.

Table 4 shows the cases and damage patterns of the Benchmark Problem considered in testing of the adaptive LMS based SHM methods presented. The four damage patterns are applied to each case in the table. Cases 2, 5 and 6 use the 120 DOF model and are not considered here. The Task Group also provided the MATLAB® based data generation codes for the 12 DOF and 120 DOF models (IASC-ASCE SHM Task group, 1999). The code allows the user to choose cases and damage patterns in which they are interested, the user also can define a damage pattern other than the 6 damage patterns defined by the Task

Group. Resultant mass and stiffness matrices of the damaged structure, force applied and acceleration measured are stored as the outputs of the code.

Table 4 Cases and Damage patterns considered in simulation

Case	Damage Pattern	Model Used	Descriptions
1	1	4 DOF	Symmetric, loads on all floors All braces in 1st story removed
	2	4 DOF	Symmetric, loads on all floors All braces in 1st & 3rd stories removed
	3	12 DOF	Symmetric, loads on all floors One brace in 1st story removed
	4	12 DOF	Symmetric, loads on all floors One brace in each of 1st & 3rd stories removed
3	1	4 DOF	Symmetric, load at the roof All braces in 1st story removed
	2	4 DOF	Symmetric, load at the roof All braces in 1st & 3rd stories removed
	3	12 DOF	Symmetric, load at the roof One brace in 1st story removed
	4	12 DOF	Symmetric, load at the roof One brace in each of 1st & 3rd stories removed
4	1	12 DOF	Asymmetric, load at the roof All braces in 1st story removed
	2	12 DOF	Asymmetric, load at the roof All braces in 1st & 3rd stories removed
	3	12 DOF	Asymmetric, load at the roof One brace in 1st story removed
	4	12 DOF	Asymmetric, load at the roof One brace in each of 1st & 3rd stories removed

6.2. Simulation Parameters

The following set of parameters are used in all simulations, unless otherwise stated:

- Input load(s) = $1 \times 10^6 \sim 10^7 \sin(30t)\ N$
- Sample rate = *100 Hz*
- $\mu = 0.3$
- Number of taps, $m = 5$

The convergence rate of the weights in the algorithm depends on the LMS parameter μ and the number of taps used. Even though faster convergence for each different case of the Benchmark problem can be achieved by varying those parameters, they would typically be fixed in a practical application. The values used here were developed by trial and error to illustrate the methods developed and may not be completely optimal.

The One Step method, using Equation (27) for uncoupled weight updating, was used throughout the tests of adaptive LMS on the Benchmark problem. This form is used because it is computationally the simplest and, as will be shown, the most effective. Trade-offs between other variations of the weight updating method presented in Equations (25) and (26) are also investigated. For cases 1 & 3 and damage patterns 1 & 2, the 12 DOF model can be approximated as a 4 DOF model (one DOF per floor), because it is symmetric and deforms only in the loading direction (y-direction). In the 4 DOF model, there are four α_i coefficients and four sub-matrices for ΔK_i, and each α_i represents the change in stiffness of each floor similar to the 3 DOF definition in Equations (7) and (8).

6.3. *Damage Profiles*

The components can be broken or deformed in various ways. It could be broken instantaneously or start to fail, then after a certain time, break completely. Obviously, in the real-time damage identification, the results of simulation, such as convergence time or rate would depend on how the damage occurred in the structure.

Hence, two different kinds of damage profile were introduced in the simulation for identifying the damage patterns using the adaptive LMS based method of the Benchmark problem. The first damage profile introduced was a sudden failure (step change) of braces and the second one was a gradual failure (ramp change). In the simulations, the damage is introduced at 5 seconds after a start of the simulation, which lasts for total simulation time of 20 seconds. The following section contains the results from the simulation of various Benchmark cases. Each case and damage pattern was simulated with two damage profiles introduced.

7. Benchmark Problem Simulation Results

As a result of the simulation run, the actual changes in stiffness of each story along with the weights approximated by adaptive LMS based method are plotted in each Benchmark case. In the result figure, the solid line(s) represent the actual stiffness change during the simulation and the dashed line(s) indicate the value, from the α_i, from applying this adaptive LMS based method. For each case and damage pattern, the test was carried out for two different damage situations, a sudden failure in the structure at 5 seconds, and damage that starts at 5 seconds and then is fully damaged gradually. All Benchmark cases were tested with the three different adaptive LMS approaches developed in the earlier section. The first method used was the One Step method without coupling terms in the gradient estimation as in Equation (27), and the second is the One Step method with coupling, which uses Equation (25). Last was the Two Step method.

7.1. *One Step Method without Coupling*

All braces at the first floor are removed in damage pattern 1, hence, only α_1 changes when the 4 DOF model used. The other values correctly remain at zero. Figure 4 and Figure 5 clearly show that only α_1 is changed and other three α's remained unchanged. The estimated α_1 using LMS reached 90% of the actual change within 1 second after the damage occurred. Interestingly, the slower rate of change takes relatively longer to converge in Figure 5. This result is likely due to the difficulty LMS-based methods can have in tracking non-stationary changes. The change in Figure 4 is sudden but then no further charge occurs, minimizing the period over which changes take place.

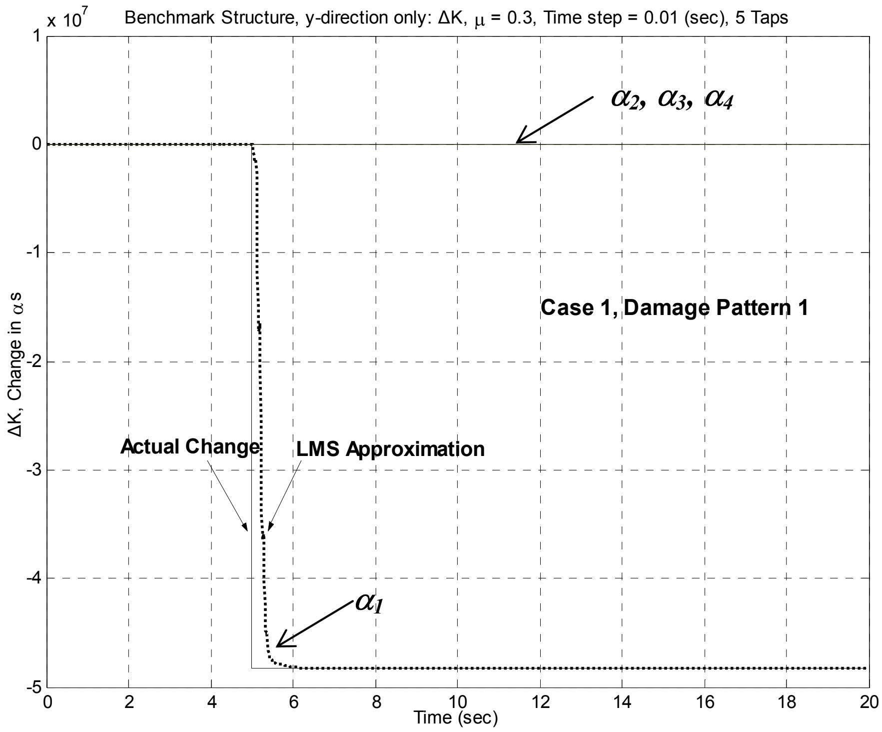


Figure 4 Identifying changes of α by adaptive LMS of 4 DOF model for case 1 with damage pattern 1 using the One Step Method without coupling for sudden failure

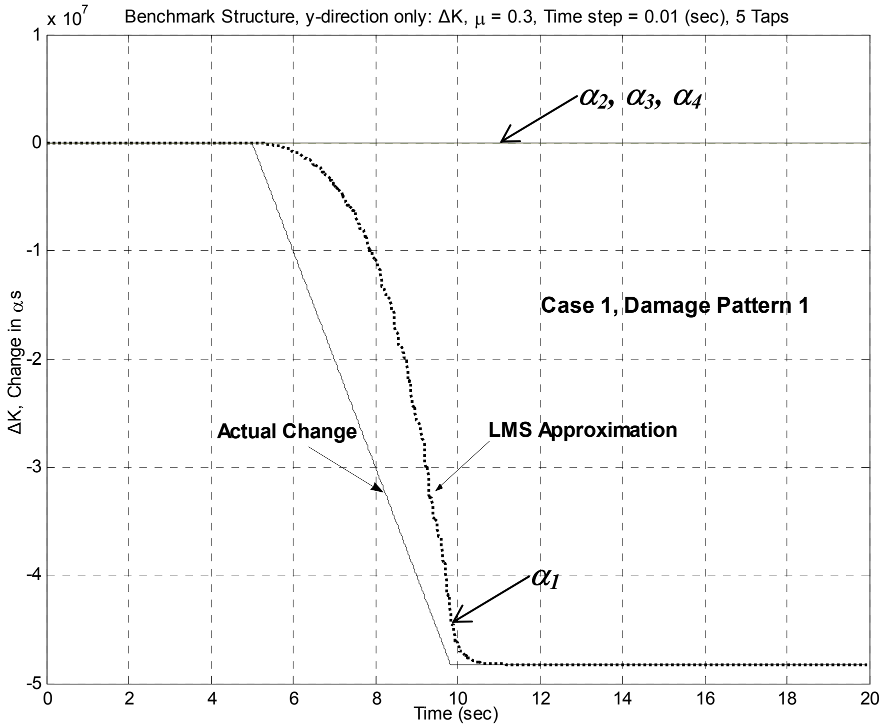


Figure 5 Identifying changes of α by adaptive LMS of 4 DOF model for case 1 and damage pattern 1 using the One Step Method without coupling due to gradual failure

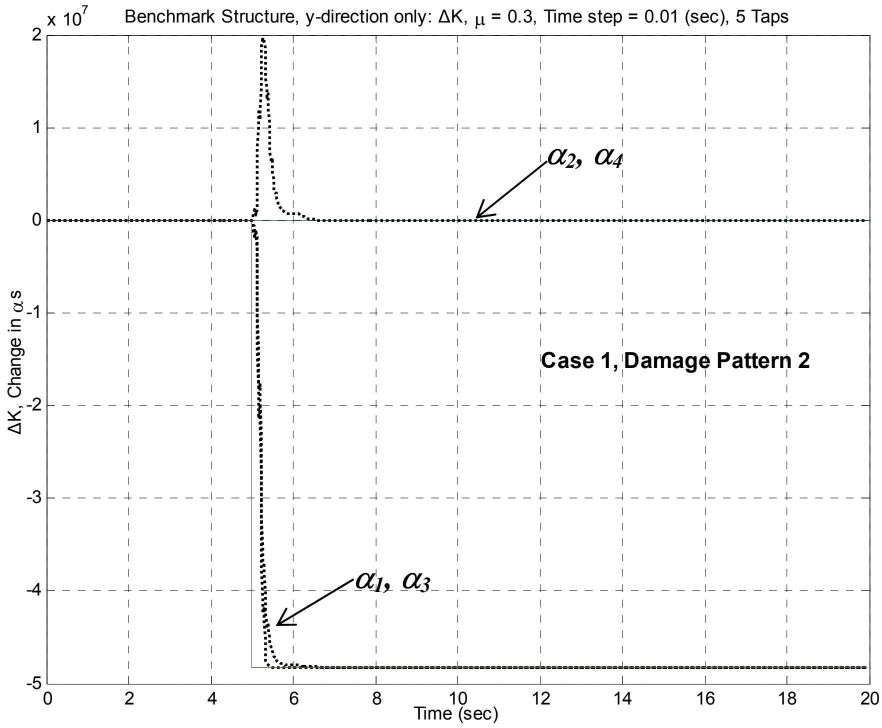


Figure 6 Case 1 and damage pattern 2 using the One Step Method without coupling due to sudden failure

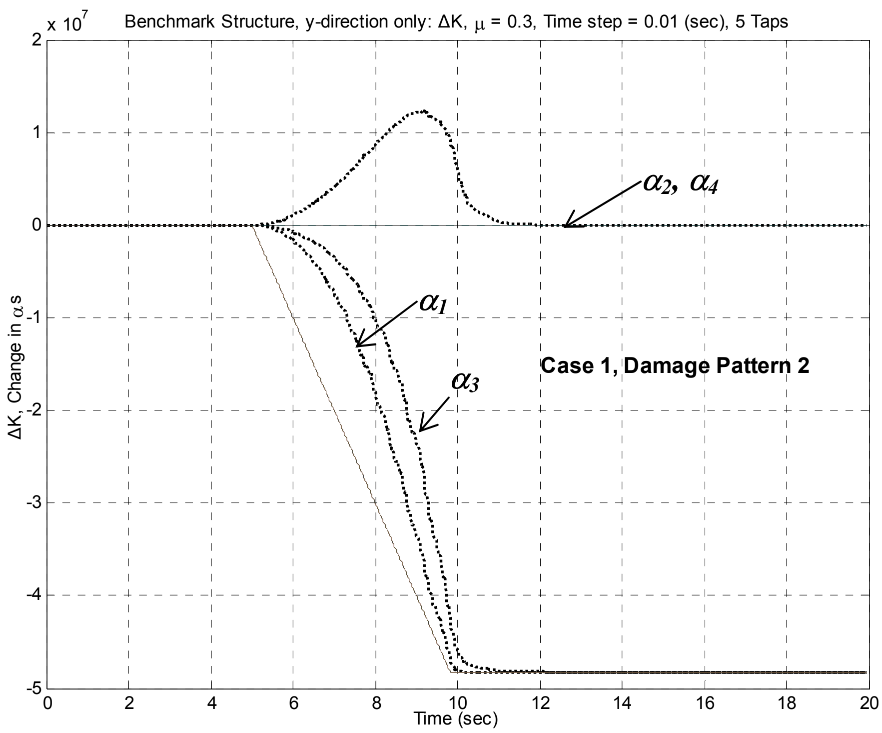


Figure 7 Case 1 and damage pattern 2 using the One Step Method without coupling due to gradual failure

Figure 6 and Figure 7 show the results for case 1 and damage pattern 2, the 4 DOF model was used as the stiffness in the 1^{st} and 3^{rd} floors is changed due to removing all of the braces in these floors. Such changes result in changes in two α_i values for the 4 DOF model (α_1 and α_3). The longest convergence times for α_1 and α_3 in both Figure 6 and Figure 7 are less than 1 second to reach 95% of the actual changes. Note that while the terms converge, some α_i that end up being zero are non-zero for a brief time.

Damage patterns 3 and 4 have partial damage in the 1^{st} floor and the 1^{st} and 3^{rd} floors, respectively, hence, damage pattern 3 and 4 were simulated using the 12 DOF model, as shown in Figure 8 – Figure 11. For case 1 and damage pattern 3 in Figure 8 and Figure 9, it required changes in two α coefficients, specifically α_2 and α_3. The weight α_2 is the change in stiffness of the 1st floor in y-direction and α_3 is the change in rotation. Figure 8 and Figure 9 show that the convergence time for α_2 is less than 1 second to reach 95 % of the actual change, however for α_3, it took longer to converge. The rest of the coefficients (change in stiffness of each DOF) are not changing in damage pattern 3, hence they remain zero.

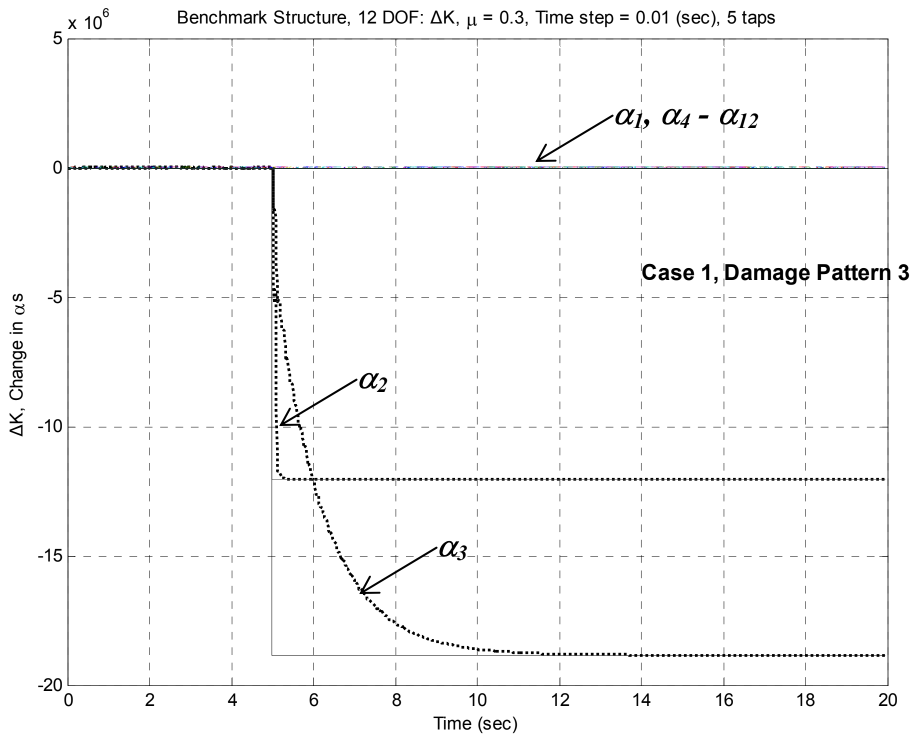


Figure 8 Case 1 and damage pattern 3 using the One Step Method without coupling due to sudden failure

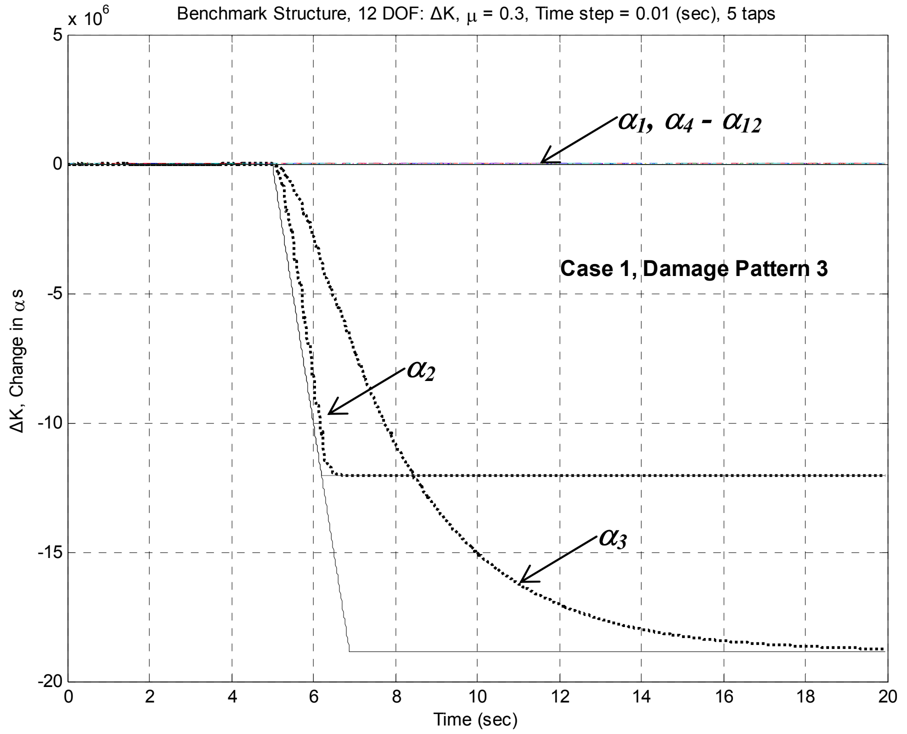


Figure 9 Case 1 and damage pattern 3 using the One Step Method without coupling due to gradual failure

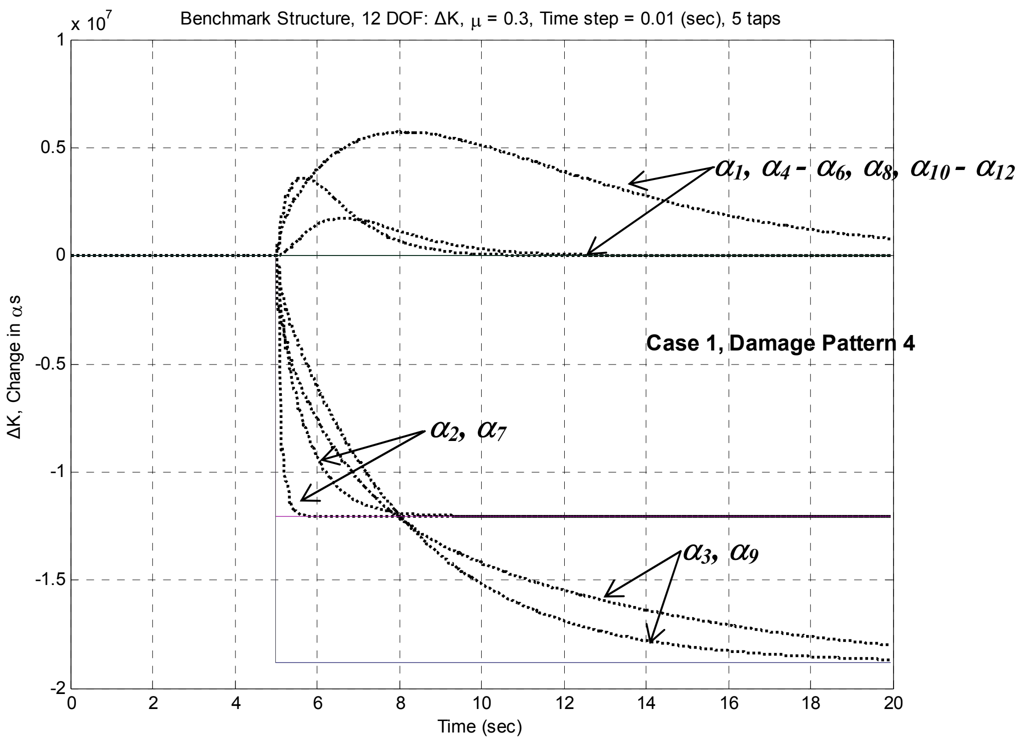


Figure 10 Case 1 and damage pattern 4 using the One Step Method without coupling due to sudden failure

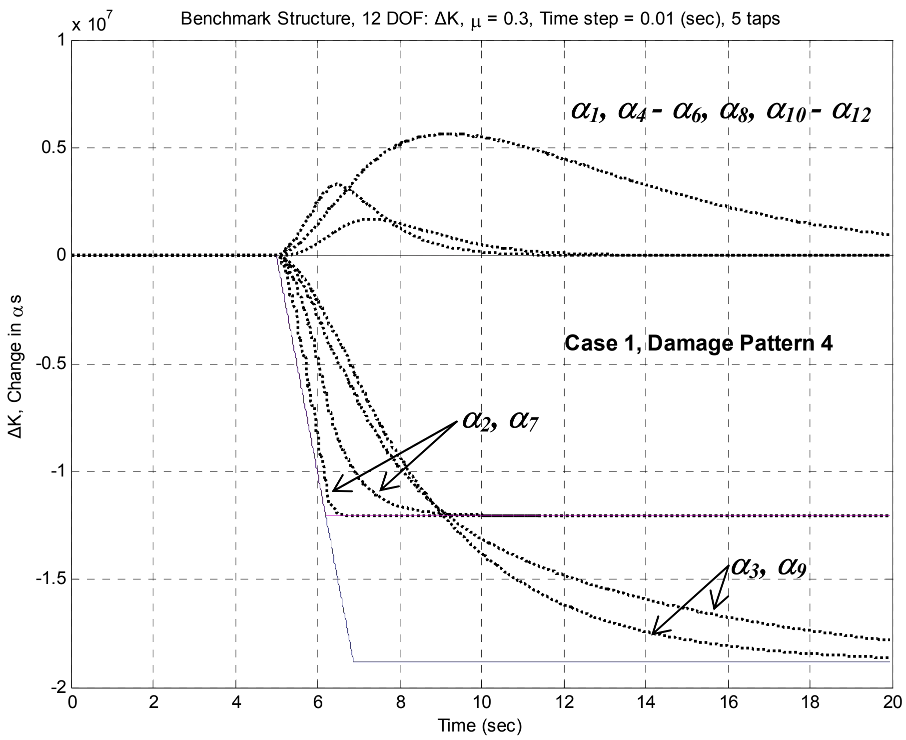


Figure 11 Case 1 and damage pattern 4 using the One Step Method without coupling due to gradual failure

Damage pattern 4 causes changes in four α coefficients; changes of α_2 and α_3 denote that there is partial damage in the 1st floor and changes of α_7 and α_9 describe the partial damage in the 3rd floor. Figure 10 and Figure 11 show the adaptive LMS based identification of damage in the Benchmark structure for case 1 and damage pattern 4. As for damage pattern 3 in Figure 8 and Figure 9, the stiffness changes in y-direction, α_2 and α_7, are well tracked by the method and converged within 1 second to 90% of the actual changes. Again, the convergence time, for identifying the stiffness changes in rotational direction, α_3 and α_9, take about 10 seconds to reach 90% of the actual changes. For the rest of α's, there are initial fluctuations right after the damage started to occur, however all those values eventually return to zero.

Results for case 3 with all damage patterns 1 – 4 in Figure 12 – Figure 19 are very similar to the result of case 1. In case 3, the load is applied only in the top floor, 4^{th} floor as described in the Benchmark problem. The 4 DOF model was used to simulate damage patterns 1 and 2, and the 12 DOF model was used for damage patterns 3 and 4. From the comparison of the results for case 1 and case 3, when the 4 DOF model is used for damage patterns 1 and 2, the results of both cases are almost identical. In contrast, when the 12 DOF model was used, the convergence in case 3 for damage patterns 3 and 4 are found to be faster than in case 1 with the same damage patterns, especially for the changes in stiffness in rotational direction.

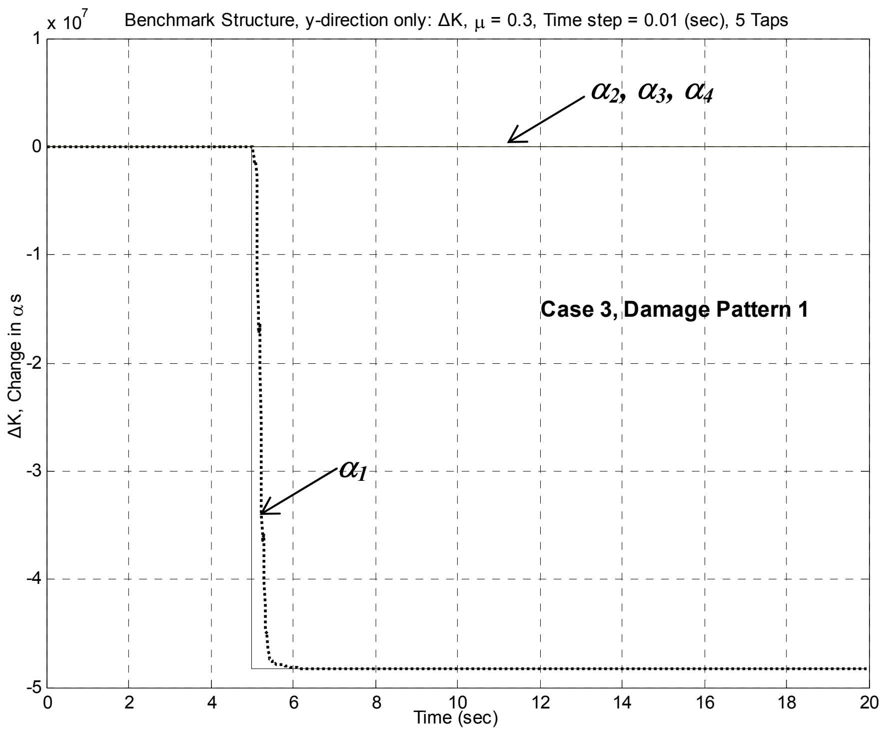


Figure 12 Case 3 and damage pattern 1 using the One Step Method without coupling due to sudden failure

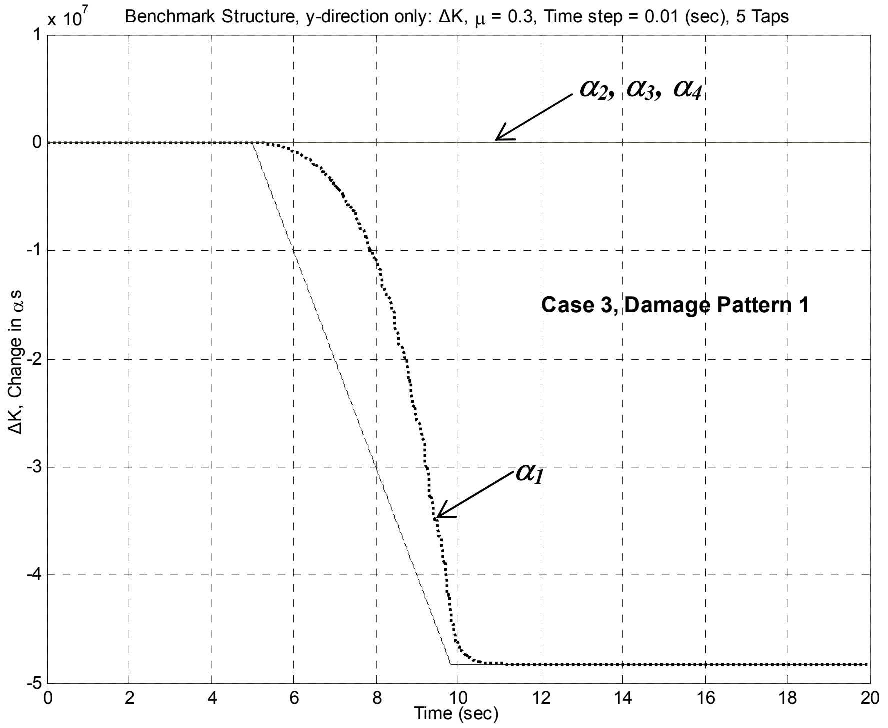


Figure 13 Case 3 and damage pattern 1 using the One Step Method without coupling due to gradual failure

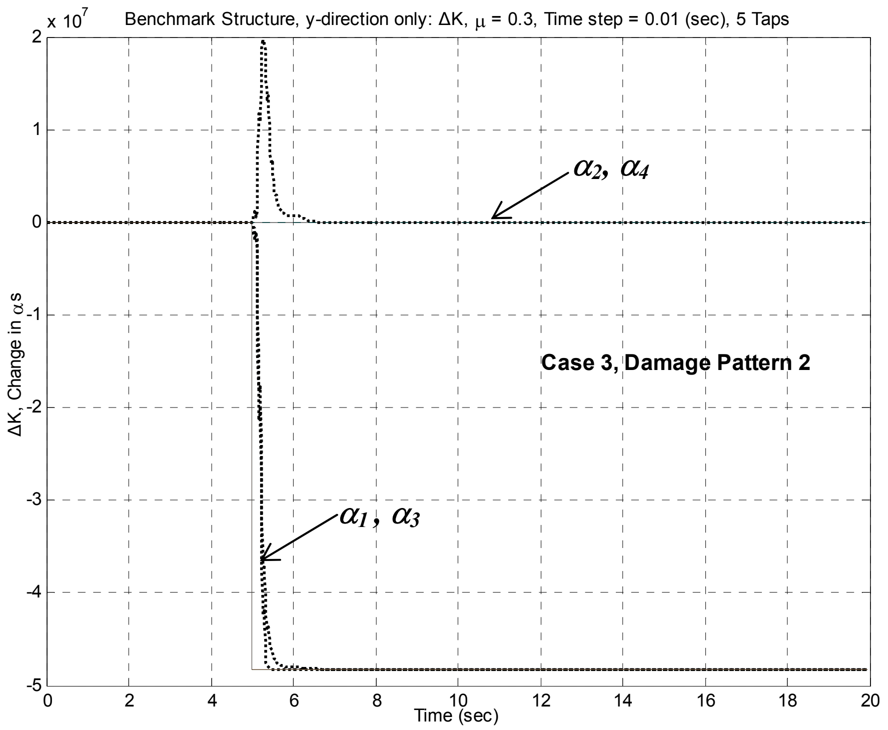


Figure 14 Case 3 and damage pattern 2 using the One Step Method without coupling due to sudden failure

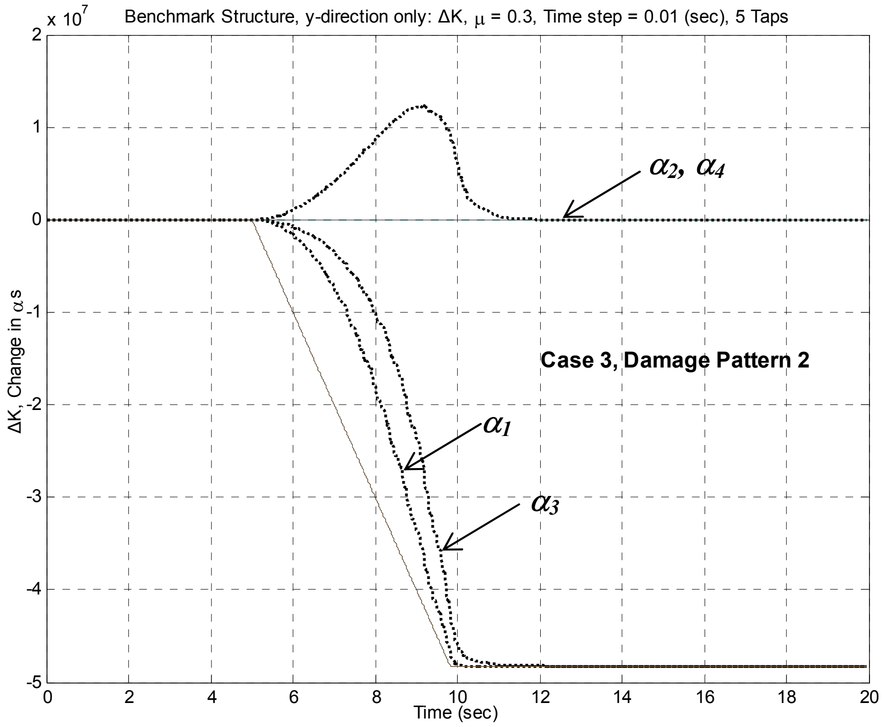


Figure 15 Case 3 and damage pattern 2 using the One Step Method without coupling due to gradual failure

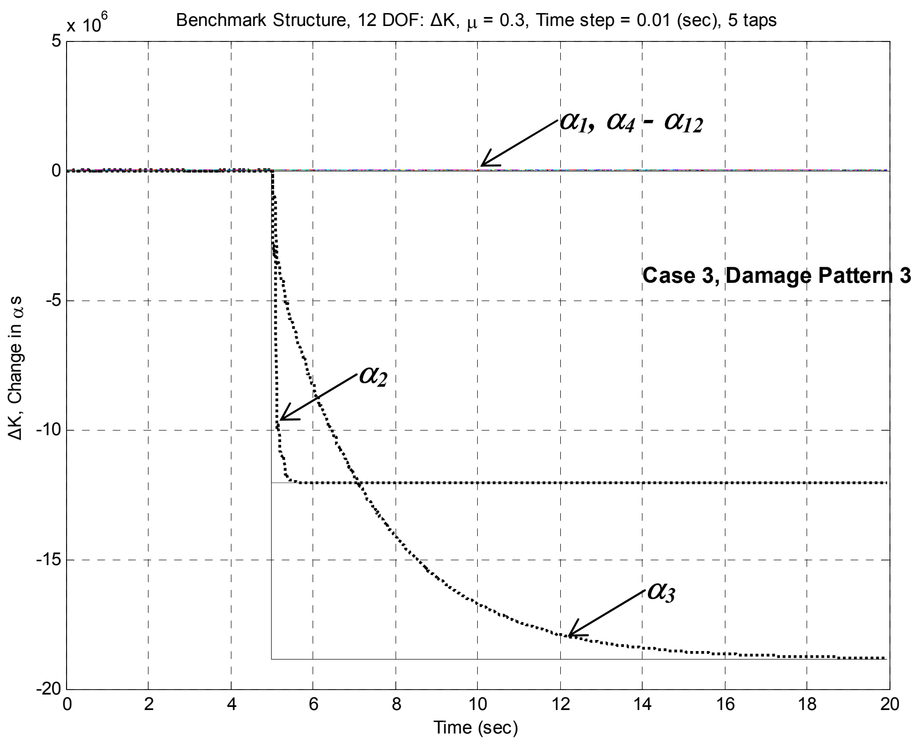


Figure 16 Case 3 and damage pattern 3 using the One Step Method without coupling due to sudden failure

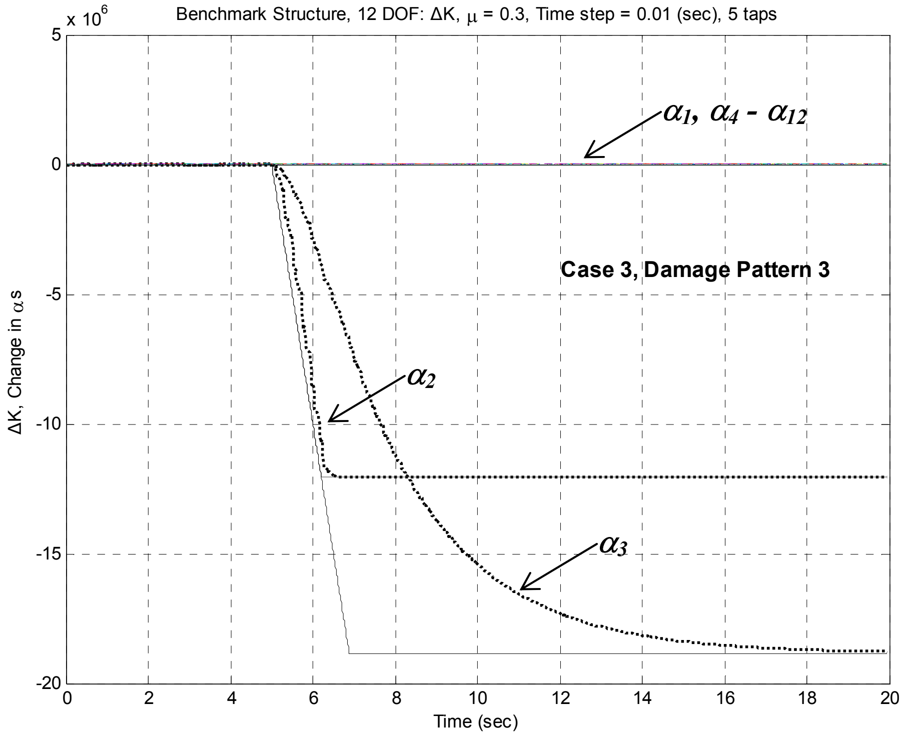


Figure 17 Case 3 and damage pattern 3 using the One Step Method without coupling due to gradual failure

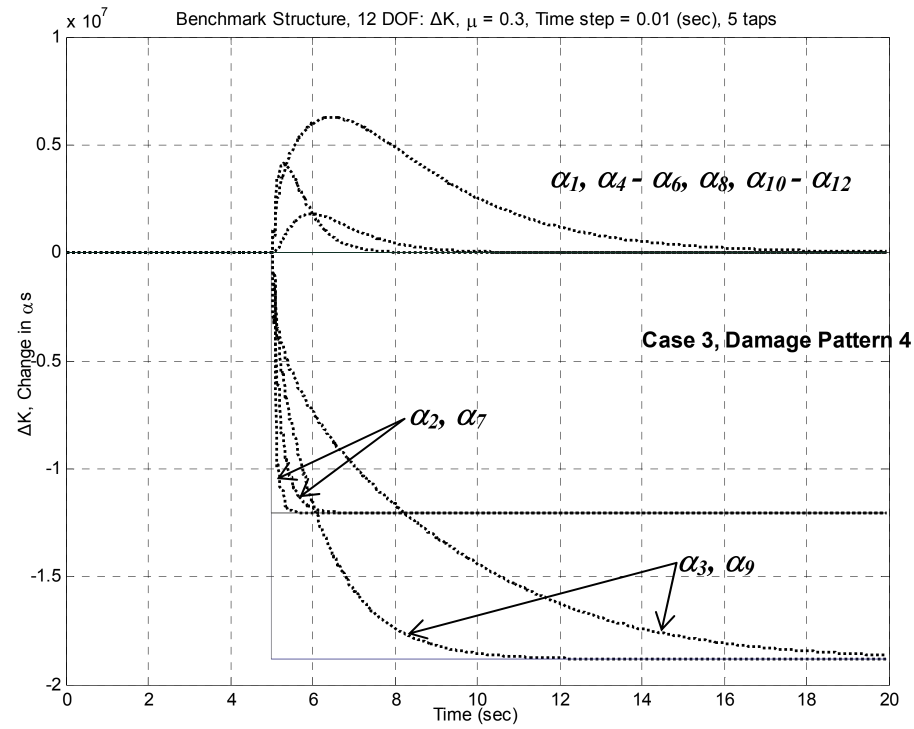


Figure 18 Case 3 and damage pattern 4 using the One Step Method without coupling due to sudden failure

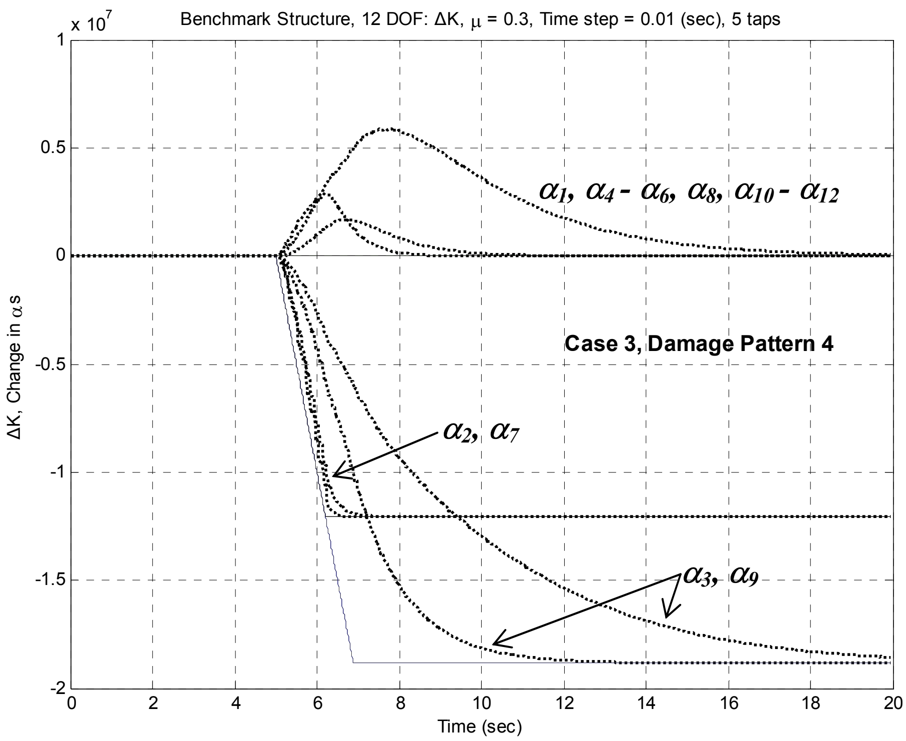


Figure 19 Case 3 and damage pattern 4 using the One Step Method without coupling due to gradual failure

Figure 20 and Figure 21 show the results for case 4 and damage pattern 1, which were simulated with the 12 DOF model. As shown in Figure 20 and Figure 21, because there are three DOF per floor in the 12 DOF model, the damage in the first floor requires changes in three α_i values for all DOF of the 1st floor. Hence, in Figure 20 and Figure 21, two more α_i values are changing than when the 4 DOF model is used in Figure 4 and Figure 5 for case 1. Similar results can also be seen in Figure 12 and Figure 13 for case 3 with the same damage pattern 1.

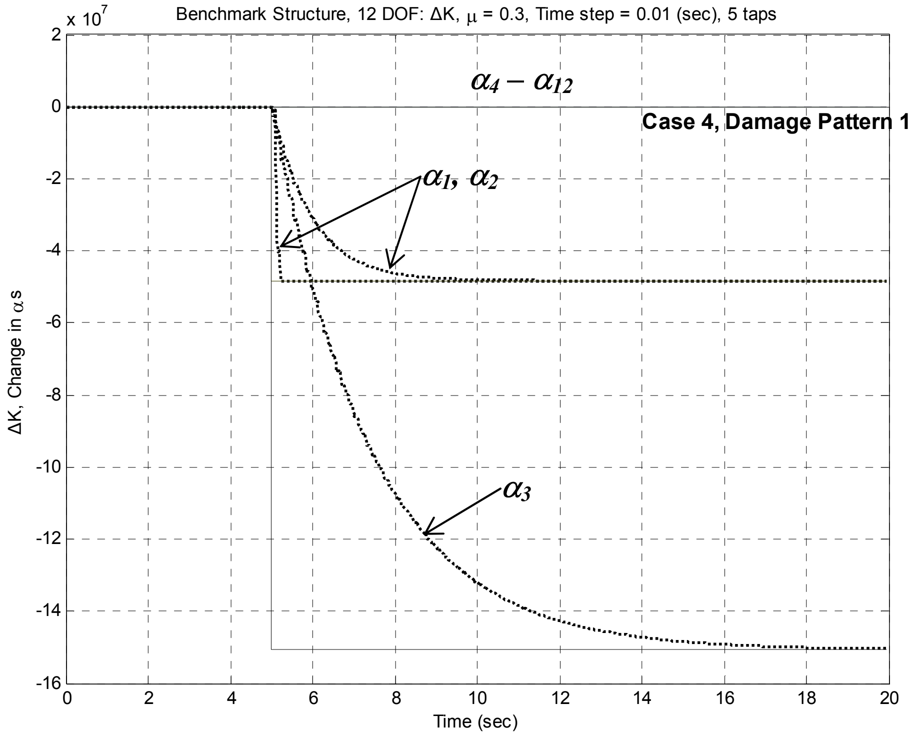


Figure 20 Case 4 and damage pattern 1 using the One Step Method without coupling due to sudden failure

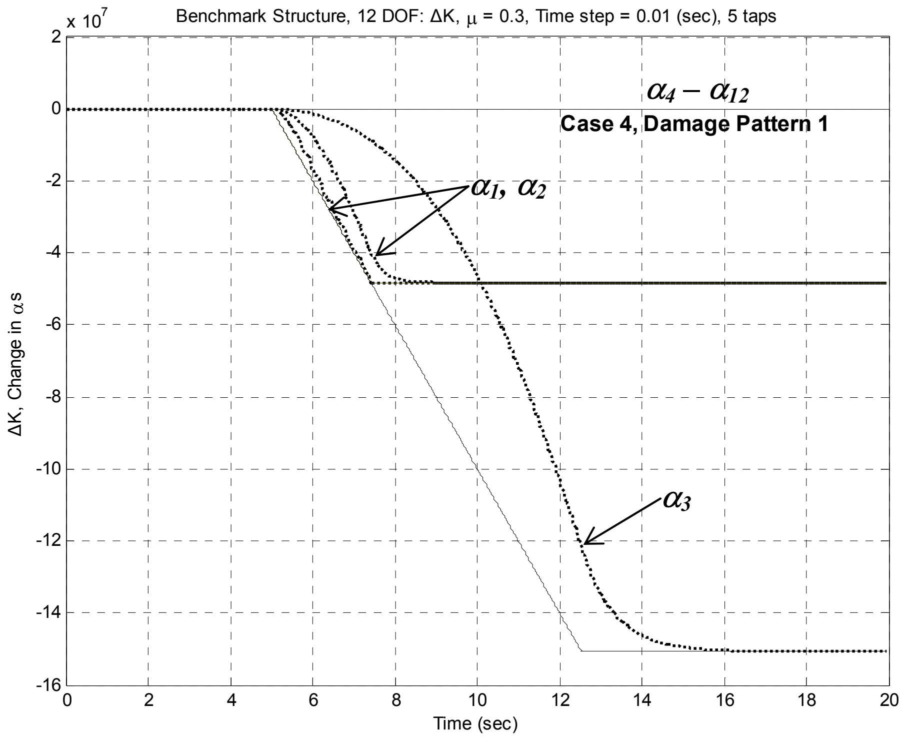


Figure 21 Case 4 and damage pattern 1 using the One Step Method without coupling due to gradual failure

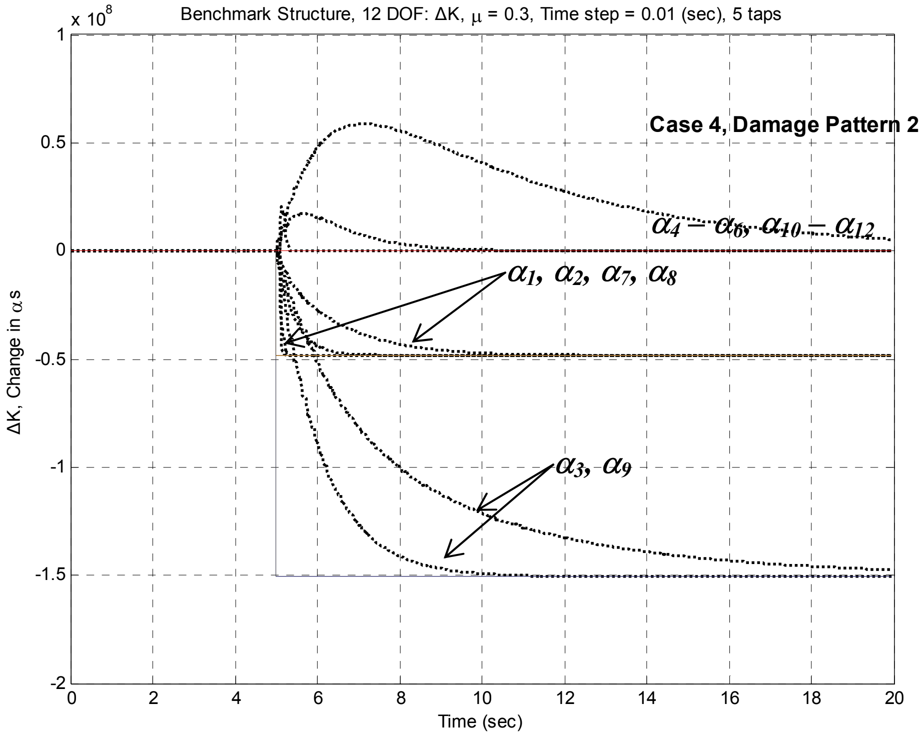


Figure 22 Case 4 and damage pattern 2 using the One Step Method without coupling due to sudden failure

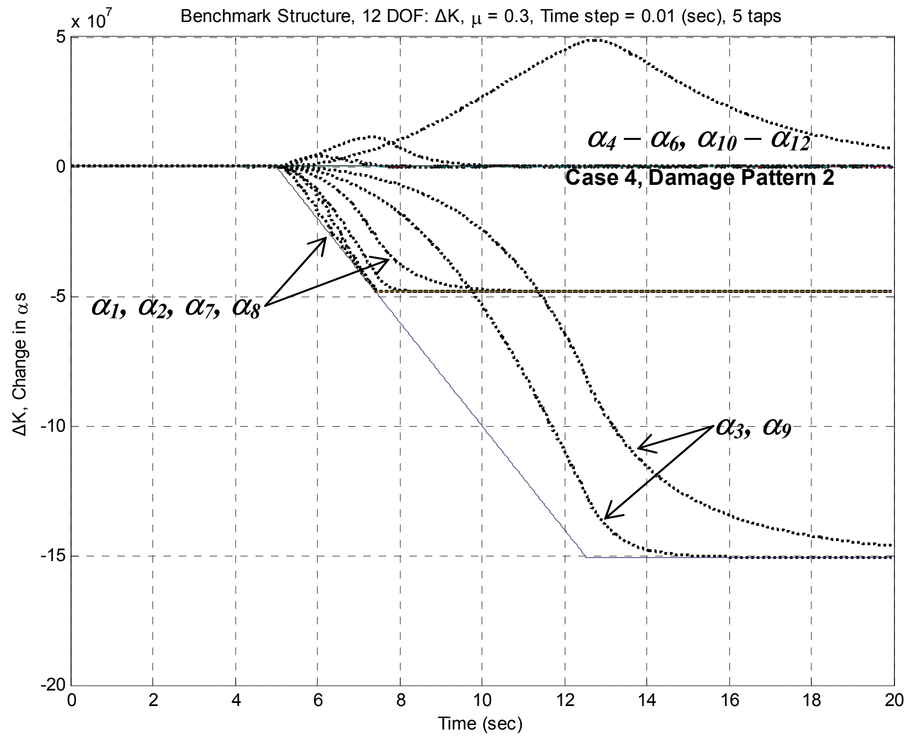


Figure 23 Case 4 and damage pattern 2 using the One Step Method without coupling due to gradual failure

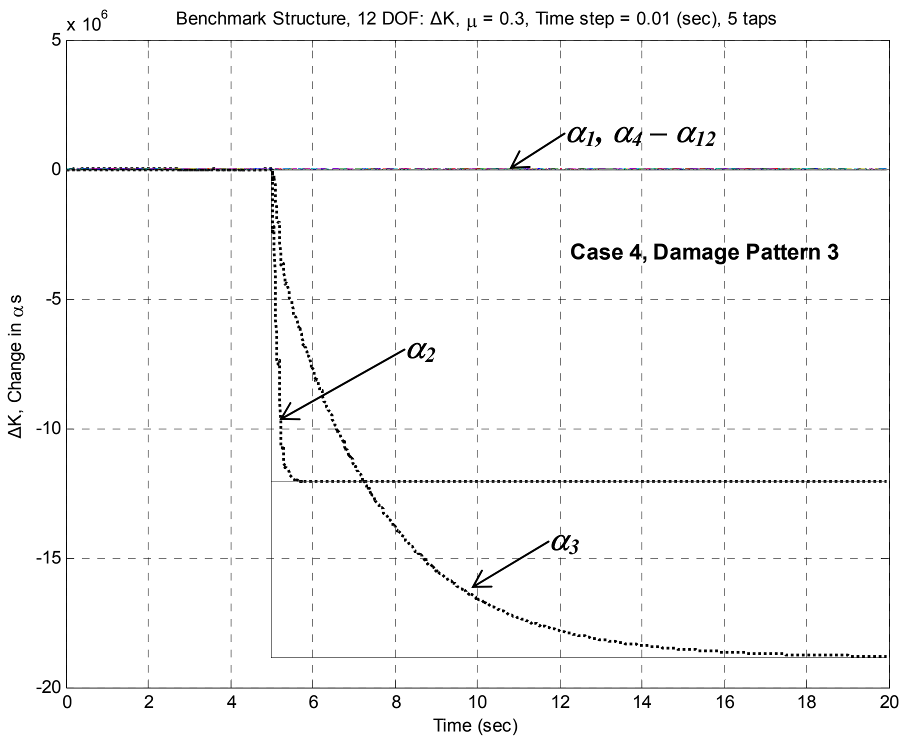


Figure 24 Case 4 and damage pattern 3 using the One Step Method without coupling due to sudden failure

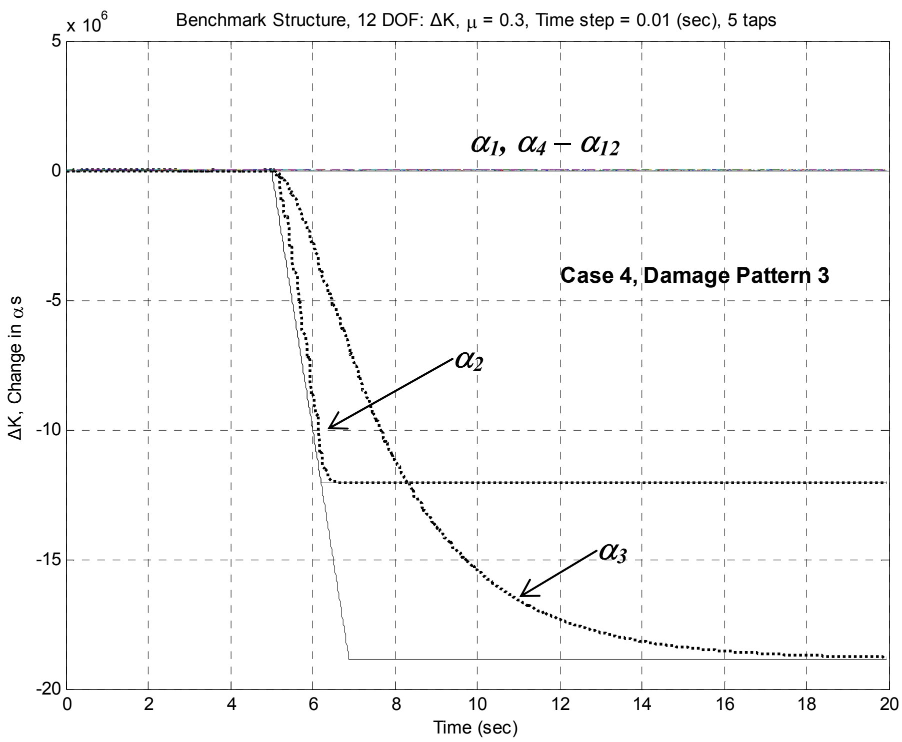


Figure 25 Case 4 and damage pattern 3 using the One Step Method without coupling due to gradual failure

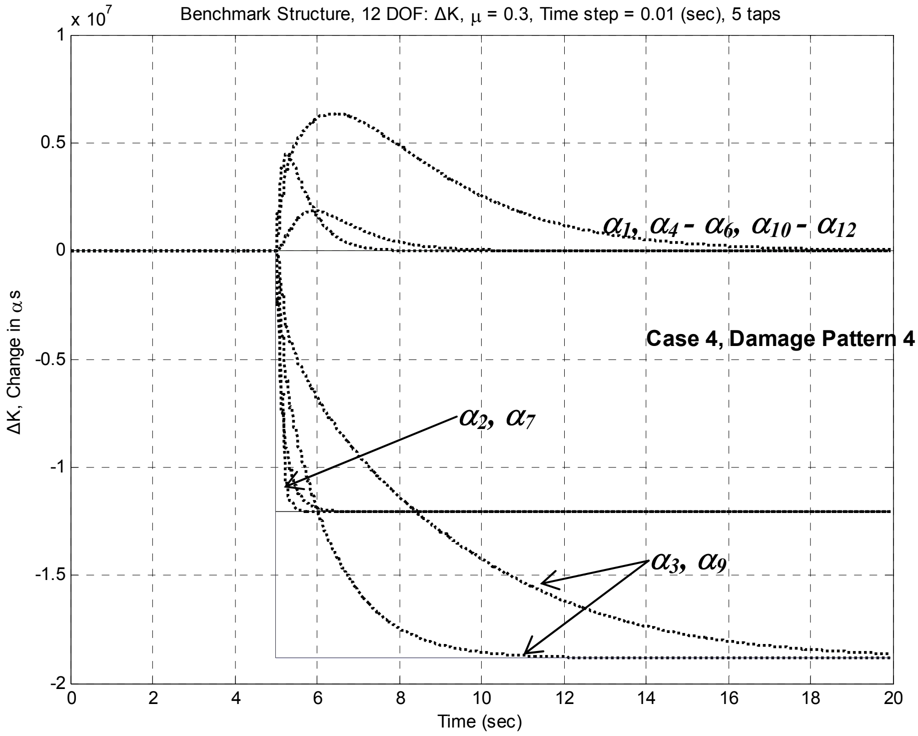


Figure 26 Case 4 and damage pattern 4 using the One Step Method without coupling due to sudden failure

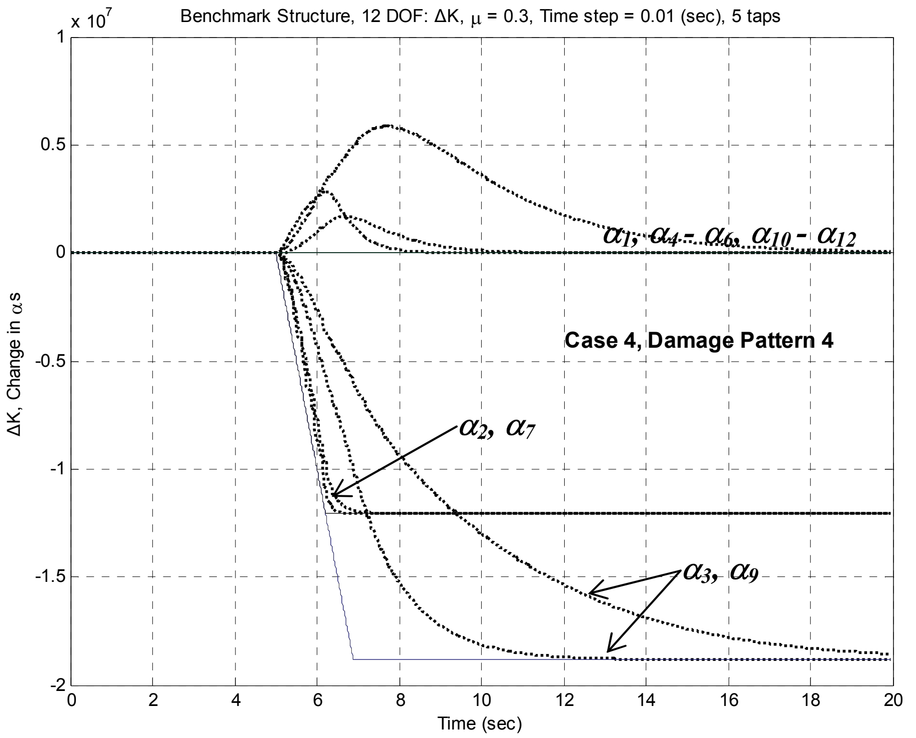


Figure 27 Case 4 and damage pattern 4 using the One Step Method without coupling due to gradual failure

7.2. *One Step Method with Coupling*

In this section, the results from the simulation of the models using the One Step method with coupling term involved in the gradient calculation, as in Equation (25), are presented. Generally, for results of 4 DOF model simulations, Figure 28 – Figure 31 and Figure 36 – Figure 39, it took a slightly longer time to converge using the One Step method with the coupling terms than when the method without coupling is used. Particularly, when there are no further changes in stiffness, the convergence rate is significantly reduced for the One Step method with coupling terms. This trend will be discussed in a later section.

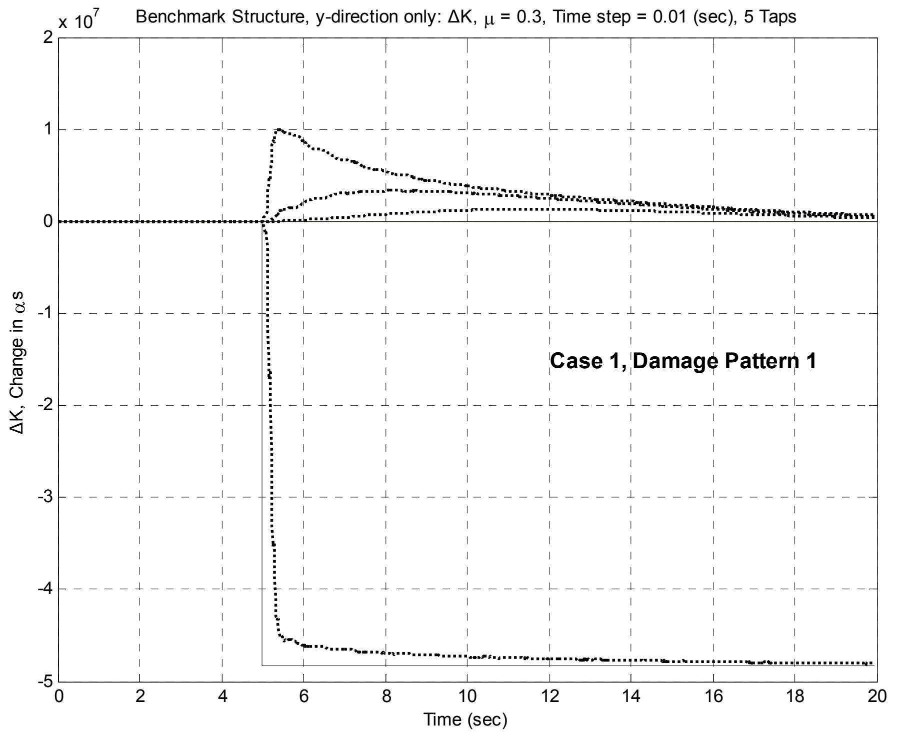


Figure 28 Case 1 and damage pattern 1 using the One Step Method with coupling due to sudden failure

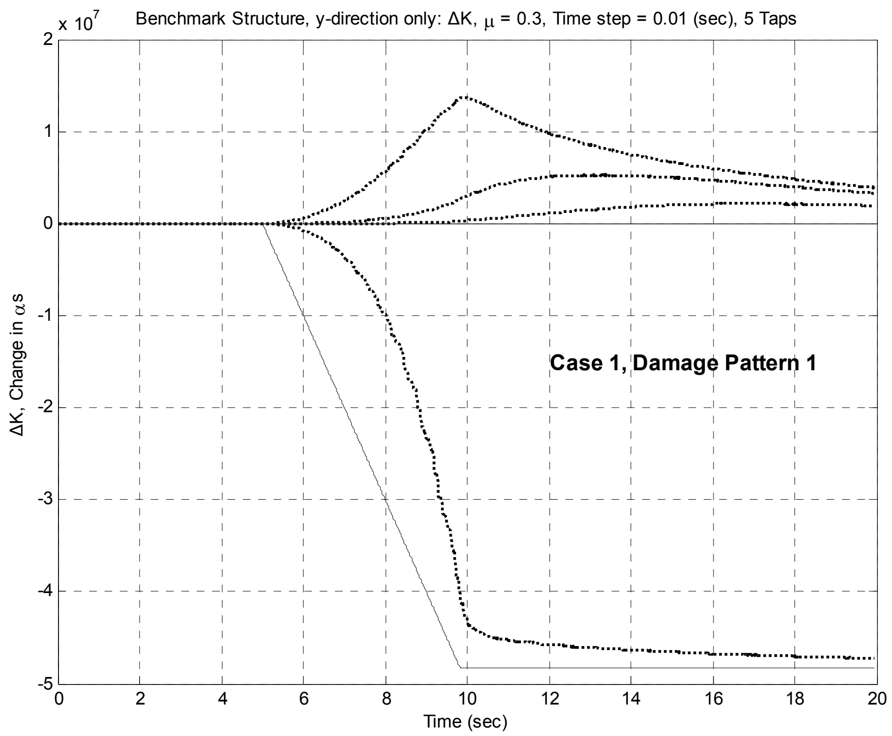


Figure 29 Case 1 and damage pattern 1 using the One Step Method with coupling due to gradual failure

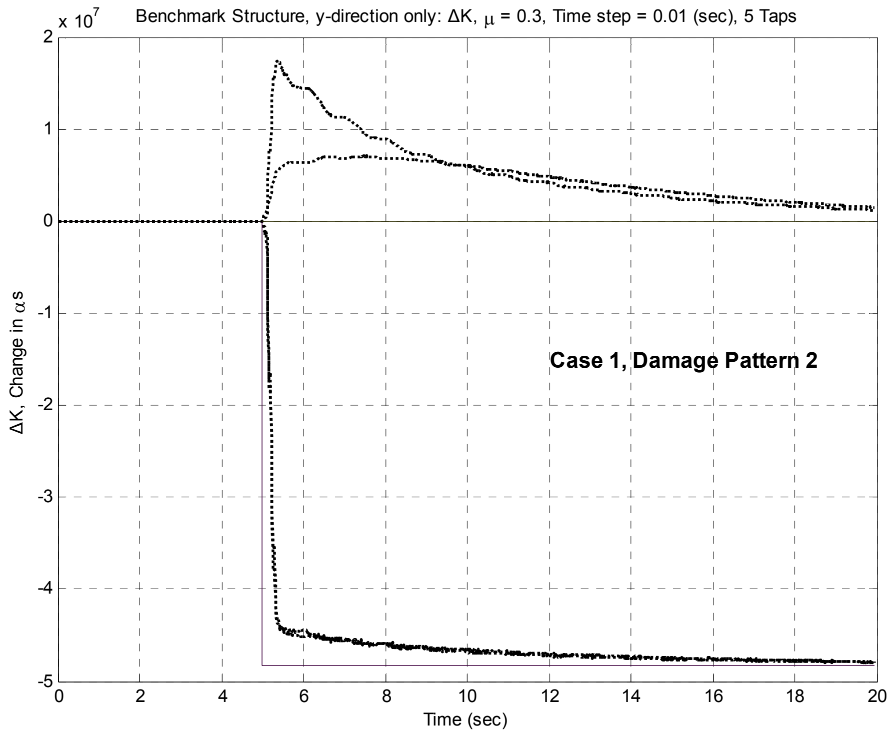


Figure 30 Case 1 and damage pattern 2 using the One Step Method with coupling due to sudden failure

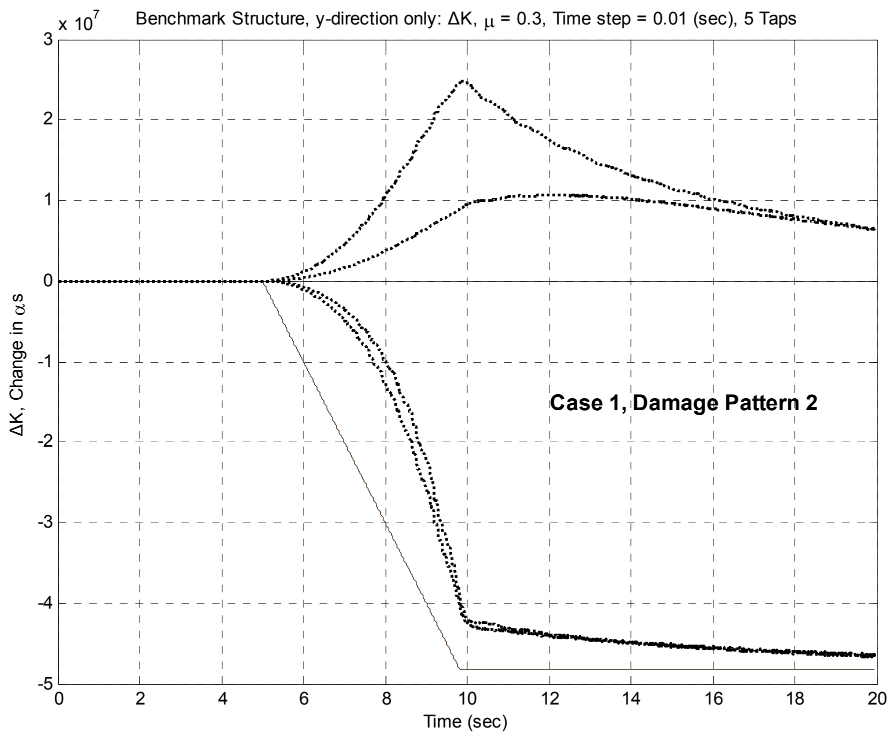


Figure 31 Case 1 and damage pattern 2 using the One Step Method with coupling due to gradual failure

Figure 32 and Figure 33 show the results for case 1 and damage pattern 3 using the One Step method with coupling. The 12 DOF model was used for simulating damage pattern 3. The method was having difficulty in identifying the changes within 20 seconds. The fastest convergence time to 95% of the actual change is about 9 seconds for α_2 in Figure 32. It is found that the One Step method with coupling has relatively slower convergence for damage patterns 3 and 4 when the 12 DOF model is used. Figure 40 – Figure 51 are the results of the 12 DOF model simulations for the One Step method with coupling, and have similar trend as in Figure 32 and Figure 33.

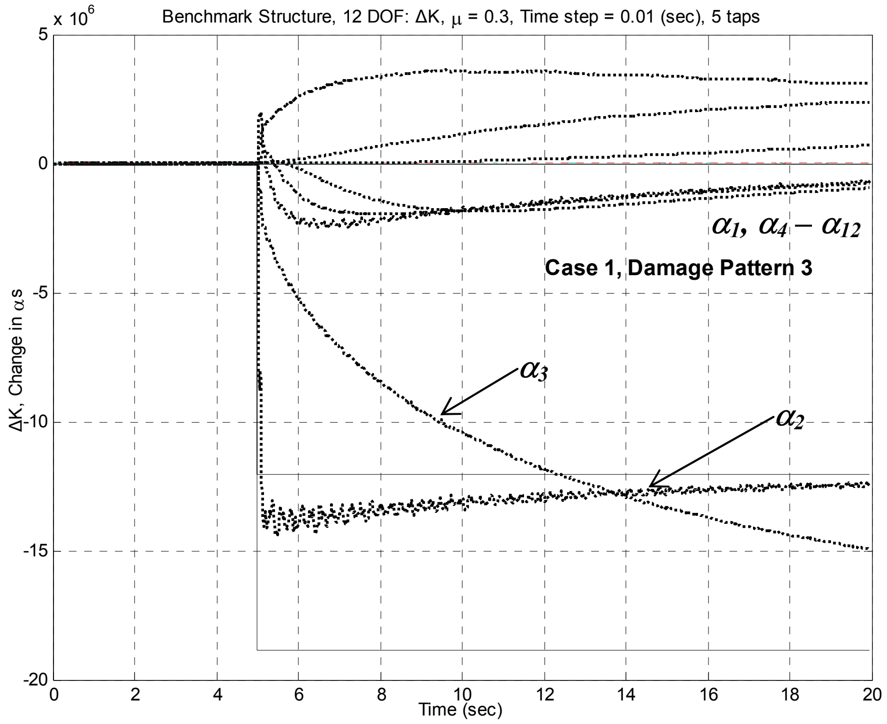


Figure 32 Case 1 and damage pattern 3 using the One Step Method with coupling due to sudden failure

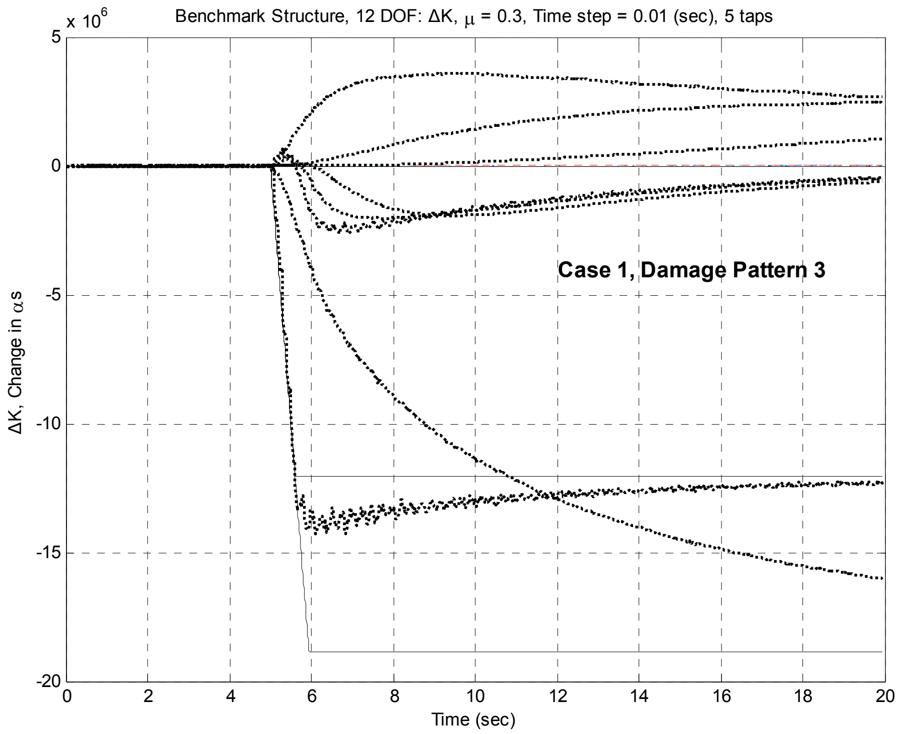


Figure 33 Case 1 and damage pattern 3 using the One Step Method with coupling due to gradual failure

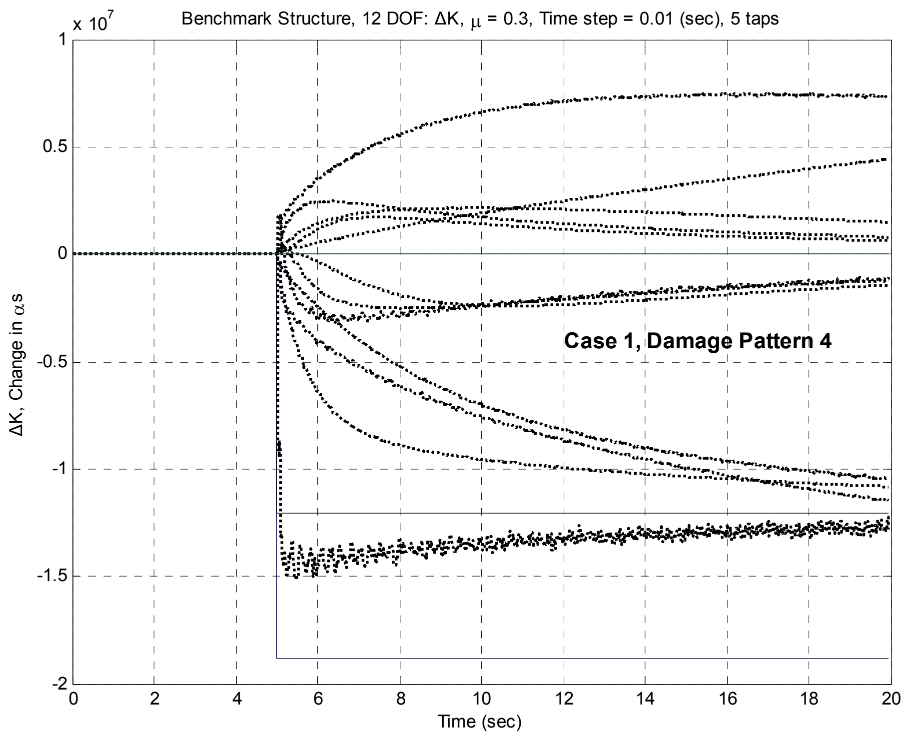


Figure 34 Case 1 and damage pattern 4 using the One Step Method with coupling due to sudden failure

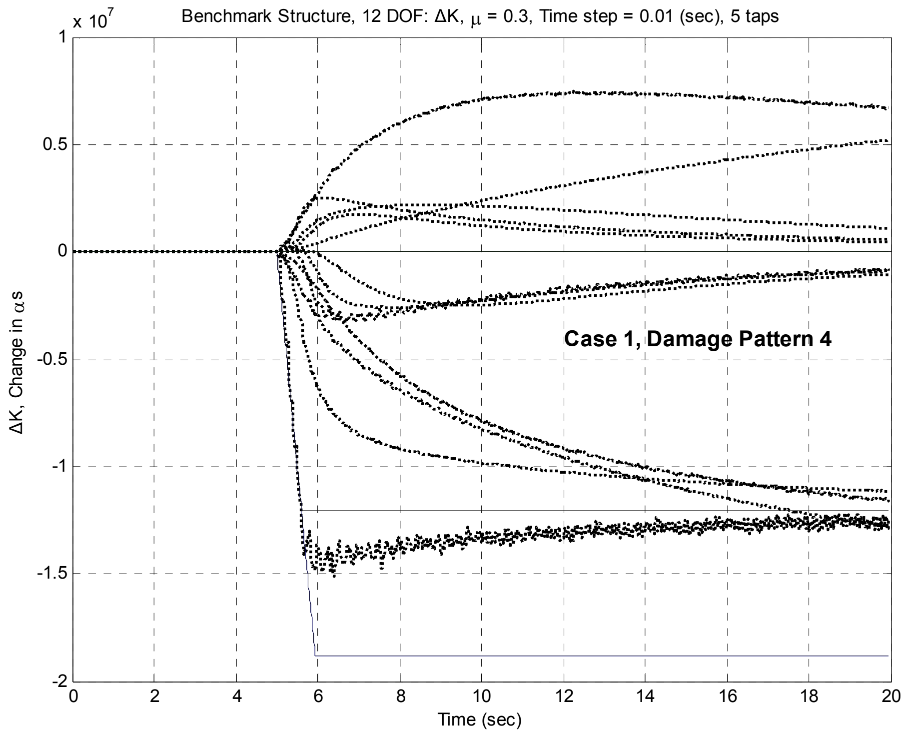


Figure 35 Case 1 and damage pattern 4 using the One Step Method with coupling due to gradual failure

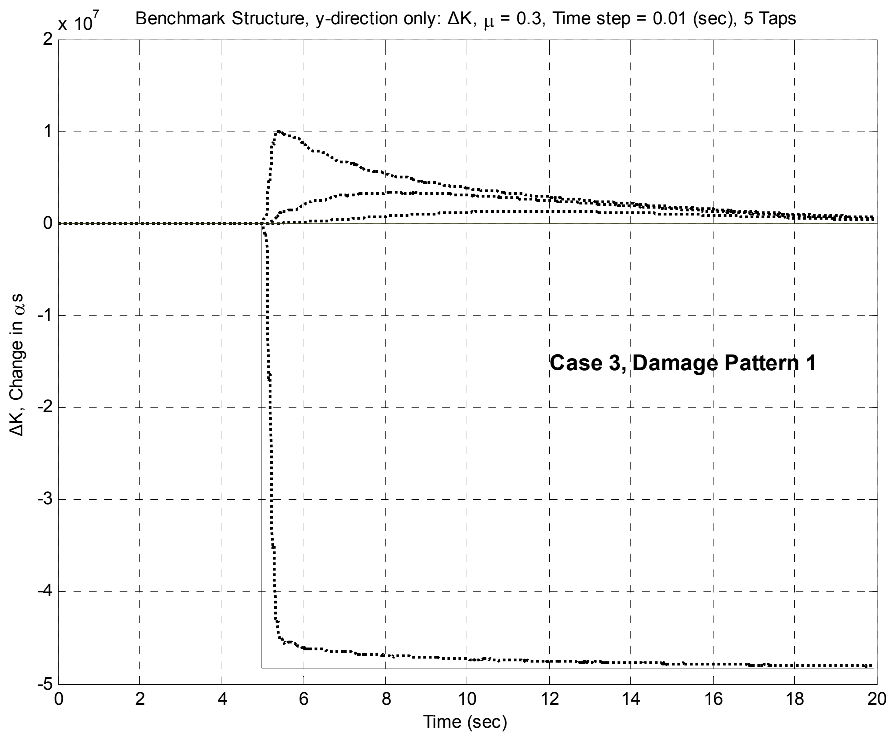


Figure 36 Case 3 and damage pattern 1 using the One Step Method with coupling due to sudden failure

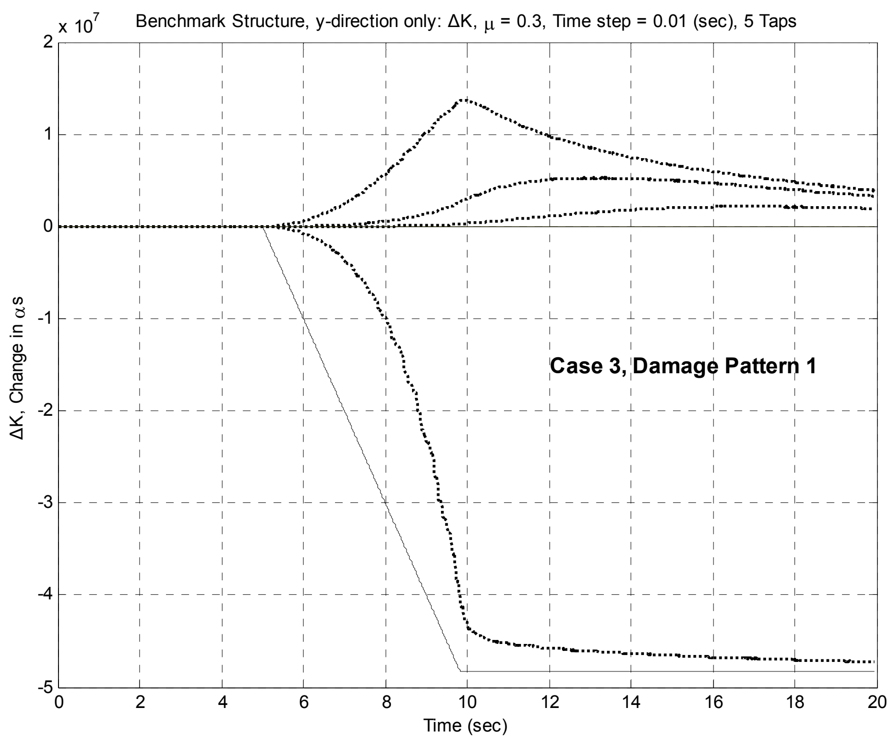


Figure 37 Case 3 and damage pattern 1 using the One Step Method with coupling due to gradual failure

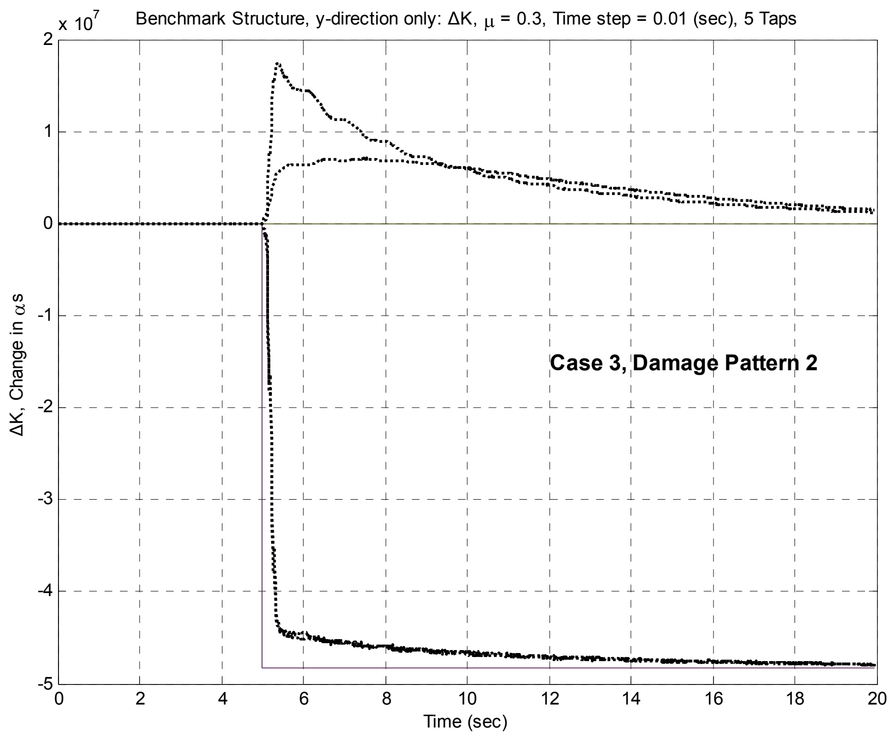


Figure 38 Case 3 and damage pattern 2 using the One Step Method with coupling due to sudden failure

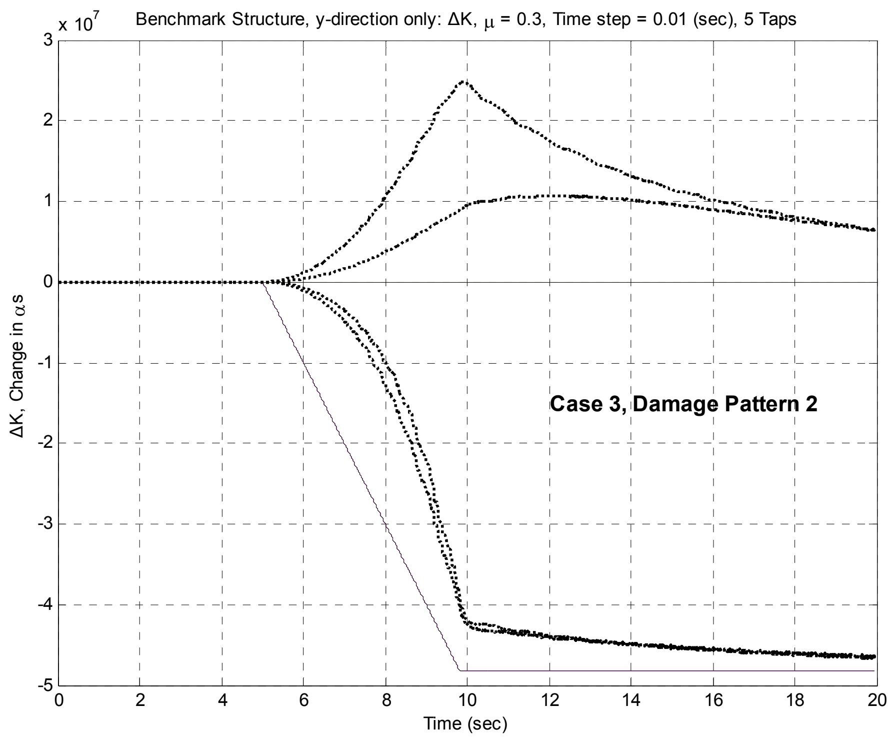


Figure 39 Case 3 and damage pattern 2 using the One Step Method with coupling due to gradual failure

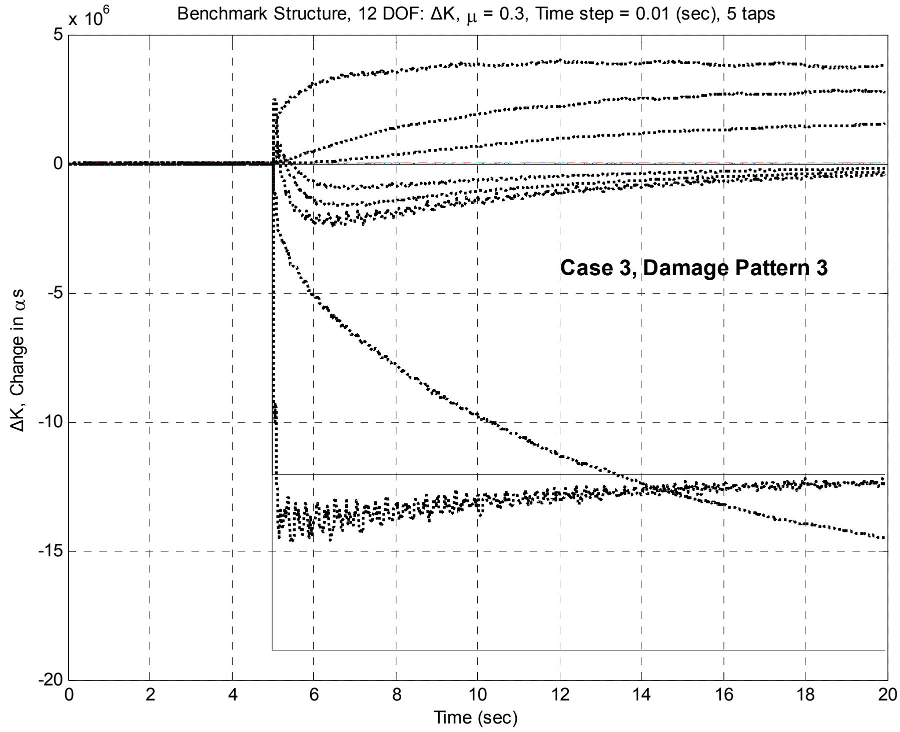


Figure 40 Case 3 and damage pattern 3 using the One Step Method with coupling due to sudden failure

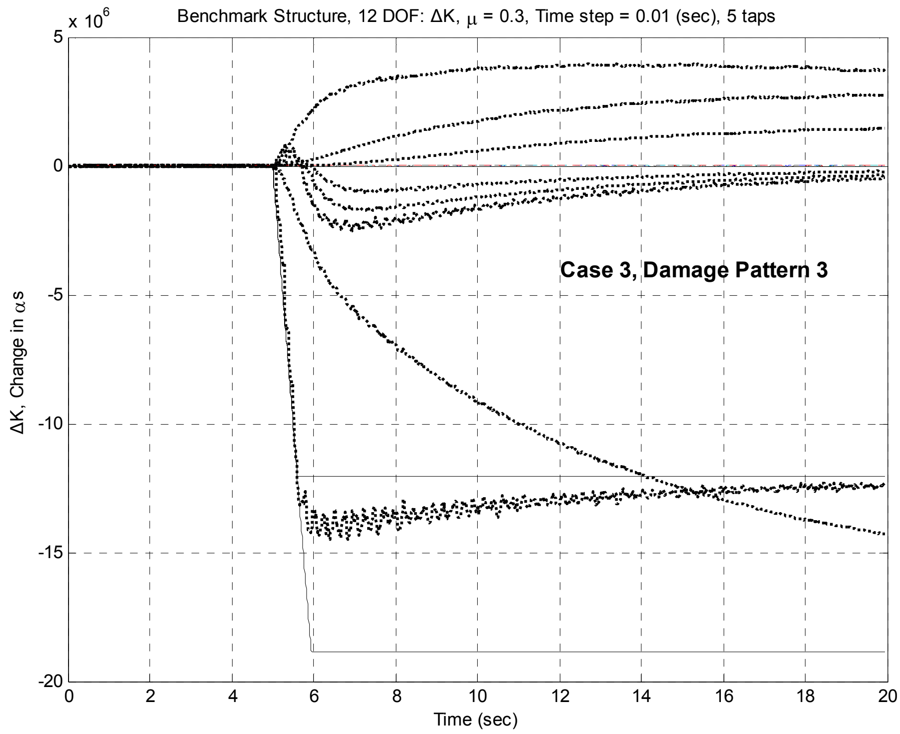


Figure 41 Case 3 and damage pattern 3 using the One Step Method with coupling due to gradual failure

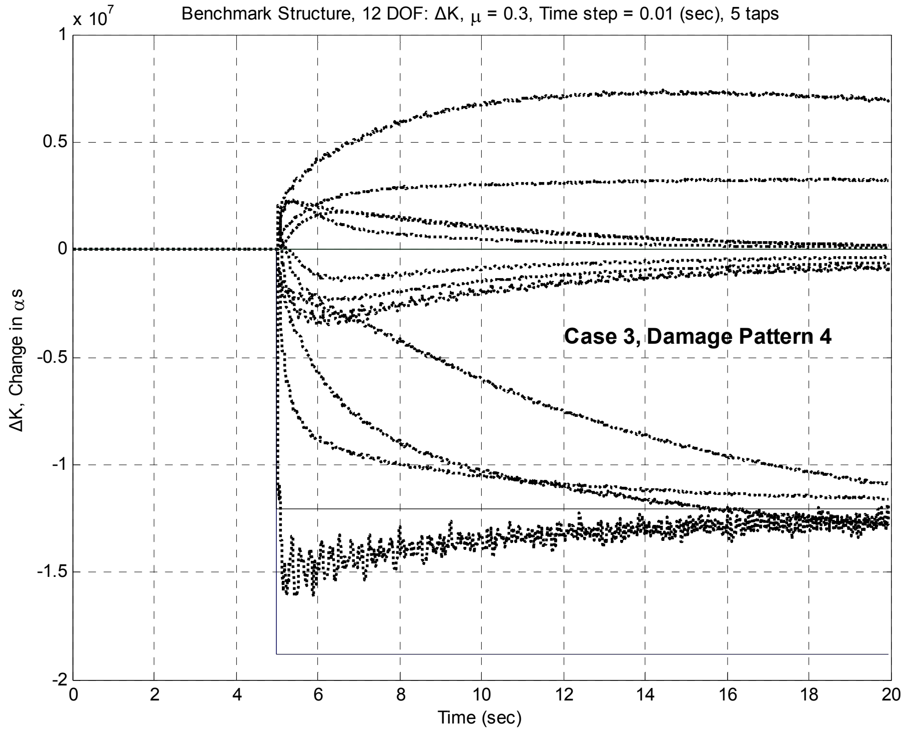


Figure 42 Case 3 and damage pattern 3 using the One Step Method with coupling due to sudden failure

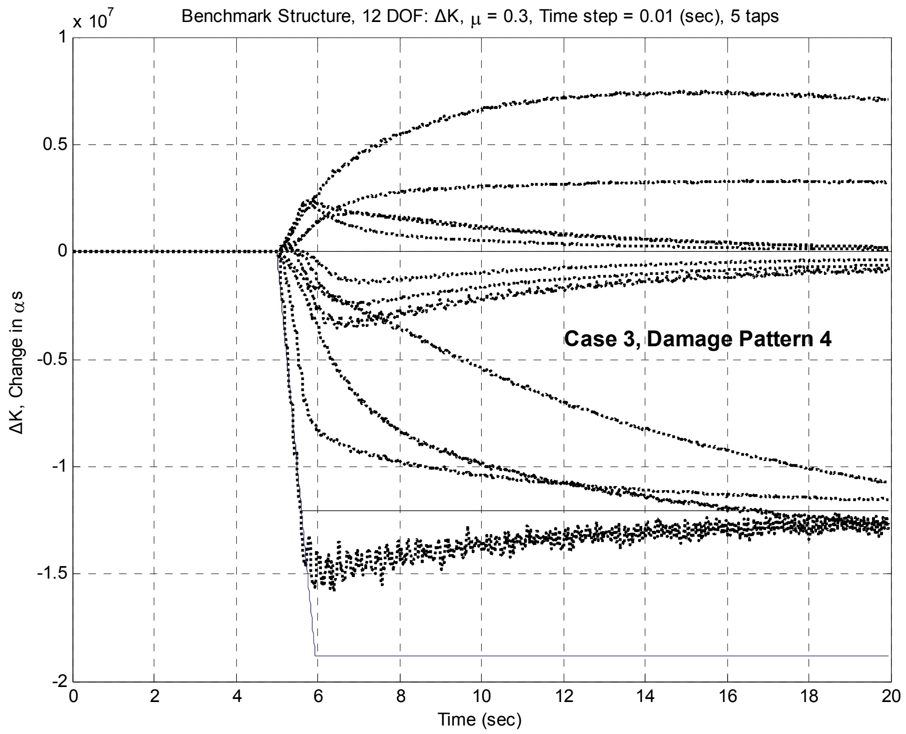


Figure 43 Case 3 and damage pattern 4 using the One Step Method with coupling due to gradual failure

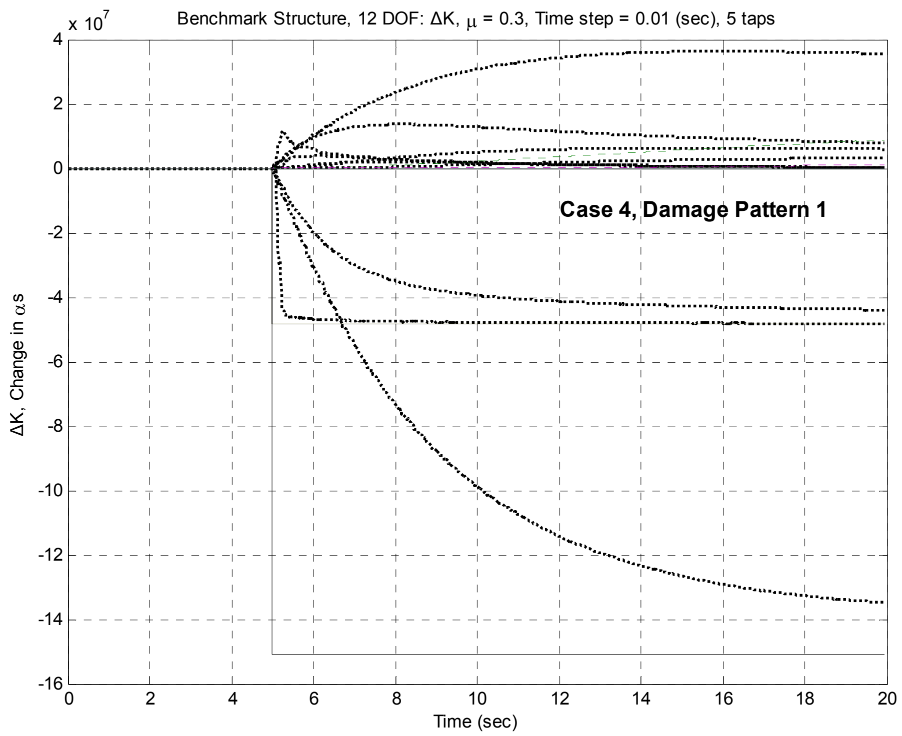


Figure 44 Case 4 and damage pattern 1 using the One Step Method with coupling due to sudden failure

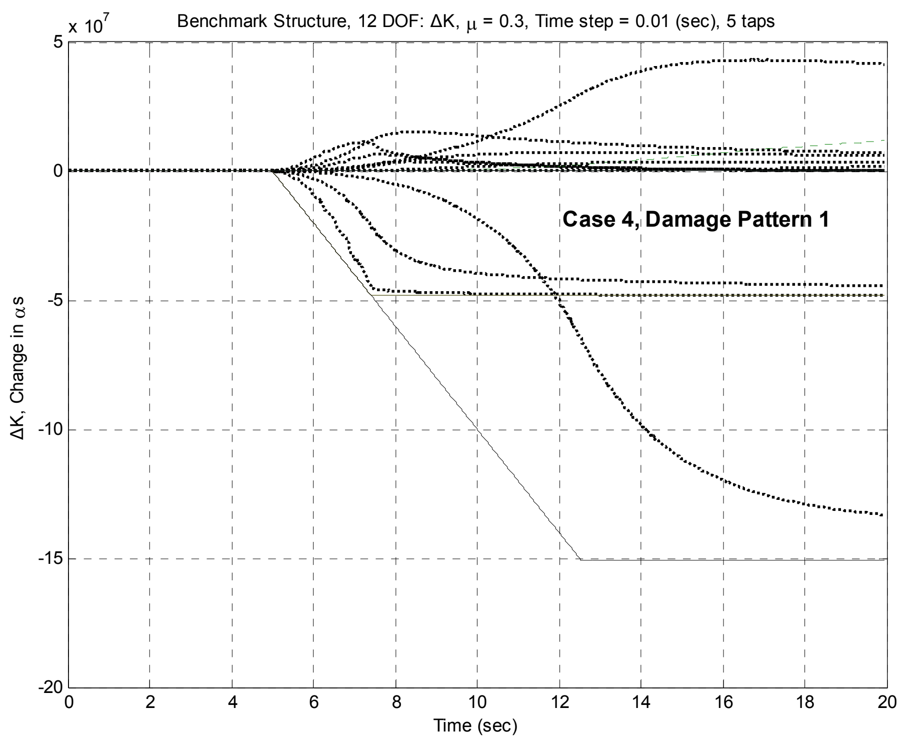


Figure 45 Case 4 and damage pattern 1 using the One Step Method with coupling due to gradual failure

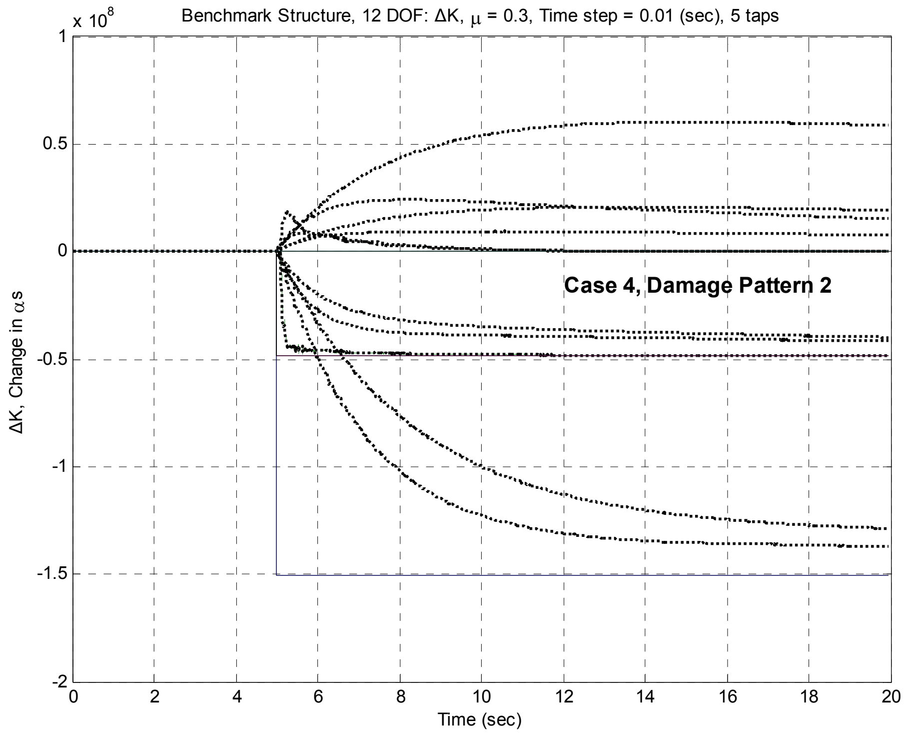


Figure 46 Case 4 and damage pattern 2 using the One Step Method with coupling due to sudden failure

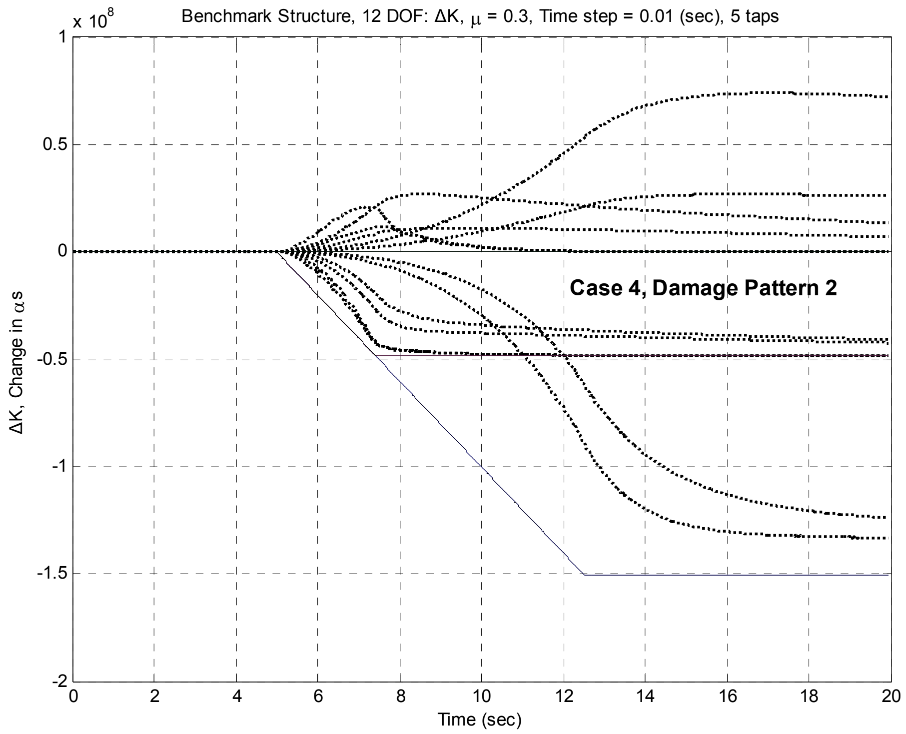


Figure 47 Case 4 and damage pattern 2 using the One Step Method with coupling due to gradual failure

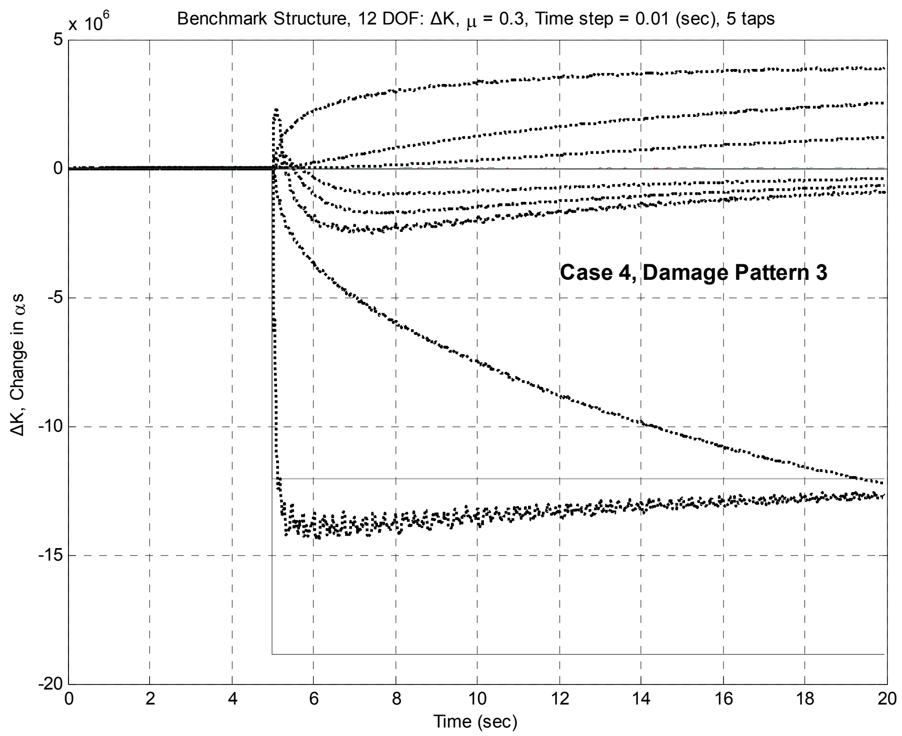


Figure 48 Case 4 and damage pattern 3 using the One Step Method with coupling due to sudden failure

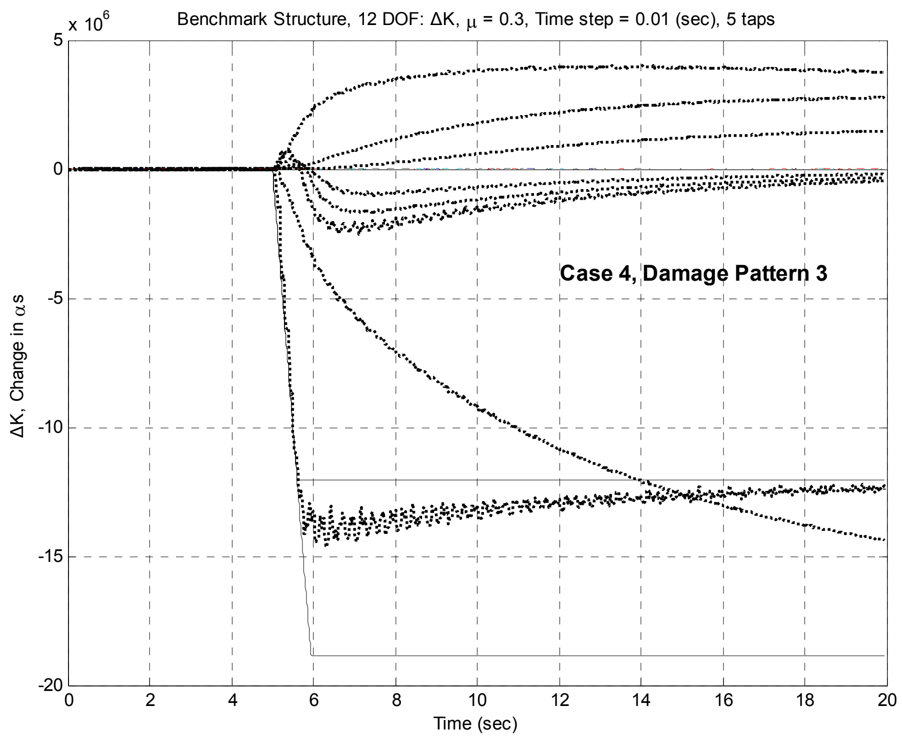


Figure 49 Case 4 and damage pattern 3 using the One Step Method with coupling due to gradual failure

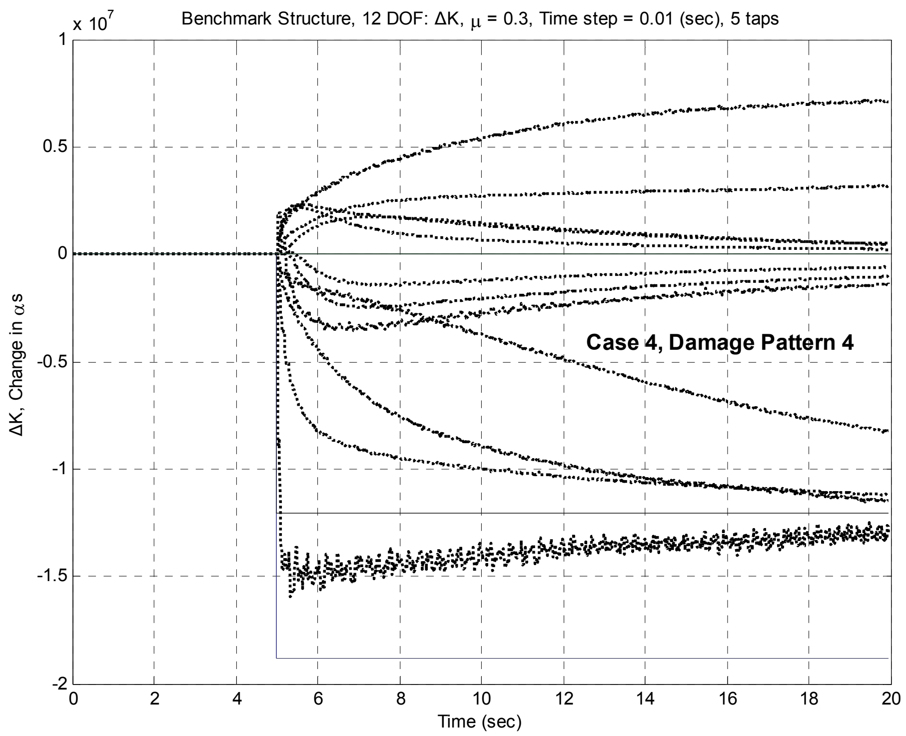


Figure 50 Case 4 and damage pattern 4 using the One Step Method with coupling due to sudden failure

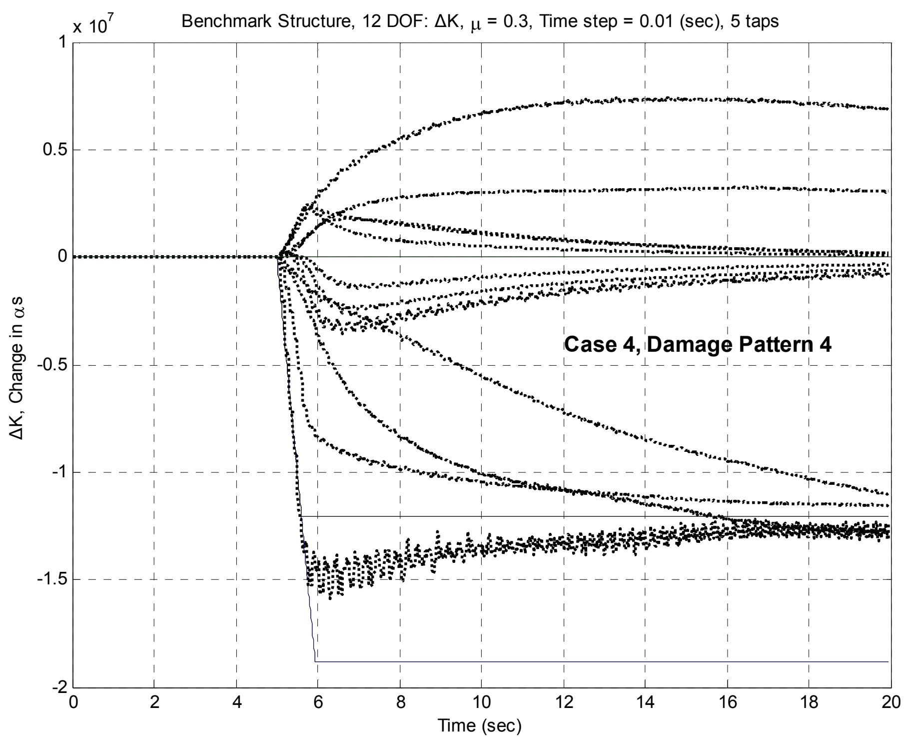


Figure 51 Case 4 and damage pattern 4 using the One Step Method with coupling due to gradual failure

7.3. Two Step Method

The Two step method was expected to have the fastest convergence among the proposed methods because the α values are computed individually with own adaptive LMS filter. Figure 52 – Figure 55 and Figure 60 – Figure 63 clearly show that the Two Step method accurately identifies the changes in stiffness almost instantaneously when the 4 DOF model was simulated. Results of the 12 DOF model simulation, Figure 56 – Figure 59 and Figure 64 – Figure 75, also show that the Two Step method has fastest convergence and accurate identification.

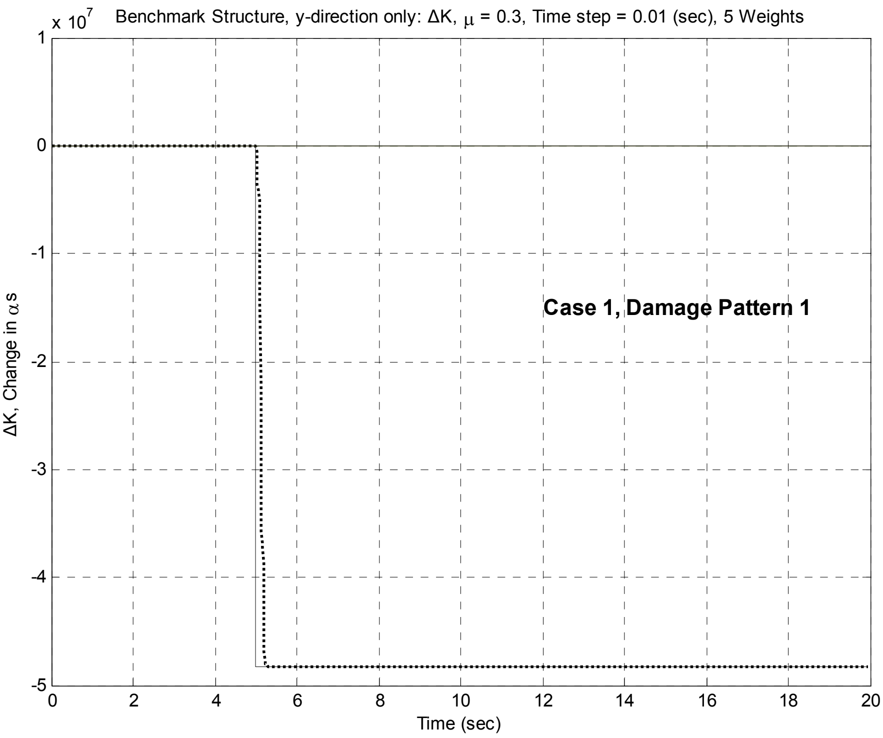


Figure 52 Case 1 and damage pattern 1 using the Two Step Method due to sudden failure

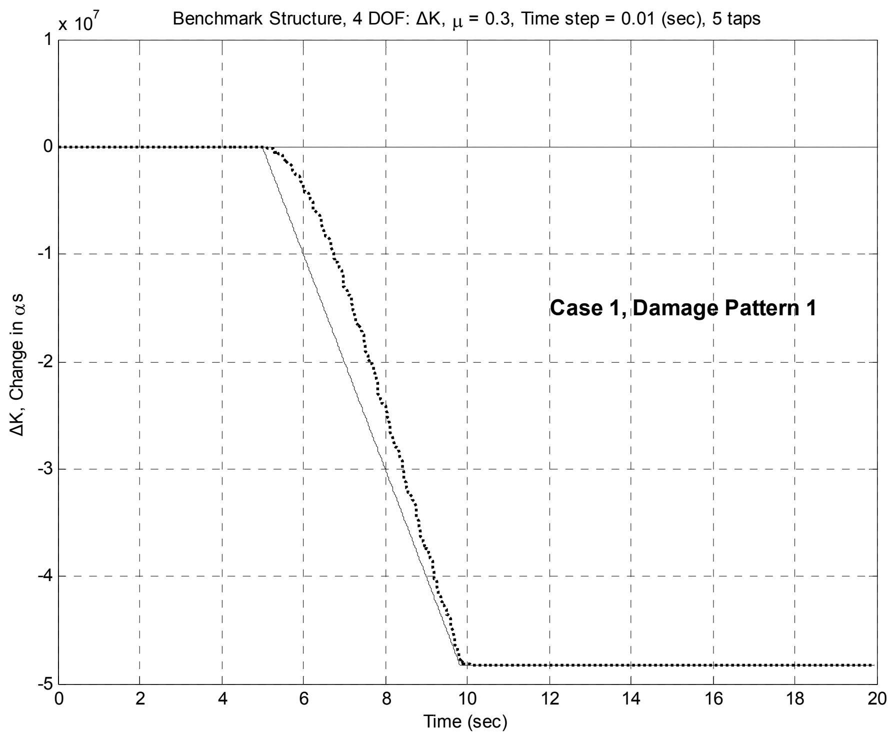


Figure 53 Case 1 and damage pattern 1 using the Two Step Method due to gradual failure

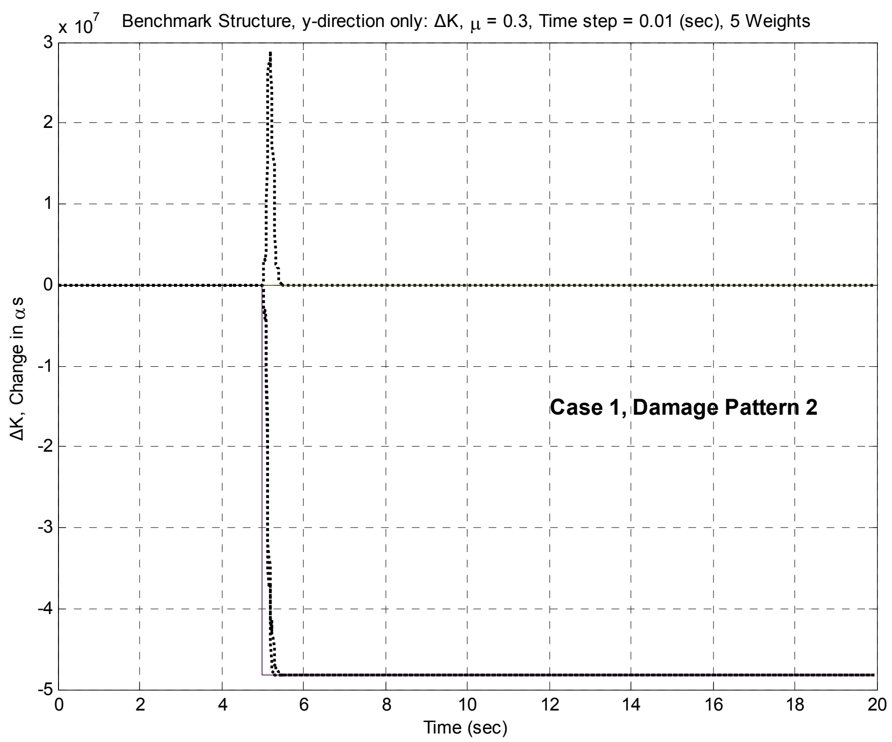


Figure 54 Case 1 and damage pattern 2 using the Two Step Method due to sudden failure

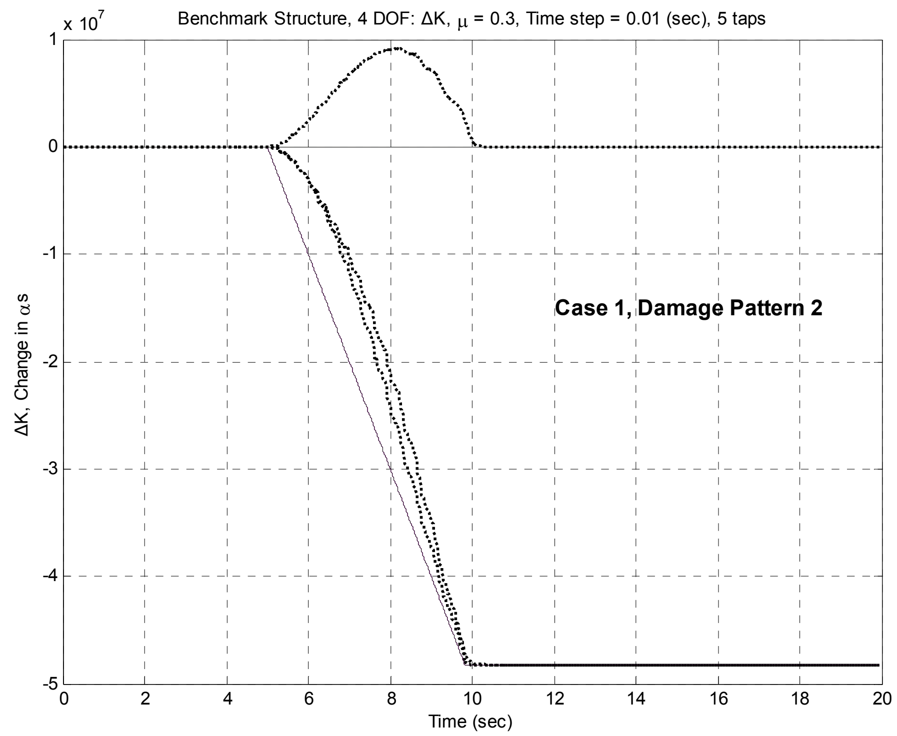


Figure 55 Case 1 and damage pattern 2 using the Two Step Method due to gradual failure

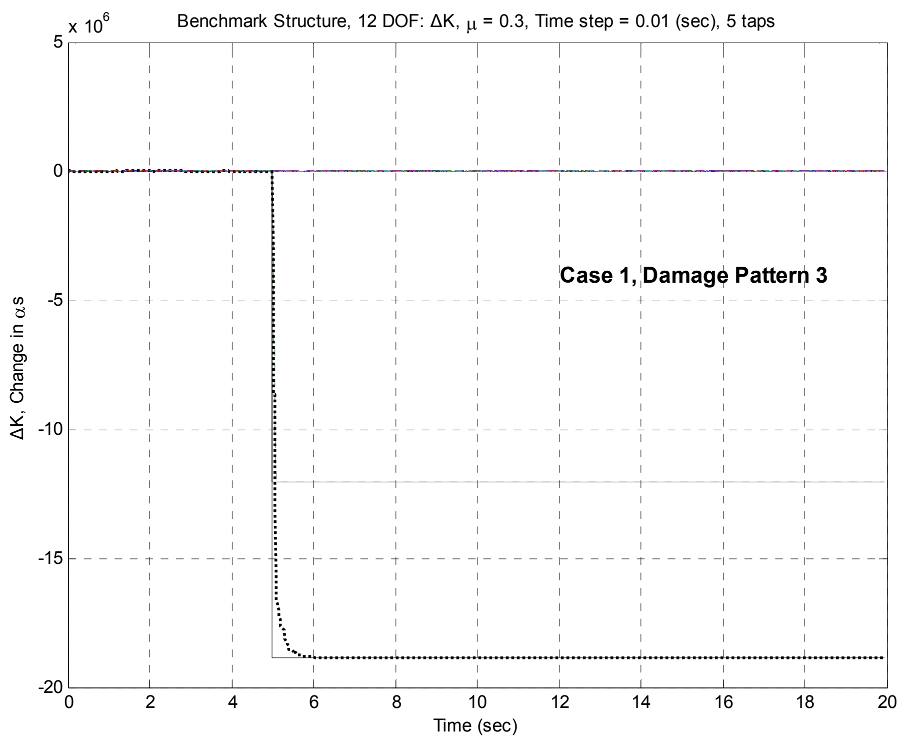


Figure 56 Case 1 and damage pattern 3 using the Two Step Method due to sudden failure

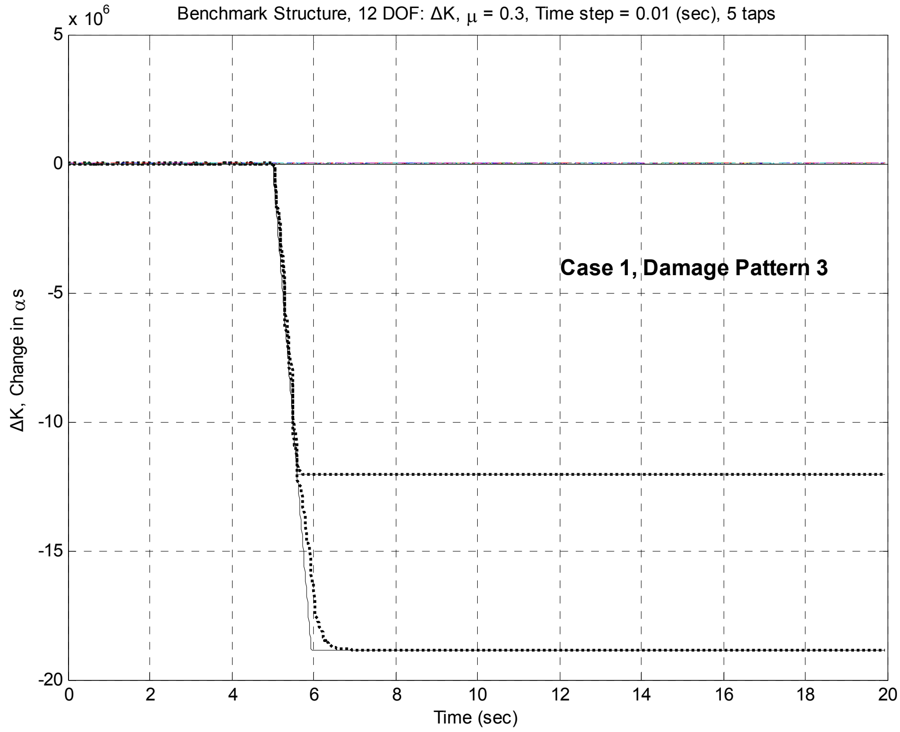


Figure 57 Case 1 and damage pattern 3 using the Two Step Method due to gradual failure

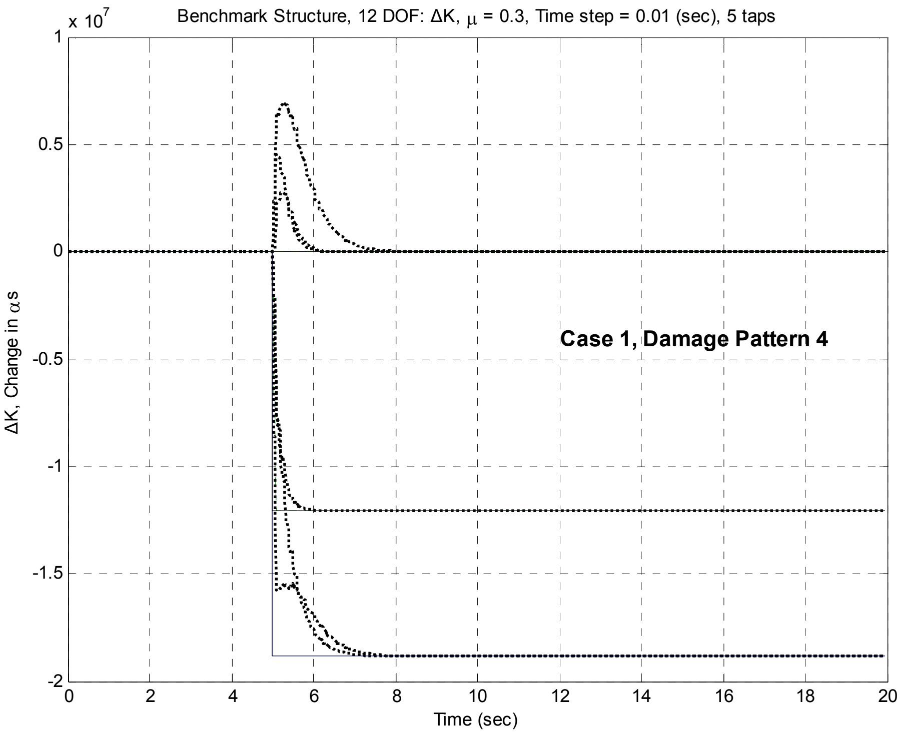


Figure 58 Case 1 and damage pattern 4 using the Two Step Method due to sudden failure

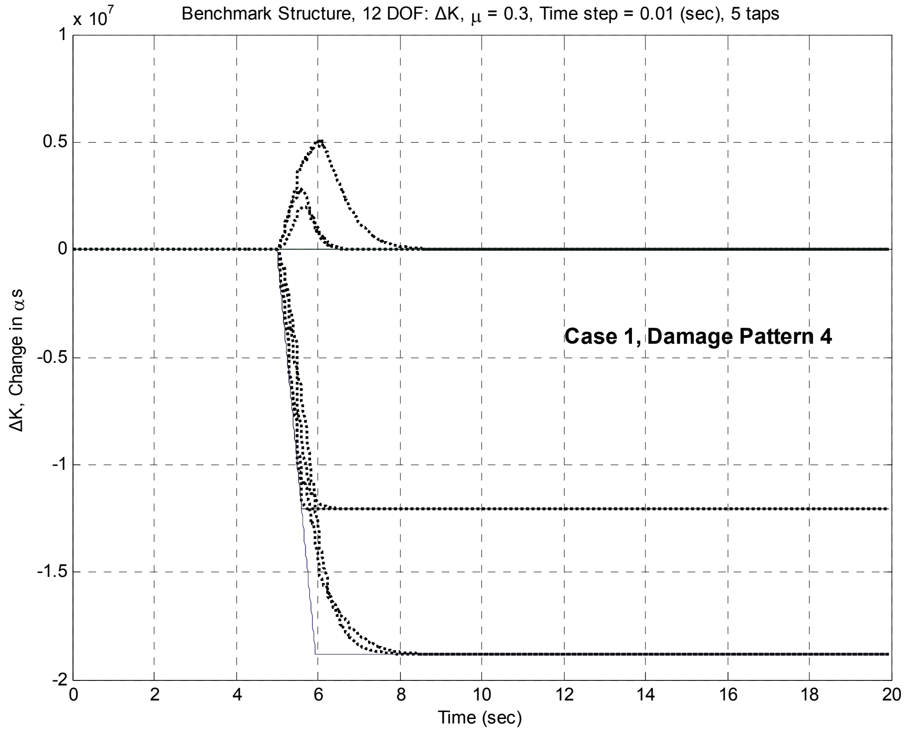


Figure 59 Case 1 and damage pattern 4 using the Two Step Method due to gradual failure

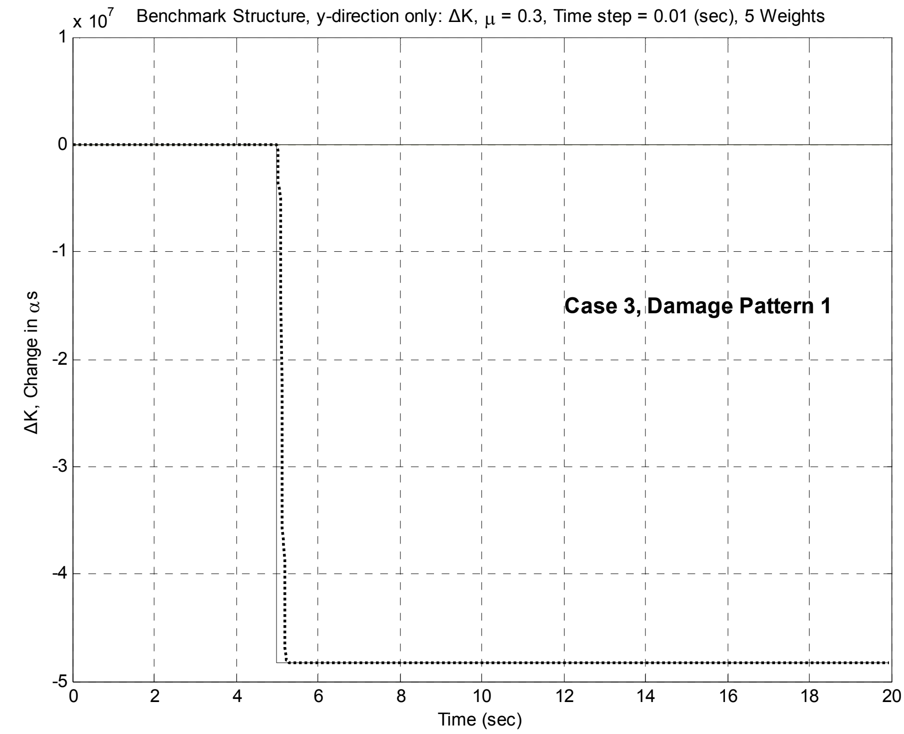


Figure 60 Case 3 and damage pattern 1 using the Two Step Method due to sudden failure

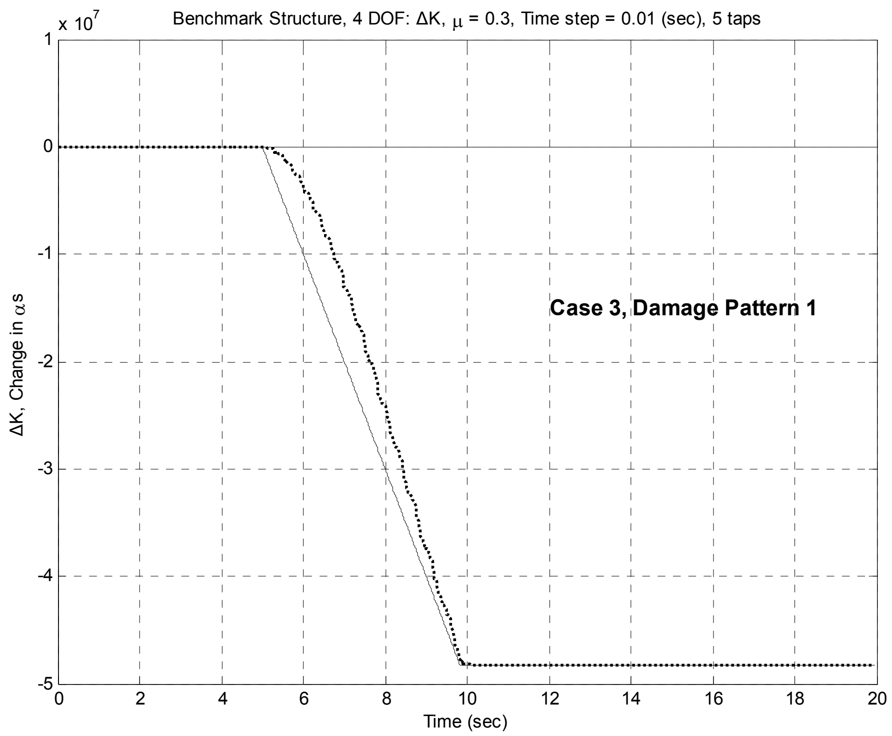


Figure 61 Case 3 and damage pattern 1 using the Two Step Method due to gradual failure

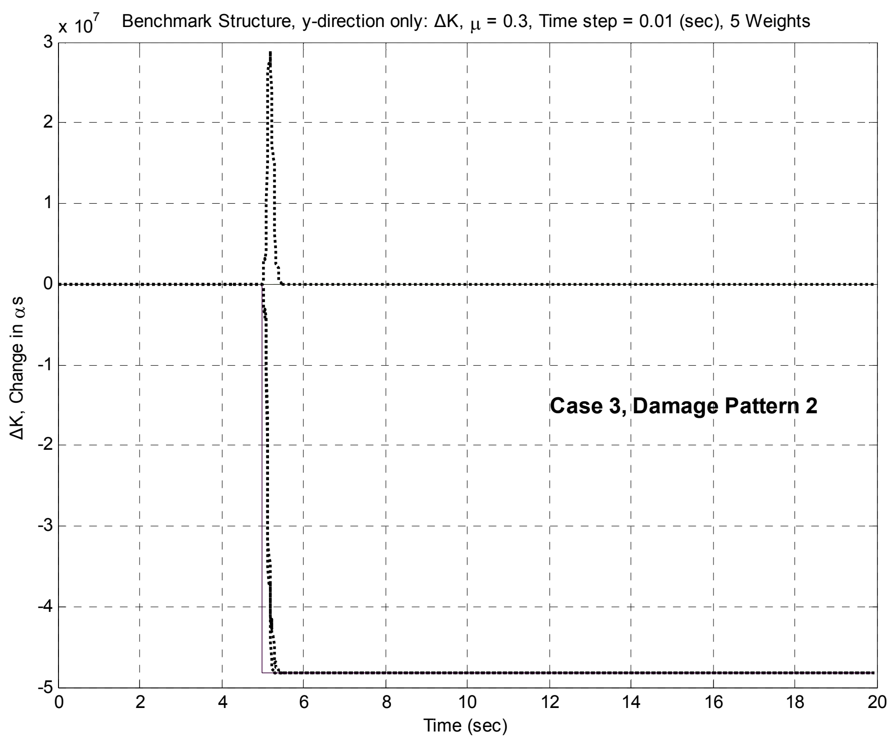


Figure 62 Case 3 and damage pattern 2 using the Two Step Method due to sudden failure

Figure 63 Case 3 and damage pattern 2 using the Two Step Method due to gradual failure

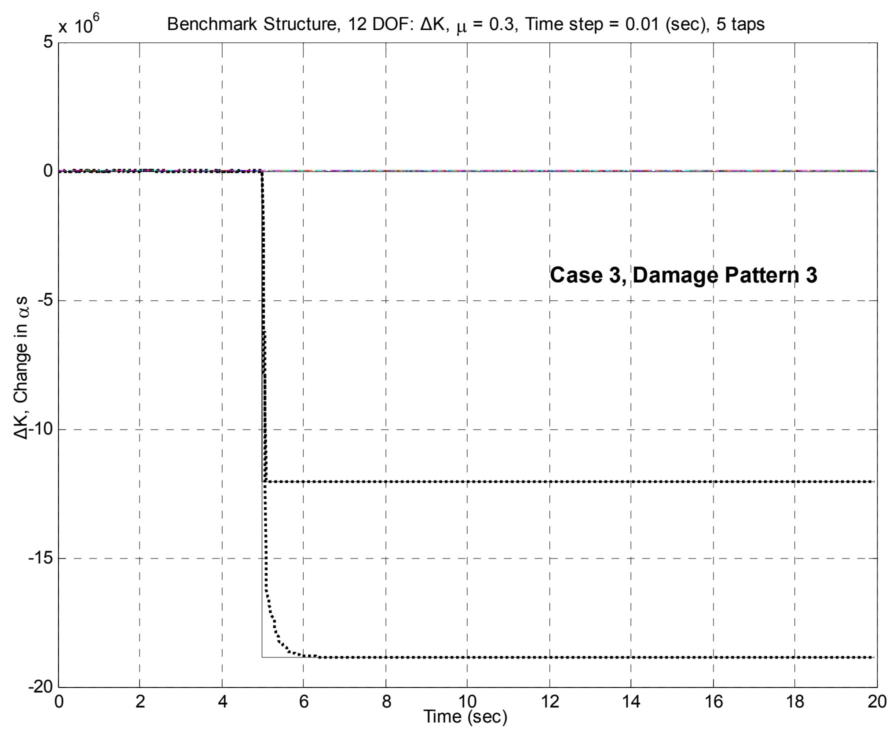


Figure 64 Case 3 and damage pattern 3 using the Two Step Method due to sudden failure

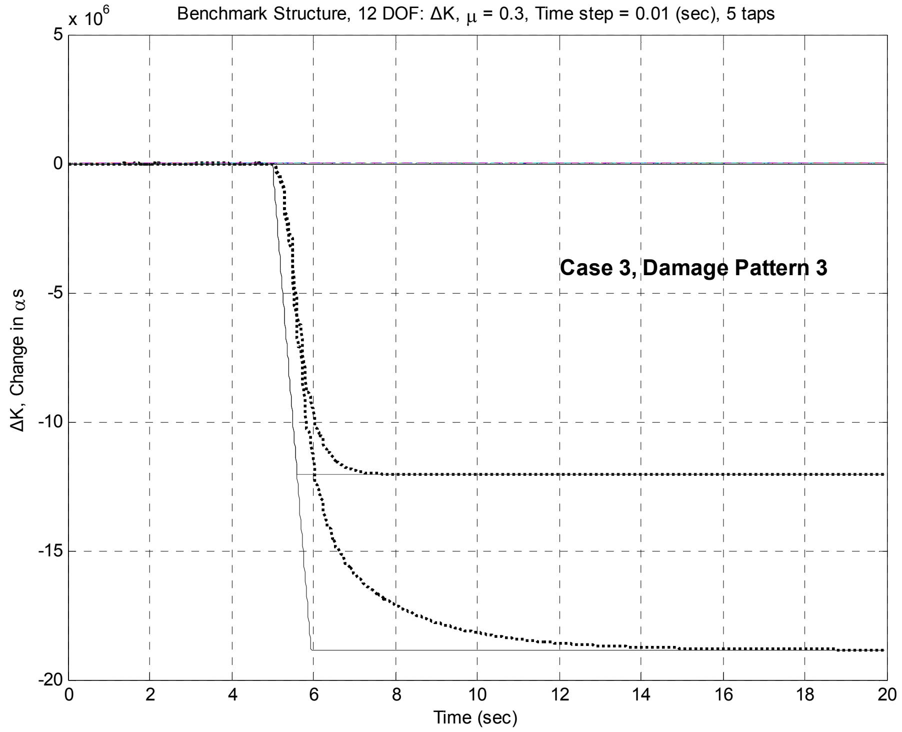


Figure 65 Case 3 and damage pattern 3 using the Two Step Method due to gradual failure

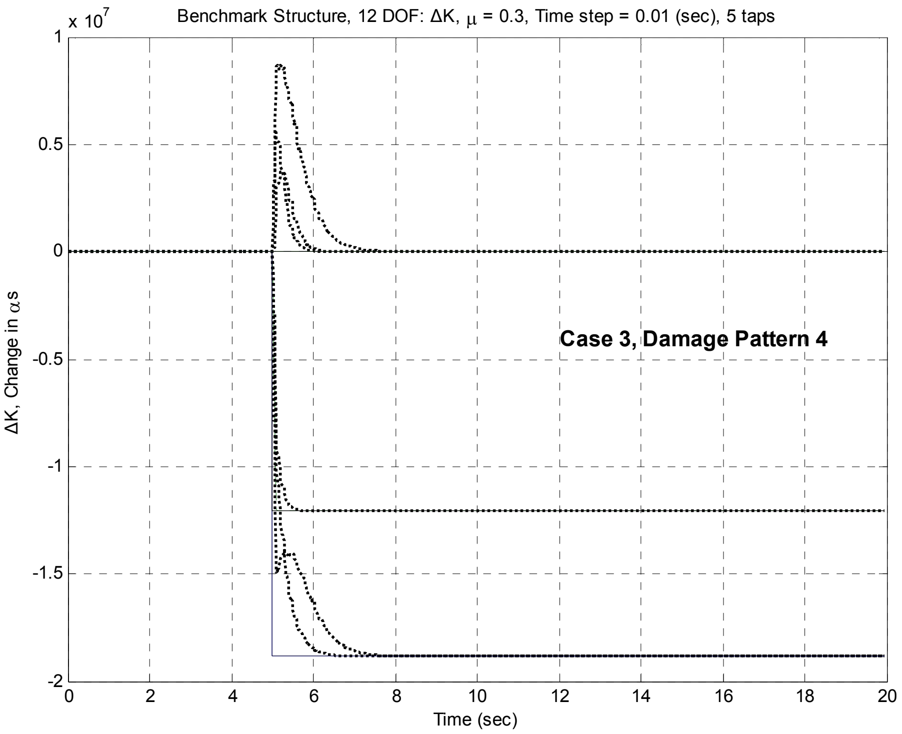


Figure 66 Case 3 and damage pattern 4 using the Two Step Method due to sudden failure

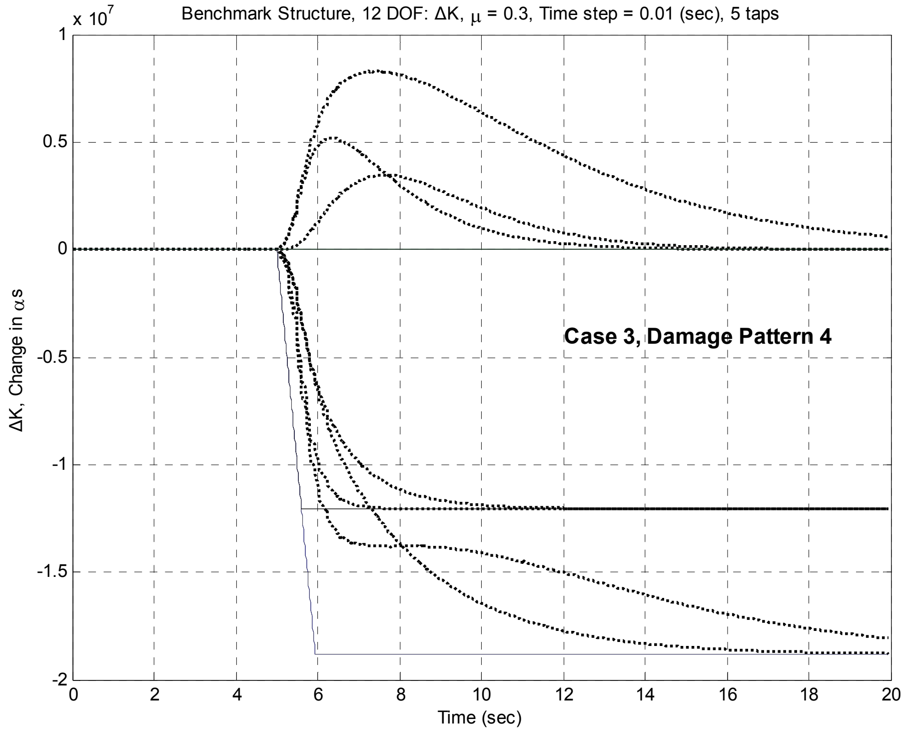


Figure 67 Case 3 and damage pattern 4 using the Two Step Method due to gradual failure

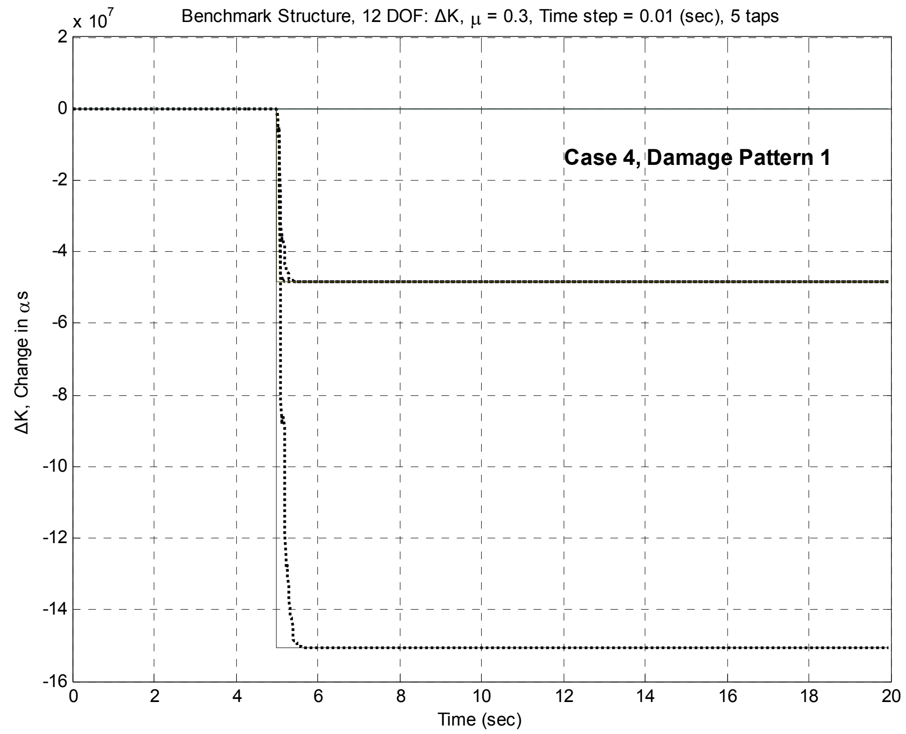


Figure 68 Case 4 and damage pattern 1 using the Two Step Method due to sudden failure

Figure 69 Case 4 and damage pattern 1 using the Two Step Method due to gradual failure

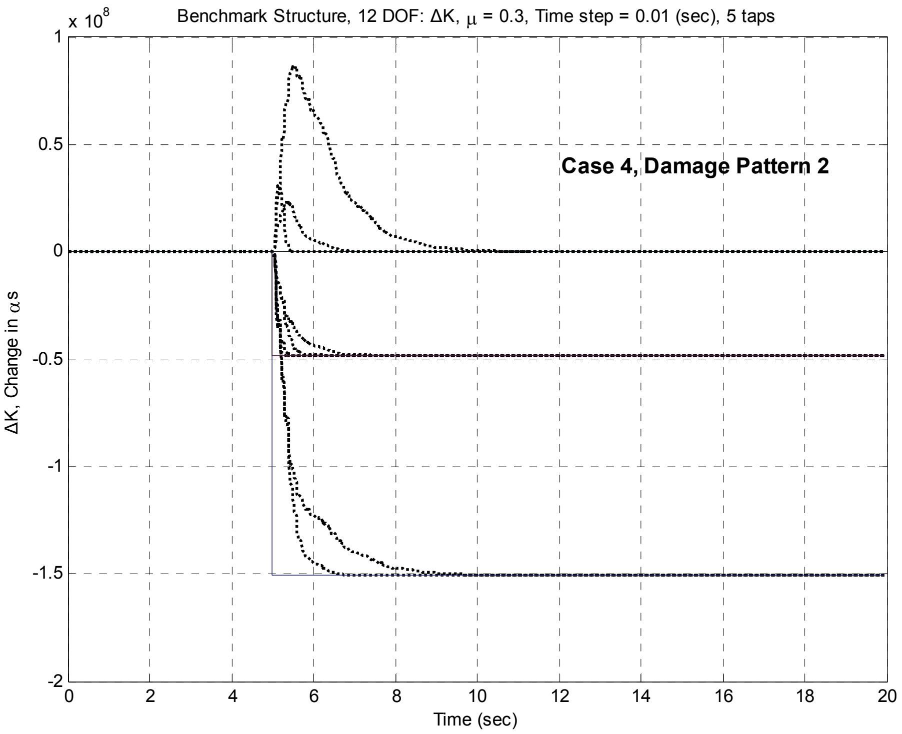


Figure 70 Case 4 and damage pattern 2 using the Two Step Method due to sudden failure

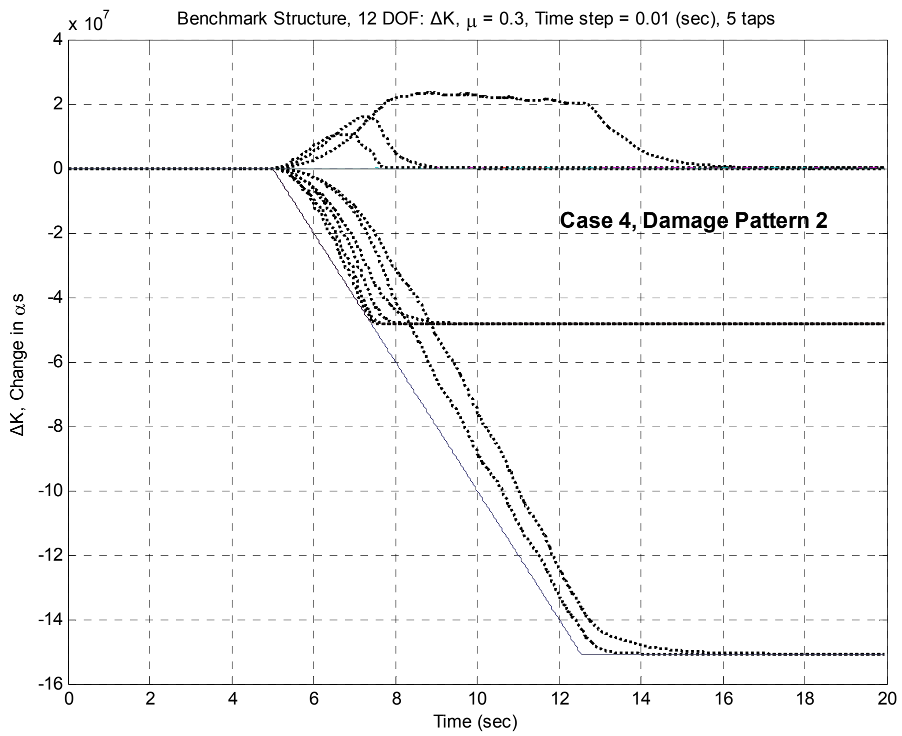


Figure 71 Case 4 and damage pattern 2 using the Two Step Method due to gradual failure

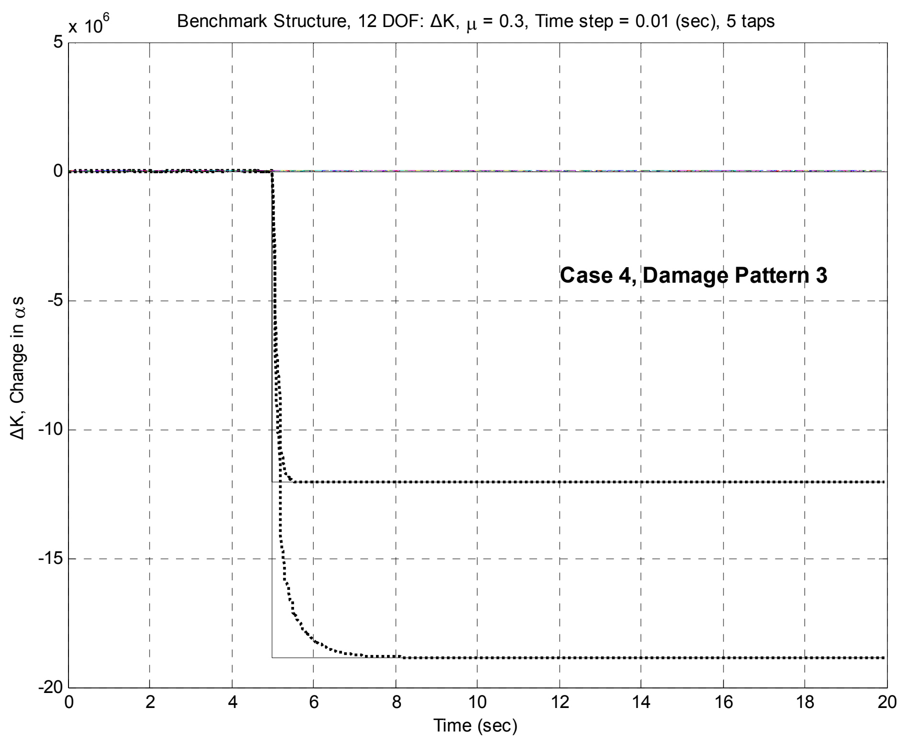


Figure 72 Case 4 and damage pattern 2 using the Two Step Method due to sudden failure

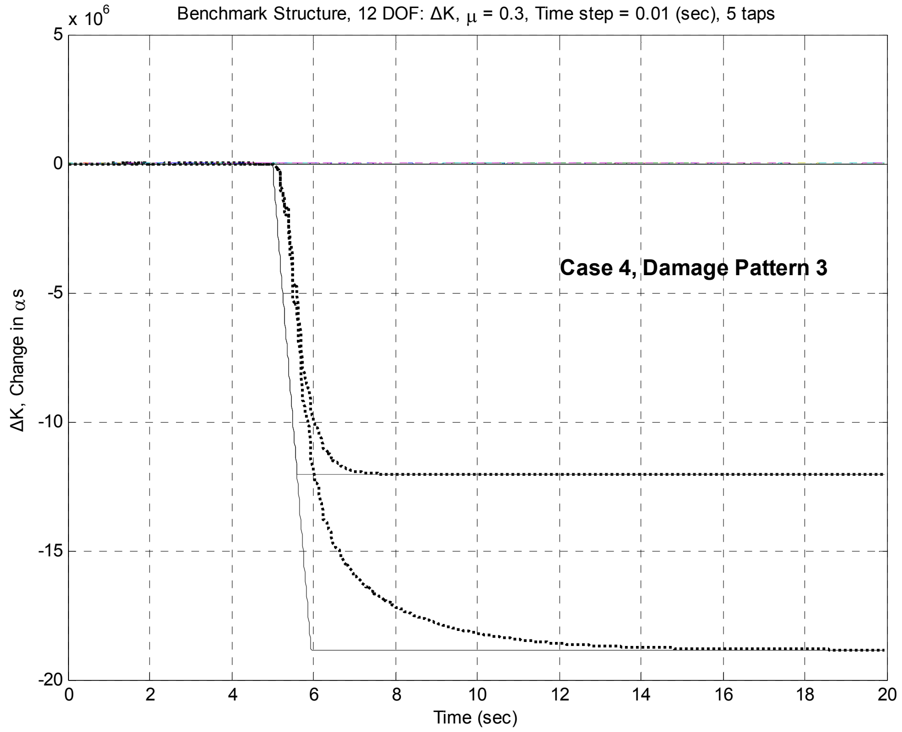


Figure 73 Case 4 and damage pattern 3 using the Two Step Method due to gradual failure

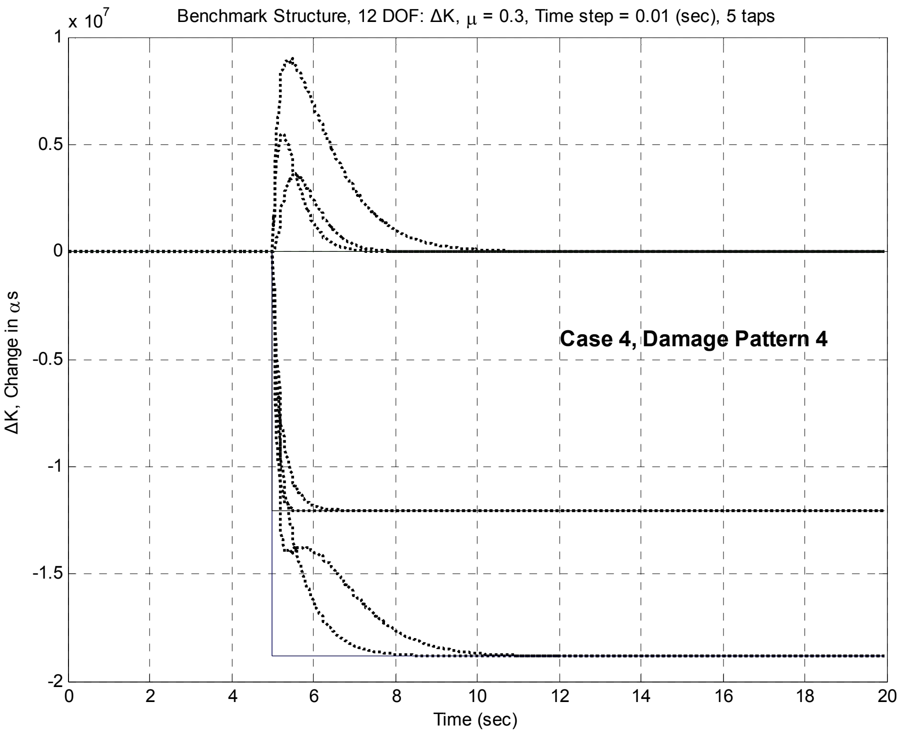


Figure 74 Case 4 and damage pattern 4 using the Two Step Method due to sudden failure

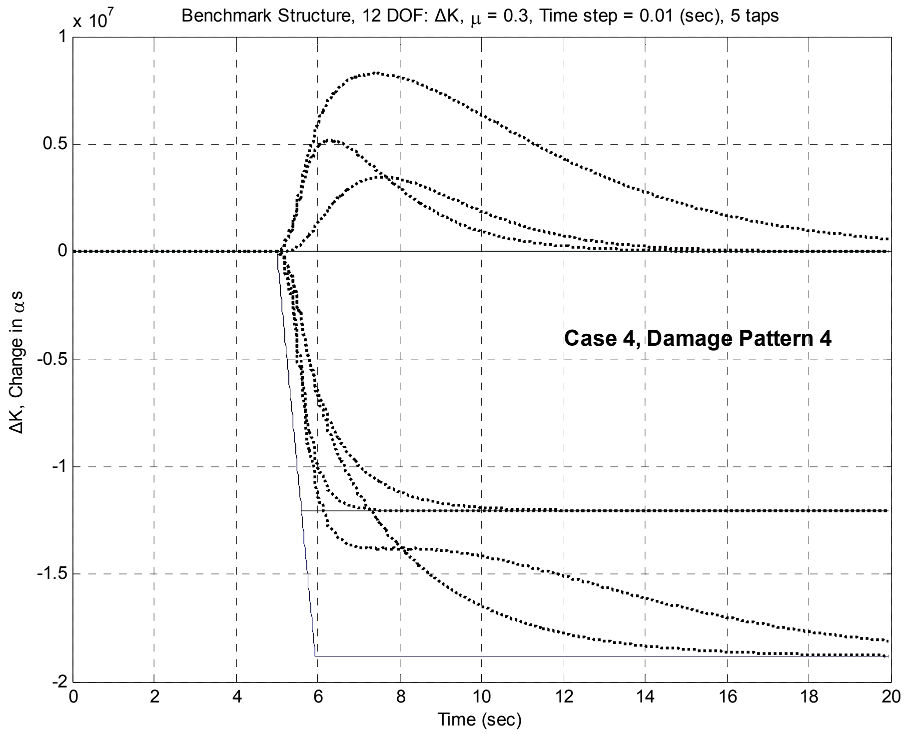


Figure 75 Case 4 and damage pattern 4 using the Two Step Method due to gradual failure

7.4. Natural Frequencies

Table 5 lists the identified natural frequencies for all cases and damage patterns considered here that use a 4 DOF model along with those from the Benchmark Task group simulation and from different published results. For the One Step adaptive LMS based method, which directly identifies changes in stiffness, the resulting net stiffness and modelled mass matrices are used to find the natural frequencies presented. The Kalman method results in Table 5 are from Bernal and Gunes (2000) and the Two-stage results are from Au et al (2000). The identified natural frequencies using the adaptive method presented are well within 1% of the Benchmark and as good as, or better, than the other published results, which means that the resulting modal parameters for the Benchmark structure can be identified accurately with this simple algorithm.

Table 6 shows the identified natural frequencies for case 4 and damage patterns 1 – 4, which uses a 12 DOF model. There are 12 modes and corresponding frequencies, where, '1 y' denotes the first mode in y-direction. Results are reported for all 12 modes for damage patterns 1 – 4 including results from Rodriguez and Barraso (2002) for these cases using a stiffness-mass method. The results are again very close to the Benchmark results, which differences well within 1% for the modes in each direction. The results are also as good as, or better, than the other published results which do not identify all 12 frequencies.

Table 5 Identified natural frequencies (in *Hz*) for cases 1 and 3 damage patterns 1 and 2 using a 4 DOF model

	Case 1							
	No Damage				Damage pattern 1			
Mode	Bench-mark	LMS	Kalman	Two-stage	Bench-mark	LMS	Kalman	Two-stage
1	9.41	9.41	9.41	9.41	6.24	6.24	6.24	6.24
2	25.54	25.54	25.54	25.6	21.53	21.53	21.53	21.6
3	38.66	38.66	38.67	38.9	37.37	37.37	37.58	37.6
4	48.01	48.01	48.01	48.4	47.83	47.83	47.83	48.2

	Case 1				Case 3			
	Damage pattern 2				No Damage			
Mode	Bench-mark	LMS	Kalman	Two-stage	Bench-mark	LMS	Kalman	Two-stage
1	5.82	5.82	5.82	5.83	9.41	9.41	9.41	9.42
2	14.89	14.89	14.89	14.9	25.54	25.54	25.53	25.6
3	36.06	36.06	36.06	36.3	38.66	38.66	38.66	38.9
4	41.35	41.35	41.35	41.6	48.01	48.01	48.09	48.5

	Case 3							
	Damage pattern 1				Damage pattern 2			
Mode	Bench-mark	LMS	Kalman	Two-stage	Bench-mark	LMS	Kalman	Two-stage
1	6.24	6.24	6.23	6.24	5.82	5.82	5.79	5.80
2	21.53	21.53	21.52	21.6	14.89	14.89	14.91	15.0
3	37.37	37.37	37.44	37.7	36.06	36.06	37.44	36.4
4	47.83	47.83	47.94	48.3	41.35	41.35	47.34	41.5

Table 6 Identified natural frequencies (in *Hz*) for case 4 with damage patterns 1 – 4

	Damage pattern 1						
Mode	Benchmark	LMS	Stiffness -Mass	Mode	Benchmark	LMS	Stiffness -Mass
1 y	6.18	6.19	6.11	2 θ	37.93	38.00	-
1 x	9.80	9.80	9.86	4 y	46.81	46.81	47.84
1 θ	11.63	11.72	-	3 x	47.54	47.54	46.75
2 y	21.27	21.27	21.47	4 x	59.63	59.63	59.94
2 x	28.59	28.59	28.49	3 θ	64.67	64.69	-
3 y	36.87	36.87	37.20	4 θ	82.89	82.89	-

	Damage pattern 2						
Mode	Benchmark	LMS	Stiffness -Mass	Mode	Benchmark	LMS	Stiffness -Mass
1 y	5.76	5.76	5.75	2 θ	35.97	35.97	-
1 x	9.39	9.39	9.37	4 y	40.60	40.60	40.83
1 θ	10.90	11.00	-	3 x	46.46	46.46	46.63
2 y	14.78	14.79	14.70	4 x	53.68	53.68	53.29
2 x	24.70	24.70	24.62	3 θ	63.44	63.51	-
3 y	28.22	28.51	35.75	4 θ	71.58	71.69	-

	Damage pattern 3						
Mode	Benchmark	LMS	Stiffness -Mass	Mode	Benchmark	LMS	Stiffness -Mass
1 y	8.79	8.79	8.89	2 θ	43.61	43.61	-
1 x	11.64	11.63	11.67	4 y	47.68	47.68	47.84
1 θ	15.80	15.80	-	3 x	47.96	47.96	47.84
2 y	24.37	24.37	24.50	4 x	59.81	59.81	59.94
2 x	31.66	31.66	31.75	3 θ	66.58	66.58	-
3 y	37.77	37.77	37.44	4 θ	83.18	83.18	-

	Damage pattern 4						
Mode	Benchmark	LMS	Stiffness -Mass	Mode	Benchmark	LMS	Stiffness -Mass
1 y	8.79	8.79	8.77	2 θ	42.91	42.91	-
1 x	11.50	11.50	11.43	4 y	47.68	47.68	47.24
1 θ	15.68	15.68	-	3 x	47.96	47.96	47.84
2 y	24.36	24.36	24.50	4 x	58.18	58.18	58.37
2 x	30.28	30.82	31.03	3 θ	66.56	66.56	-
3 y	37.76	37.76	37.68	4 θ	81.76	81.76	-

Table 7 shows identified natural frequencies for cases 1 and 3 with damage pattern 3 and 4. To the best of the authors' knowledge, none of the published articles have given results for these cases except the SHM Task group. The resulting frequencies are within 1% of the benchmark results.

Table 7 Identified natural frequencies (in Hz) for cases 1 and 3 with damage patterns 3 and 4

	Case 1				Case 3			
	Damage pattern 3		Damage pattern 4		Damage pattern 3		Damage pattern 4	
Mode	Bench-mark	LMS	Bench-mark	LMS	Bench-mark	LMS	Bench-mark	LMS
1 y	8.89	8.89	8.89	8.89	8.89	8.89	8.89	8.89
1 x	11.79	11.79	11.68	11.66	11.79	11.79	11.66	11.66
1 θ	16.01	16.00	15.89	15.89	16.01	16.00	15.89	15.89
2 y	24.60	24.60	24.60	24.60	24.60	24.60	24.59	24.60
2 x	32.01	32.01	31.14	31.14	32.01	32.01	31.14	31.14
3 y	38.24	38.24	38.23	38.23	38.24	38.24	38.23	38.23
2 θ	43.99	43.99	43.21	43.21	43.99	43.99	43.21	43.21
4 y	47.96	47.96	47.96	47.96	47.96	47.96	47.96	47.96
3 x	48.44	48.44	48.41	48.41	48.44	48.44	48.41	48.41
4 x	60.15	60.15	58.62	58.62	60.15	60.15	58.62	58.62
3 θ	67.17	67.17	67.14	67.14	67.17	67.17	67.14	67.14
4 θ	83.58	83.57	82.18	82.18	83.58	83.58	82.18	82.18

8. Discussions and Comparison of Results

8.1. Convergence Time

Table 8 compares the convergence time between the One Step method as in Equation (22), the One Step method with coupling, which uses Equation (25) for weight updating, and the Two Step method. These convergence times are the time taken for α_1 (change in stiffness of the first floor in y-direction) to reach 90 and 95 percent of the actual change from when the damage occurred. The Two Step method was found to have faster convergence than the One Step method, because each element of the vector y_k is modelled individually so the individual filters converge more quickly. In addition, the One Step method is not an exact adaptive LMS filter so its convergence may be limited by the assumptions made. Figure 76 illustrates these results, showing how the convergence rate using the Two Step method is faster for case 4 and damage pattern 1. The convergence times for the One Step method using prior time steps and coupling, as in Equation (26), are approximately two times slower than the One Step method with coupling.

Table 8 Convergence (in seconds) to 90 and 95 % of the actual change of α_1, due to sudden failure

		One Step method with coupling		One Step method without coupling		Two Step method	
Case	Damage Pattern	90 %	95 %	90 %	95 %	90 %	95 %
1	1	0.39	0.59	0.33	0.41	0.20	0.21
	2	0.35	0.36	0.31	0.33	0.21	0.22
	3	3.23	9.11	0.11	0.12	0.20	0.21
	4	5.46	11.53	0.31	0.34	0.21	0.22
3	1	0.39	1.59	0.33	0.41	0.20	0.21
	2	0.35	3.26	0.31	0.33	0.21	0.22
	3	2.99	6.70	0.21	0.32	0.08	0.08
	4	5.14	7.37	0.21	0.32	0.08	0.08
4	1	0.33	0.35	0.21	0.22	0.09	0.01
	2	0.23	0.54	0.13	0.15	0.21	0.23
	3	4.44	12.91	0.29	0.32	0.22	0.28
	4	6.42	13.21	0.29	0.32	0.21	0.28

In comparing convergence times between the One Step methods with and without coupling for the same case, the convergence times are faster when no coupling terms are involved in calculation of the gradient. Particularly, for more complex cases, such as damage patterns 3 and 4, the differences between the two One Step methods are much greater. Using the One

Step method without coupling, the stiffness change in the first floor, α_1 converges within 0.41 seconds for all cases and damage patterns of the Benchmark problem tested. However, with coupling the maximum time is 13.21 seconds. For damage patterns 1 and 2 the convergence times, particularly to 90%, are similar. The similarity at 90% and the greater difference at 95% for damage pattern 1 and 2 shows how the coupling in Equation (25) can impact the effectiveness of the gradients near zero error close to convergence. Finally, damage patterns 3 and 4 are far slower with coupling.

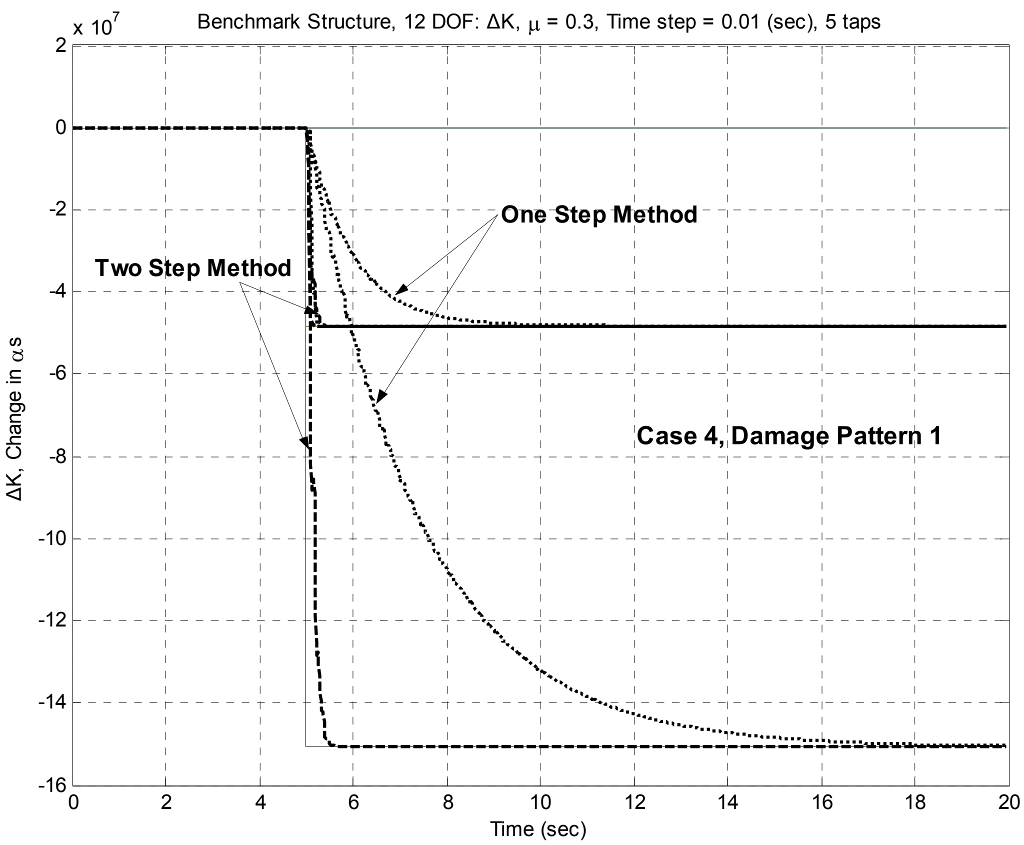


Figure 76 Adaptive LMS approximation using One Step and Two Step method for case 4 and damage pattern 1

Figure 77 shows the convergence using the One Step methods with and without coupling for case 3 and damage pattern 2. As per the results shown in Table 8, it took 0.35 and 3.26 seconds to reach 90% and 95% of the actual change, respectively when the One Step method with coupling is used. Without coupling, it took 0.31 and 0.33 seconds to 90% and 95%, respectively. Note that there is only 0.04 seconds difference for 90% convergence, whereas the difference is about 3 seconds between the two versions of the method to reach 95% of the actual change. This difference occurs due to the involvement of coupling terms in the weight update. With the coupling terms, the algorithm has difficulty finding the final value when there are no more changes due to damage. Figure 78 shows the adaptive LMS based approximation as in Figure 77, but with damage due to gradual failure. As shown in Figure 78, it is clear that the One Step method with coupling had slow convergence just after the value was settled. This result indicates that the stationary assumptions impact the method with coupling much more than the method without it.

Finally, both methods converge to the correct values in a time period suitable for Civil Defence or other immediate needs. In addition, the computational effort required is very small, although without coupling it is even lower. The only major difference is that the method without coupling is more suitable for adaptive control applications given its fast convergence.

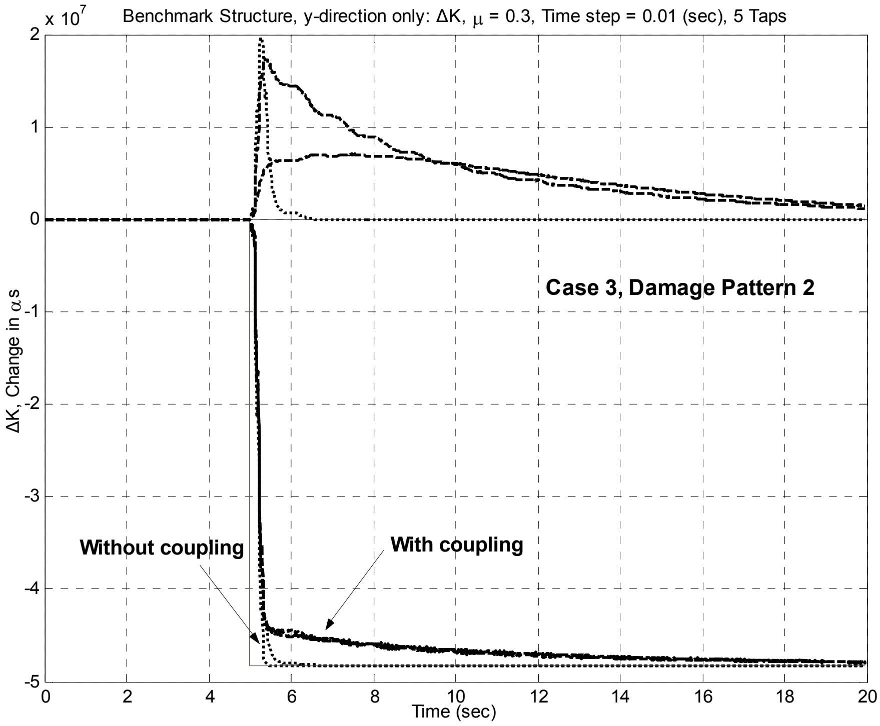


Figure 77 Adaptive LMS approximation using One Step methods with coupling and without coupling for case 3 and damage pattern 2

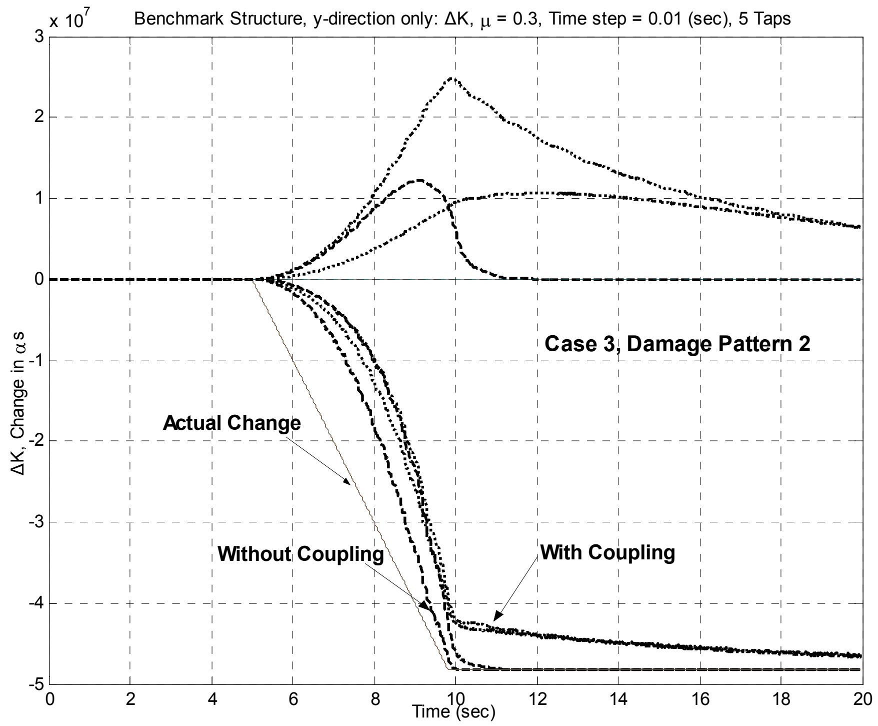


Figure 78 Adaptive LMS approximation using One Step methods with coupling and without coupling for case 3 and damage pattern 2 due to gradual failure

8.2. Sample Rate and Implementation Issues

The convergence rate in adaptive LMS depends on the number of taps used and the LMS parameter, μ. It is also very dependent on the sampling rate. The sampling rate is important because it is directly related to the computational time and capability of the hardware, and the greater number of samples per cycle of the structural response, the faster the potential convergence.

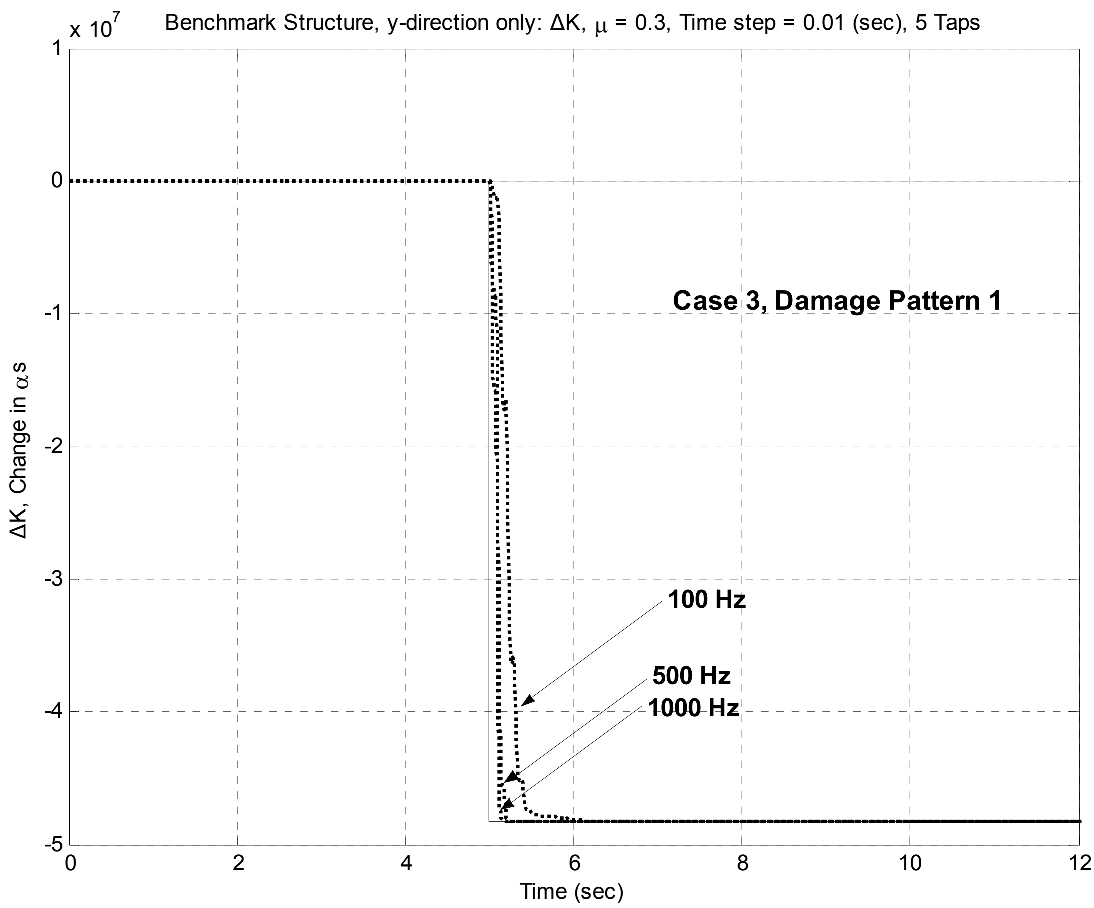


Figure 79 Adaptive LMS approximations for case 3 and damage pattern 1 with sampling rates of 100, 500 and 1000 Hz for sudden failure

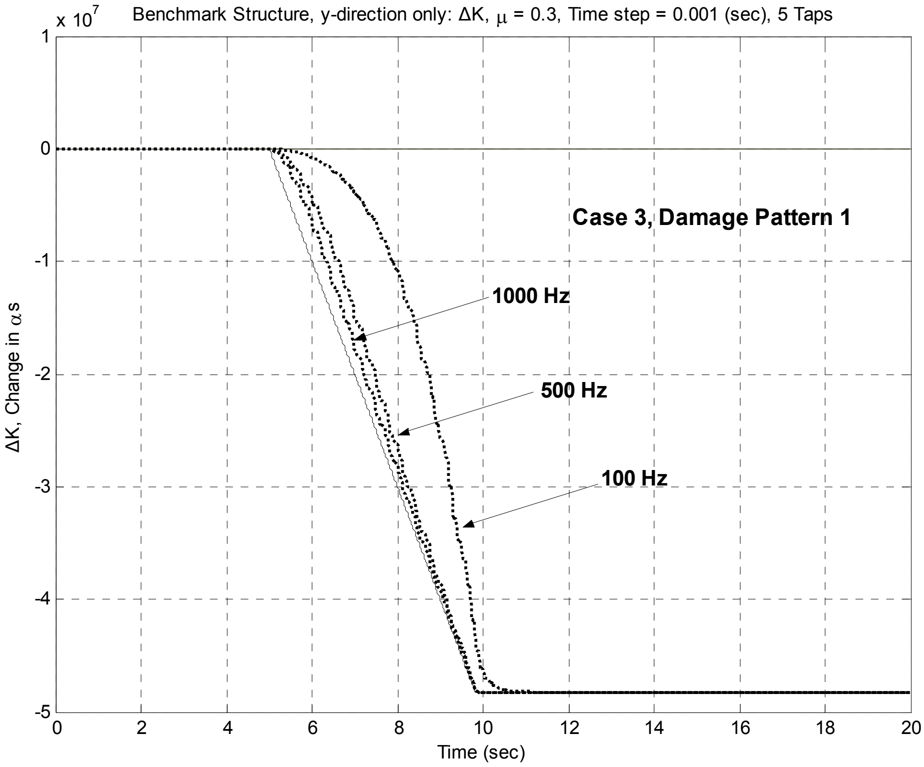


Figure 80 Adaptive LMS approximations for case 3 and damage pattern 1 with sampling rates of 100, 500 and 1000 Hz for gradual failure

Figure 79 and Figure 80 show the damage identification using the adaptive LMS based filtering without coupling for case 3 and damage pattern 1 of the SHM Benchmark problem with different sampling rates. Both Figure 79 and Figure 80 show that the convergence time declines significantly with higher sampling rate. Particularly, in Figure 80, the adaptive LMS identification has almost instantaneous convergence with sampling rates of 500 Hz and 1000 Hz. Note that these sampling rates are possible with modern sensor and data acquisition equipment. The trade-off is that the computations must occur within a far smaller time step, placing a greater requirement on the computational elements.

From an examination of the One Step method, there are, conservatively, 1400 single cycle operations per time step, including memory storage and retrieval for a 4 DOF model. If a sampling rate of 100 Hz is used, then 0.14 MHz (or mega-cycles) of computation is required and 1.4 MHz are required for a sampling rate of 1000 Hz. A 12 DOF model would require approximately 3 times more computational effort. A current Digital Signal Processing (DSP) chip operates at 300 – 1000 MHz. At a single operation per chip clock cycle, and many such chips have up to four operations per cycle, computation of the One Step method is well within this range. The Two Step method would involve approximately ten times more computation due to the matrix solutions required. Therefore, SHM for civil structures using the adaptive LMS filtering based methods as presented could be readily implemented in real-time, even without any significant computational simplifications or parallelization.

9. FUTURE WORK

The developed identification algorithm could be verified experimentally. A simple experimental structure needs to be constructed and tested to verify the results from the computer simulation. A slight modification of the computer model would be required to match the test structure's circumstances.

The algorithm at the moment is used to identify changes in stiffness matrix. The simulation model and method can be modified to include identifying the changes in the structural damping matrix. This task could be done using the same concept as for stiffness matrix identification, however two times the number of variables is required. There is also the potential for misidentifying the stiffness and damping contribution to the linear model error.

The model needs to be tested under various scenarios such as one or more sensors failing during the testing (which is case 6 in the Benchmark problem) or a series of damage occurring in sequence. It would also be worthwhile to simulate the model with the real earthquake data.

Finally, to adapt the method for use in even more practical and realistic situations, an investigation into the validity of the sensors and data acquisition equipment required needs to be done. This task could be one stage of the experimental verification.

10. CONCLUSIONS

This thesis presents SHM methods for civil structures using adaptive Least Mean Square filtering theory. Damage that occurs in the structure can be identified by changes in the stiffness matrix. One Step and Two Step adaptive LMS based methods were developed and tested. All of the 4 and 12 DOF cases of the SHM Task group's Benchmark problems were tested using the proposed methods, and the results show that the adaptive LMS filtering is very effective for identifying damage in real-time.

The different variations are compared and the method without coupling terms in the gradient calculation is seen to converge the fastest. However, the final results for all methods converge to the desired final values. In each case, the changes in stiffness are determined directly and then the modal parameters presented are calculated for comparison. The resulting modal parameters are well within 1% of the IASC-ASCE Benchmark problem results.

The methods presented conservatively require only 0.14 – 1.4 Mega-cycles of computation and can operate on a sample to sample basis without requiring the entire record. Hence, they are all suitable for real-time implementation, and the One Step method without coupling in the gradient calculation has convergence times for the Benchmark problem under 0.41 seconds making it suitable for adaptive control applications. Convergence times for the Two Step method presented are faster, however the computational costs are significantly higher. Finally, the convergence times of the adaptive LMS methods presented improve as sampling rate increases from the 100 Hz of the Benchmark problem to a still practicable value of 1000 Hz. Overall, these methods provide accurate, robust identification of damage with stability, little computational cost, and fast convergence.

Implementation of the activities detailed in the ‘Future Work’ section will serve to further qualify the use of LMS for identifying damage in real-time. All software and models used in this work are also presented in the Appendices.

11. REFERENCES

Allison, A., Abbott, B. (1999) “Parameter Identification using the Hilbert Transform.” *Centre for Biomedical Engineering*, University of Adelaide

Au, S. K., Yuen, K. V. and Beck, J. L. (2000) “Two-Stage System Identification Results for Benchmark Structure” *Proc. of the 14th ASCE Engineering Mechanics Conference*, Austin, Texas, May 21–24.

Bernal, D. and Gunes, B. (2000) “Observer/Kalman and Subspace Identification of the UBC Benchmark Structural Model” *Proc. of the 14th ASCE Engineering Mechanics Conference*, Austin, Texas, May 21–24.

Caicedo, J. M., Dyke, S. J. and Johnson, E. A. (2000) “Health Monitoring Based on Component Transfer Functions” *Proceedings of the 2000 International Conference on Advances in Structural Dynamics*, Hong Kong, December 13-15.

Caicedo, J. M., Marulanda, J., Thomson, P. and Dyke, S. J. (2001) “Monitoring of Bridges to Detect Changes in Structure Health” *Proceedings of the 2001 America Control Conference*, Arlington, Virginia, June 25-27.

Corbin, M., Hera, A. and Hou, Z. (2000) “Locating Damage Regions Using Wavelet Approach” *Proc. of the 14th ASCE Engineering Mechanics Conference*, Austin, Texas, May 21–24.

Doebling, S.W., Farrar, C.R. and Prime, M.B. (1996) “A Summary Review of Vibration-Based Damage Identification Methods” *Los Alamos National Laboratory*, Report.

Doebling, S.W., Farrar, C.R., Prime, M.B., and Shevitz, D.W. (1996a) “Damage Identification and Health Monitoring of Structural and Mechanical Systems from Changes in Their Vibration Characteristics: a Literature Review” *Los Alamos National Laboratory*, Report LA-13070-MS.

Doherty, J. E. (1987) "Non-destructive Evaluation," Chapter 12 in *Handbook on Experimental Mechanics*, A. S. Kobayashi Edt., Society for Experimental Mechanics, Inc.

Dyke, Shirley J., Caicedo, Juan M. and Johnson, Erik A. (2000) "Monitoring of a Benchmark Structure for Damage Identification," *Proc. of the 14th ASCE Engineering Mechanics Conference*, Austin, Texas, May 21–24.

Farrar, C. R. and James III, G. H. (1997) "System Identification from Ambient Vibration Measurements on a Bridge" *Journal of Sound and Vibration*, Vol. 205, No. 1, pp 1-18.

Farrar, C.R. Doebling, S.W., and Nix, D.A. (2000) "Vibration-Based Structural Damage Identification" *Los Alamos National Laboratory*, Report.

Feldman, M. (1997) "Non-linear Free Vibration Identification via the Hilbert Transform" *Journal of Sound and Vibration*, Vol. 208, No. 3, pp 475-489.

Feng M. Q., Liu, C., He, X., and Shinozuka, M. (2000) "Eelectromagnetic Image Reconstruction for Damage Detection" *Journal of Engineering Mechanics*, Vol.126, No. 7, July, pp 725-729.

Franklin G. F., Powell, J. D., Emami-Naeini, A. (2002) *Feedback Control of Dynamic Systems (4th Edition)*, Prentice Hall.

Fugate, M.L., Sohn, H. and Farrar C.R. (2001) "Vibration-Based Damage Detection Using Statistical Process Control" *Journal of Mechanical Systems and Signal Processing*, Vol. 15, No. 4, pp 707-721.

Gattulli, V. and Romeo, F. (2000) "Integrated Procedure for Identification and Control of MDOF Structures" *Journal of Engineering Mechanics,* Vol. 126, No. 7, July, pp 730-737.

Ghanem, R., Bujakov, M., Torikkoshi, K., Itoh, H. Inazuka, T. Hiei, H. and Watanabe, T. (1997) "Adaptive Control of Nonlinear Uncertain Dynamical Systems" *Journal of Engineering Mechanics,* Vol. 123, No.11, pp 1161-1169.

Hahn, S. L. (1996) *Hilbert Transforms in Signal Processing*, Artech House, 1996.

Hoshiya, M. and Saito, E. (1984) "Structural Identification by Extended Kalman Filter" *Journal of Engineering Mechanics*, Vol. 110, No. 12, pp. 1757-1770.

Hou, Z., Noori, M. and Amand, R. (2000) "Wavelet-Based Approach for Structural Damage Detection" *Journal of Engineering Mechanics,* Vol.126, No.7, pp 677-683.

IASC-ASCE SHM Task group website (1999) "http://wusceel.cive.wustl.edu/asce.shm"

Ifeachor, E. C. and Jervis, B. W. (1993), *Digital Signal Processing: A Practical Approach*, Addison-Wesley.

Johnson, E. A., Lam, H. F., Katafygiotis, L. S., and Beck, J. L. (2000) "A Benchmark Problem for Structural Health Monitoring and Damage Detection" *Proc. of the 14th ASCE Engineering Mechanics Conference*, Austin, Texas, May 21–24.

Ljung, L. (1999) *System Identification: Theory for the User (2nd Edition)*, Prentice Hall.

Loh, C.-H., Lin, C.-Y., and Huang, C.-C. (2000) "Time Domain Identification of Frames under Earthquake Loadings" *Journal of Engineering Mechanics*, Vol.126, No.7, pp 693-703.

Lus, H. and Betti, R. (2000) "Damage Identification in Linear Structural Systems" *Proc. of the 14th ASCE Engineering Mechanics Conference*, Austin, Texas, May 21–24.

Norton, J. P. (1986) *An Introduction to Identification*, Academic Press, 1986.

Rodriguez, R. and Barroso, L. R. (2002) "Stiffness-Mass Ratio Method for Baseline Determination and Damage Assessment of a Benchmark Structure" *American Control Conference*, Anchorage, Alaska, May 8-10.

Rorabaugh, C. B. (1998) *DSP Primer*, McGraw-Hill.

Rytter, A., (1993) "Vibration Based Inspection of Civil Engineering Structures" Ph.D. Dissertation, Department of Building Technology and Structural Engineering, Aalborg University, Denmark.

Sato, T. and Qi, K. (1998) "Adaptive H_∞ Filter: Its Application to Structural Identification" *Journal of Engineering Mechanics,* Vol.124, No.11, pp 1233-1240.

Shi, T., Jones, N.P. and Ellis J.H. (2000) "Simultaneous Equation of System and Input Parameters from Output Measurements" *Journal of Engineering Mechanics*, Vol. 126, No. 7, pp 746-753.

Sohn, H., and Law, K. H. (2001) "Damage Diagnosis Using Experimental Ritz Vectors" *Journal of Engineering Mechanics,* Vol. 127, No. 11, pp 1184-1193.

Sohn, H., Czarnecki, J.A. and Farrar, C.R. (2000) "Structural Health Monitoring Using Statistical Process Control" *Journal of Structural Engineering*, Vol.126, No.11, pp 1356-1363.

Solo, V. and Kong, X. (1995) *Adaptive Signal Processing Algorithms*, Prentice Hall.

Vanik, M. W., Beck, J. L. and Au, S. K. (2000) "A Bayesian Probabilistic Approach to Structural Health Monitoring" *Journal of Engineering Mechanics,* Vol. 126, No. 7, pp 738-745.

Vestroni, F. and Capecchi, D. (2000) "Damage Detection in Beam Structures Based on Frequency Measurements" *Journal of Engineering Mechanics,* Vol. 126, No. 7, pp 761-768.

Vu-Manh, H., Abe, M., Fuhino, Y. and Kaito, K. (2001) "The Eigensystem Realization Algorithm for Ambient Vibration Measurement Using Laser Doppler Vibrometers" Proc. of the American Control Conference, Arlington, VA, Jun 25-27.

APPENDIX

A1. Matlab Codes: 4 DOF model

```
%*********************************************************************************
% FILE NAME: LMS_BENCHMARK_NEW.m
%       calculates delta_K - changes in stiffness matrix for four DOF system
%       EOM for baseline: Mv" + Cv' + Kv = F, and
%           for damaged:  Mv" + Cv' + (K+ΔK)v = F.
%       A 4 DOF model of SHM Benchmark structure (y-direction only)
%       Cases 1 & 3 with damage pattern 1 & 2 only
%       associated files: EOM_benchmark.mdl, s_delta_benchmark.m
%       Created by Leo Hwang on 13th Mar. 2003
%       Last Modified by Leo Hwang on 16th Jul. 2003
%       Department of Mechanical Engineering, Unveristy of Canterbury, New Zealand
%*********************************************************************************

clear all, clc, %close all

% ---------------------------------------------------------------------------------
% define simulation parameters and model characteristics
% ---------------------------------------------------------------------------------

% simulation parameters
t_final = 20;                   % total simulation time
t_step  = 0.01;                 % sample time step
t = 0:t_step:t_final;           % time vector
time_no = t_final/t_step + 1;   % number of time sample
t_change = 5;                   % time when delta_k starts changing

% choose case interested (only for case 1 & 3)
disp('Case 0 - Symmetric, 12 DOF, ground excitation (not in benchmark problem)')
disp('Case 1 - Symmetric, 12 DOF, load at all stories')
disp('Case 3 - Sysmetric, 12 DOF, load at roof')
case_no = input('** Enter a Case number [0, 1, or 3] = ');

% model characteristics
dof = 4;                        % degree of freedom

m = [3452.4 2652.4 2652.4 1809.9];          % mass of each story
c = 1e2 * [0.2 0.2 0.2 0.2];                % damping coefficients
k = 1e+8 * [0.6790 0.6790 0.6790 0.6790];   % stiffness coefficients

% construct mass matrix
M = diag(m);
inv_M = inv(M);

% construct damping matrix
for i = 2:dof-1,
    c_diag(i) = c(i) + c(i+1);
    C(i,:) = [zeros(1,i-2) -c(i) c_diag(i) -c(i) zeros(1,dof-1-i)];
end
C(1,:) = [c(1)+c(2) -c(1) zeros(1,dof-2)];
```

```
C(dof,:) = [zeros(1,dof-2) -c(dof) c(dof)];

% construct stiffness matrix
for i = 2:dof-1,
    k_diag(i) = k(i) + k(i+1);
    K(i,:) = [zeros(1,i-2) -k(i) k_diag(i) -k(i) zeros(1,dof-1-i)];
end
K(1,:) = [k(1)+k(2) -k(1) zeros(1,dof-2)];
K(dof,:) = [zeros(1,dof-2) -k(dof) k(dof)];


% -----------------------------------------------------------------------------------
% select damage pattern and determine △K matrix for each case
% -----------------------------------------------------------------------------------

% select damage patterns of Benchmark problem (for pattern 1 & 2)
disp('  ')
disp('Damage pattern 0 - undamaged (baseline structure)')
disp('Damage pattern 1 - all braces of 1st story are broken')
disp('Damage pattern 2 - all braces of 1st and 3rd story are broken')
damage_no = input('** Enter a number for Damage pattern [0, 1 or 2] = ');

% submatirces of delta_K, each alpha represents change in stiffness for each floor
% △K = α1*K_in1 + α2*K_in2 + α3*K_in3 + α4*K_in4
% (α1 = △k1, α2 = △k2, α3 = △k3, α4 = △k4)
K_in(:,:,1)=[1 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0];
K_in(:,:,2)=[1 -1 0 0;-1 1 0 0 ;0 0 0 0;0 0 0 0];
K_in(:,:,3)=[0 0 0 0;0 1 -1 0;0 -1 1 0;0 0 0 0];
K_in(:,:,4)=[0 0 0 0;0 0 0 0;0 0 1 -1;0 0 -1 1];


% -----------------------------------------------------------------------------------
% determine input force
% -----------------------------------------------------------------------------------

% magnitude of wind excitation and ground force [N] - sine wave
freq = 30;      % frequency of a sine wave [rad/sec]

% define load applied according to the selected case
if case_no == 0,                % ground excitation
    wind_f = [0 0 0 0];
    ground_f = 500;

elseif case_no == 1,            % case 1 - load at all story
    wind_f = 1e6*[1 1 1 1];
    ground_f = 0;

elseif case_no == 3,            % case 3 - roof excitation
    wind_f = 8e6*[0 0 0 1];
    ground_f = 0;

end


% -----------------------------------------------------------------------------------
% start simulation for four story buliding (4 DOF model)
```

```
% -------------------------------------------------------------------------------------

sim('EOM_benchmark',[0:t_step:t_final]);


% -------------------------------------------------------------------------------------
% determine K_damanged = K + △K analytically
% just for checking with simulation result
K_delta = alphas(time_no,1)*K_in(:,:,1) + alphas(time_no,2)*K_in(:,:,2) + ...
    alphas(time_no,3)*K_in(:,:,3) + alphas(time_no,4)*K_in(:,:,4);
K_damaged = K + K_delta;


% -------------------------------------------------------------------------------------
% compute difference (error) between the original and damaged
acc = acc'; vel = vel'; dis = dis'; F_in = F_in';
y_out = F_in - M*acc - C*vel - K*dis;

% to verify the simulation data (y_zero should be zero)
y_zero = F_in - M*acc - C*vel - K_damaged*dis;


% -------------------------------------------------------------------------------------
% Adaptive LMS filtering, for n weights, m taps
% -------------------------------------------------------------------------------------

filter_no = dof;        % number of signal (filter) = number of DOF
tap_no = 5;             % number of taps for each weight
mu = 0.3;               % LMS factor: controls stability and rate of convergence
w(dof,tap_no,tap_no+time_no) = 0;  % initialise weight matrix

% redefine x and y
% add m-1 zero vectors for prior time step calculation
x_temp = dis;
y_temp = y_out;

x = zeros(dof,tap_no+time_no);
y = zeros(dof,tap_no+time_no);

x(:,tap_no+1:tap_no+time_no) = x_temp;
y(:,tap_no+1:tap_no+time_no) = y_temp;

% adaptive LMS weight updating, One Step method
for i1 = tap_no+1:time_no,

    n_temp = zeros(dof,dof);
    sum_n = 0;
    Q = 0;
    G = zeros(dof,tap_no);

    for j = 1:tap_no,
        for i2 = 1:dof

            % i) exact w/ prior time step disps
            %n_temp = w(i2,j,i1)*K_in(:,:,i2)*x(:,i1-j+1);

            % ii) w/o prior time step disps (main method)
```

```
            n_temp = w(i2,j,i1)*K_in(:,:,i2)*x(:,i1);

            sum_n = sum_n + n_temp;

        end         % for i
    end             % for j

    Q = sum_n;

    e(:,i1) = y(:,i1) - Q;

    for i3 = 1:dof,
        for j2 = 1:tap_no,

            % i) exact with prior time displacements
            %G(i3,j2) = e(:,i1)'*K_in(:,:,i3)*x(:,i1-j2+1);

            % ii) exact, current disps only
            %G(i3,j2) = e(:,i1)'*K_in(:,:,i3)*x(:,i1);

            % iii) w/o coupling and prior disps (main method)
            G(i3,j2) = e(i3,i1)*x(i3,i1-j2+1);

            % iv) w/o coupling w/o prior disps
            %G(i3,j2) = e(i3,i1)*x(i3,i1);
        end
    end

    G = 2*mu*G;

    w(:,:,i1+1) = w(:,:,i1) + G;

end

% sum up for m elements of a weight
w_temp = 0;
for j = 1:tap_no,
    w_temp = w_temp + w(:,j,:);
end

% redefine index for plotting
for i = 1:time_no-tap_no+1,
    w1(:,:,i) = w_temp(:,:,i+tap_no-1);
end


% -------------------------------------------------------------------------------------
% plot results (LMS and actual alphas vs time)
% -------------------------------------------------------------------------------------
%t_axis = 0:t_step:t_final;
t_axis = 0:t_step:t_final-t_step*(tap_no-1);
figure,
%plot(t_axis,w(:,1:time_no),':',t_axis,alphas)
plot(t_axis,w1(:,1:time_no-tap_no+1),':',t_axis,alphas(1:time_no - tap_no+1,:))
title(['Benchmark Structure, y-direction only: ΔK, \mu = ' num2str(mu(1)), ...
        ', Time step = ' num2str(t_step) ' (sec), ' num2str(tap_no) ' Taps']), grid
ylabel('ΔK, Change in \alphas'), xlabel('Time (sec)')
```

```
text(12,-1.5e7,['Case ' num2str(case_no) ', Damage Pattern ' num2str(damage_no)], ...
    'fontsize',12,'fontweight','bold')


%************************************************************************************
%*************************** End of LMS_benchmark_new.m ****************************
%************************************************************************************
```

A2. Matlab Codes: S-function

```
%*************************************************************************************
% FILE NAME: S_DELTA_BENCHMARK.M
%       changes in stiffness coefficent as the simulation runs
%       this function called by EOM_benchmark.mdl
%       Created by Leo Hwang
%       Department of Mechanical Engineering, Unveristy of Canterbury, New Zealand
%*************************************************************************************

function [sys,x0,str,ts] = s_delta_benchmark(t,x,u,flag,t_change,damage_no)

switch flag,
    % Initialization %
    case 0,
        [sys,x0,str,ts]=mdlInitializeSizes;
        % Derivatives %
    case 1,
        sys=mdlDerivatives(t,x,u,t_change);
        % Update %
    case 2,
        sys=mdlUpdate(t,x,u,t_change);
        % Outputs %
    case 3,
        sys=mdlOutputs(t,x,u,t_change,damage_no);
        % GetTimeOfNextVarHit %
    case 4,
        sys=mdlGetTimeOfNextVarHit(t,x,u,t_change);
        % Terminate %
    case 9,
        sys=mdlTerminate(t,x,u,t_change);
        % Unexpected flags %
    otherwise
        error(['Unhandled flag = ',num2str(flag)]);
end

%==============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%==============================================================================
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 4;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;   % at least one sample time is needed
sys = simsizes(sizes);
% initialize the initial conditions
x0  = [];
% str is always an empty matrix
str = [];
% initialize the array of sample times
```

```
ts  = [0 0];
% end mdlInitializeSizes

%=============================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================================
function sys=mdlDerivatives(t,x,u,t_change)
sys = [];
% end mdlDerivatives

%=============================================================================
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time step
% requirements.
%=============================================================================
function sys=mdlUpdate(t,x,u,t_change)
sys = [];
% end mdlUpdate

%=============================================================================
% mdlOutputs
% Return the block outputs.
%=============================================================================
function sys=mdlOutputs(t,x,u,t_change,damage_no)

% changing in alphas
if damage_no == 0,                  % undamage
    alpha = 1e8 * [0 0 0 0];
elseif damage_no == 1,              % damage pattern 1
    alpha = 1e8 * [-0.4822 0 0 0];
else                                % damage pattern 2
    alpha = 1e8 * [-0.4822 0 -0.4822 0];    % for new K_in's
end

% damage profile
% i) sudden failure (step change)

% for i = 1:4,
%     if u(1) < t_change,
%         sys(i) = 0;
%         %sys(i) = 0 + 1e6*randn;     % non-linear
%     else
%         sys(i) = alpha(i);
%         %sys(i) = alpha(i) + 1e6*randn;    % non-linear
%     end
% end


% ii) gradual failure (ramp change)

grad = 1e7 * [-1 -1 -1 -1];
for i = 1:4
    if u <= t_change,
        sys(i) = 0;
    elseif u <= t_change + alpha(i)/grad(i),
        sys(i) = grad(i)*u - grad(i)*t_change;
```

```
    else
        sys(i) = alpha(i);
    end
end

%end mdlOutputs

%=============================================================================
% mdlGetTimeOfNextVarHit
% Return the time of the next hit for this block.  Note that the result is
% absolute time.  Note that this function is only used when you specify a
% variable discrete-time sample time [-2 0] in the sample time array in
% mdlInitializeSizes.
%=============================================================================
function sys=mdlGetTimeOfNextVarHit(t,x,u,t_change)
sys = [];
% end mdlGetTimeOfNextVarHit

%=============================================================================
% mdlTerminate
% Perform any end of simulation tasks.
%=============================================================================
function sys=mdlTerminate(t,x,u,t_change)
sys = [];
% end mdlTerminate

%*************************************************************************************
%*************************** End of s_delta_benchmark.m ****************************
%*************************************************************************************
```

A3. Simulink Model: 4 DOF model

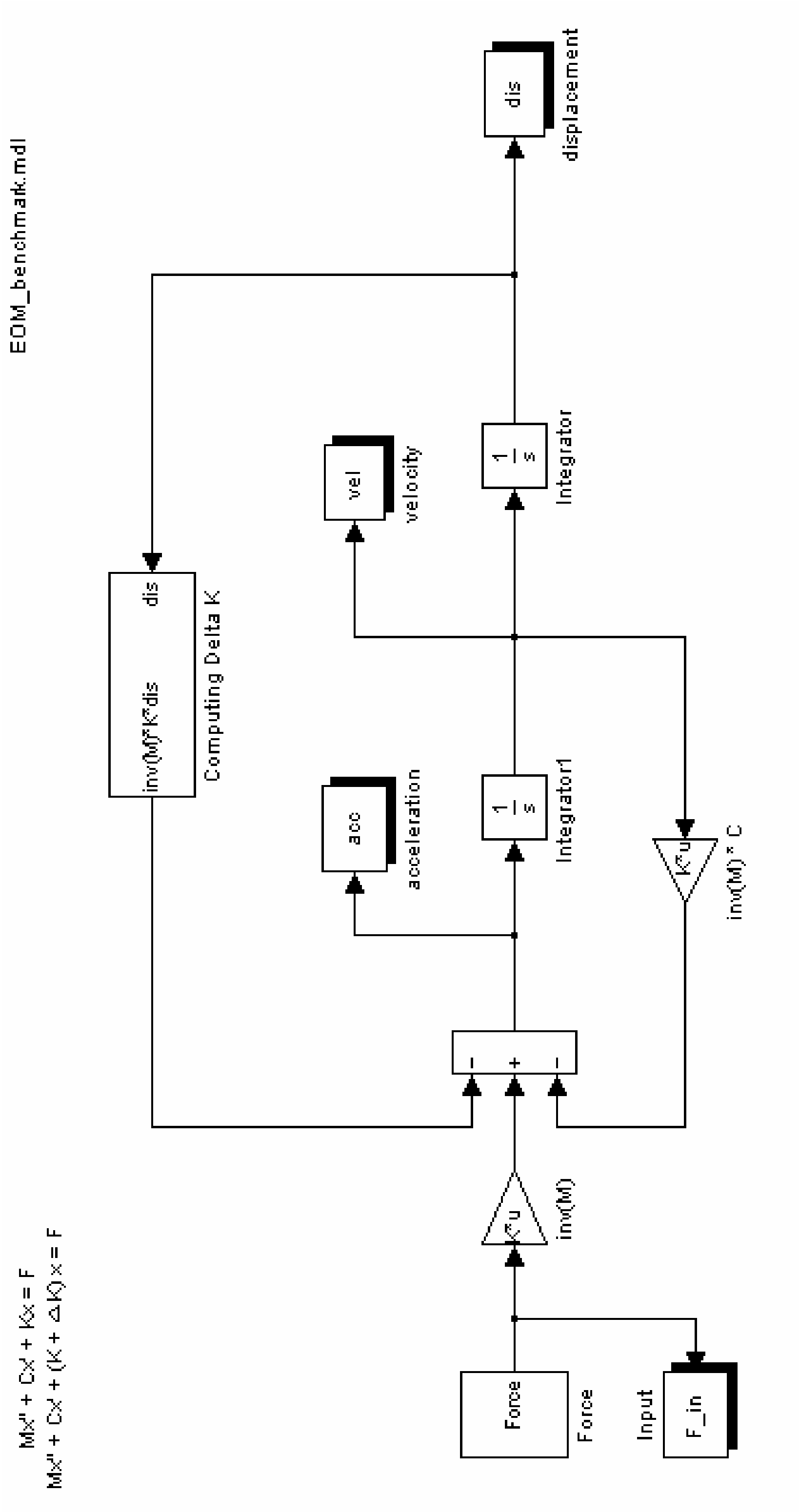


Figure 81 Simulink representation for 4 DOF model of Benchmark structure: `EOM_benchmark.mdl`

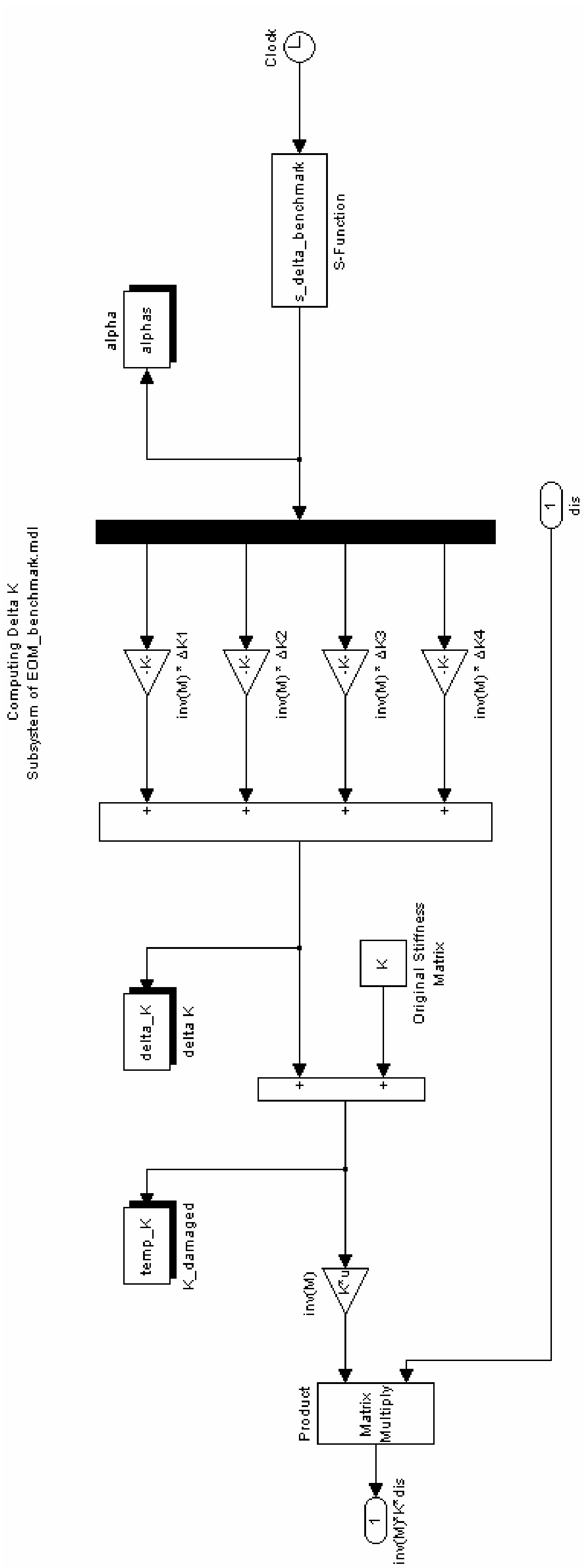


Figure 82 `Computing Delata K`: Subsystem of `EOM_benchmark.mdl`, creating damage by changing α values

A4. Matlab Codes: 12 DOF model

```
%**************************************************************************************
% FILE NAME: LMS_BENCHMARK12_NEW.m
%       calculates delta_K - changes in stiffness matrix for four DOF system
%       EOM for baseline: Mv" + Cv' + Kv = F, and
%           for damaged:  Mv" + Cv' + (K+ΔK)v = F.
%       12 DOF (3 DOF per floor) model
%       Cases 1, 3 and 4 with damage pattern 1 - 4
%       associated files: EOM_benchmark12.mdl, s_delta_benchmark12.m
%       Created by Leo Hwang on 29 April 2003
%       Last Modified by Leo Hwang on 20th July 2003
%       Department of Mechanical Engineering, Unveristy of Canterbury, New Zealand
%**************************************************************************************

clear all, clc, %close all

% ------------------------------------------------------------------------------------
% define simulation parameters and model characteristics
% ------------------------------------------------------------------------------------

% simulation paramters
t_final = 20;                  % total simulation time
t_step  = 0.01;                % sample time step
t = 0:t_step:t_final;          % time vector
time_no = t_final/t_step + 1;  % number of time sample
t_change = 5;                  % time when delta_k starts changing

% choose case interested (all 12 DOF cases: 1, 3 and 4)
disp('Case 0 - Symmetric, 12 DOF, ground excitation (not in benchmark problem)')
disp('Case 1 - Symmetric, 12 DOF, load at all stories')
disp('Case 3 - Sysmetric, 12 DOF, load at roof')
disp('Case 4 - Asymmetric, 12 DOF, load at roof')
case_no = input('** Enter a Case number [0, 1, 3, or 4] = ');

% model characteristics
dof = 12;                  % degree of freedom

% mass, damping and stiffness coefficients per dof.
m = [3452.4 3452.4 3819.4 2652.4 2652.4 2986.1 2652.4 2652.4 2986.1 1809.9 ...
        1809.9 2056.9];
c = 1e2 * [0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2];
k = 1e+8 * [1.066 0.6790 2.3202 1.066 0.6790 2.3202 1.066 0.6790 2.3202 ...
        1.066 0.6790 2.3202];

% construct mass matrix
% for case 1 & 3 - symmetric
M = diag(m);

% for case 4 - asymmetric
if case_no == 4;
    M(10:12,10:12) = [1959.9 0 93.8;0 1959.9 -93.8;93.8 -93.8 2213.1];
end

inv_M = inv(M);
```

```
% construct damping matrix of baseline structure
C = [
    c(1)+c(4)    0    0  -c(4)    0    0    0    0    0    0    0    0
    0   c(2)+c(5)    0    0  -c(5)    0    0    0    0    0    0    0
    0    0   c(3)+c(6)    0    0  -c(6)    0    0    0    0    0    0
    -c(4)    0    0   c(4)+c(7)    0    0  -c(7)    0    0    0    0    0
    0  -c(5)    0    0   c(5)+c(8)    0    0  -c(8)    0    0    0    0
    0    0  -c(6)    0    0   c(6)+c(9)    0    0  -c(9)    0    0    0
    0    0    0  -c(7)    0    0   c(7)+c(10)    0    0  -c(10)    0    0
    0    0    0    0  -c(8)    0    0   c(8)+c(11)    0    0  -c(11)    0
    0    0    0    0    0  -c(9)    0    0   c(9)+c(12)    0    0  -c(12)
    0    0    0    0    0    0  -c(10)    0    0   c(10)    0    0
    0    0    0    0    0    0    0  -c(11)    0    0   c(11)    0
    0    0    0    0    0    0    0    0  -c(12)    0    0   c(12)
];

% construct stiffness matrix of baseline structure - fixed
K = [
    k(1)+k(4)    0    0  -k(4)    0    0    0    0    0    0    0    0
    0   k(2)+k(5)    0    0  -k(5)    0    0    0    0    0    0    0
    0    0   k(3)+k(6)    0    0  -k(6)    0    0    0    0    0    0
    -k(4)    0    0   k(4)+k(7)    0    0  -k(7)    0    0    0    0    0
    0  -k(5)    0    0   k(5)+k(8)    0    0  -k(8)    0    0    0    0
    0    0  -k(6)    0    0   k(6)+k(9)    0    0  -k(9)    0    0    0
    0    0    0  -k(7)    0    0   k(7)+k(10)    0    0  -k(10)    0    0
    0    0    0    0  -k(8)    0    0   k(8)+k(11)    0    0  -k(11)    0
    0    0    0    0    0  -k(9)    0    0   k(9)+k(12)    0    0  -k(12)
    0    0    0    0    0    0  -k(10)    0    0   k(10)    0    0
    0    0    0    0    0    0    0  -k(11)    0    0   k(11)    0
    0    0    0    0    0    0    0    0  -k(12)    0    0   k(12)
];

% ---------------------------------------------------------------------------------------
% select damage pattern and determine △K matrix for each case
% ---------------------------------------------------------------------------------------

% select damage patterns of Benchmark problem (for pattern 1 - 4)
disp('  ')
disp('Damage pattern 0 - undamaged (baseline structure)')
disp('Damage pattern 1 - all braces of 1st story are broken')
disp('Damage pattern 2 - all braces of 1st and 3rd story are broken')
disp('Damage pattern 3 - one brace of 1st story is broken')
disp('Damage pattern 4 - one brace of 1st and 3rd story are broken')
damage_no = input('** Enter a number for Damage pattern [0, 1, 2, 3 or 4] = ');


% new delta K, each alpha represents change in stiffness for each floor
% △K = α1*K_in1 + α2*K_in2 + α3*K_in3 + α4*K_in4 + ...
% (α1 = △k1, α2 = △k2, α3 = △k3, α4 = △k4, ...)
if damage_no == 3,
    k23 = 0.1507/0.1205;          % k23 - entry at row 2 & column 3
    k32 = 0.1507/0.1205;
    k46 = 0; k64 = 0; k49 = 0; k94 = 0; k67 = 0; k76 = 0; k79 = 0; k97 = 0;
elseif damage_no == 4,
    k23 = 0.1507/0.1205;    k32 = 0.1507/0.1205;
    k46 = 0.1507/0.1205;    k64 = 0.1507/0.1205;
```

```
    k49 = -0.1507/0.1205;   k94 = -0.1507/0.1205;
    k67 = -0.1507/0.1205;   k76 = -0.1507/0.1205;
    k79 = 0.1507/0.1205;    k97 = 0.1507/0.1205;
else
    k23 = 0; k32 = 0; k46 = 0; k64 = 0; k49 = 0;
    k94 = 0; k67 = 0; k76 = 0; k79 = 0; k97 = 0;
end

K_in(:,:,1) = ...
    [1 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0];
% K_in(:,:,2) = ...
%     [0 0 0 0 0 0 0 0 0 0 0 0;0 1 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
%         0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
%         0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
%         0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0];
K_in(:,:,2) = ...
    [0 0 0 0 0 0 0 0 0 0 0 0;0 1 k23 0 0 0 0 0 0 0 0 0;0 k32 0 0 0 0 0 0 0 0 0 0;...
   0 0 0 0 0 k46 0 0 k49 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 k64 0 0 k67 0 0 0 0 0;...
   0 0 0 0 0 k76 0 0 k79 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 k94 0 0 k97 0 0 0 0 0;...
   0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0];

K_in(:,:,3) = ...
    [0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0];
K_in(:,:,4) = ...
    [1 0 0 -1 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        -1 0 0 1 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0];
K_in(:,:,5) = ...
    [0 0 0 0 0 0 0 0 0 0 0 0;0 1 0 0 -1 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 -1 0 0 1 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0];
K_in(:,:,6) = ...
    [0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 1 0 0 -1 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 -1 0 0 1 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0];
K_in(:,:,7) = ...
    [0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 1 0 0 -1 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 -1 0 0 1 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0];
K_in(:,:,8) = ...
    [0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 1 0 0 -1 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 -1 0 0 1 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0];
K_in(:,:,9) = ...
    [0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 1 0 0 -1 0 0 0;...
```

```
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 -1 0 0 1 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0];
K_in(:,:,10) = ...
    [0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 1 0 0 -1 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 -1 0 0 1 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0];
K_in(:,:,11) = ...
    [0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 1 0 0 -1 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 -1 0 0 1 0;0 0 0 0 0 0 0 0 0 0 0 0];
K_in(:,:,12) = ...
    [0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 1 0 0 -1;...
        0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 0 0 0 0;0 0 0 0 0 0 0 0 -1 0 0 1];


% -------------------------------------------------------------------------------------
% determine input force
% -------------------------------------------------------------------------------------
% magnitude of wind excitation and ground force [N] - sine wave
freq = 30;      % frequency of a sine wave [rad/sec]

% define load applied according to the selected case
if case_no == 0,
    wind_f = [0 0 0 0];
    ground_f = 500;

elseif case_no == 1,
    wind_f = 2.5e6*[1 1 1 1];   % case 1 - load at all story
    ground_f = 0;

elseif case_no == 3 | 4,
    wind_f = 8e6*[0 0 0 1];     % case 3 & 4 - roof excitation
    ground_f = 0;

end

% -------------------------------------------------------------------------------------
% run simulink model 'EOM_benchmark.mdl'
% -------------------------------------------------------------------------------------

% Three different method to produce dis, vel, acc
sim('EOM_benchmark12',[0:t_step:t_final]);


% -------------------------------------------------------------------------------------
-
% determine K_damanged = K + △K analytically
% just for checking with simulation result
K_delta = alphas(time_no,1)*K_in(:,:,1) + alphas(time_no,2)*K_in(:,:,2) + ...
    alphas(time_no,3)*K_in(:,:,3) + alphas(time_no,4)*K_in(:,:,4) + ...
    alphas(time_no,5)*K_in(:,:,5) + alphas(time_no,6)*K_in(:,:,6) + ...
    alphas(time_no,7)*K_in(:,:,7) + alphas(time_no,8)*K_in(:,:,8) + ...
    alphas(time_no,9)*K_in(:,:,9) + alphas(time_no,10)*K_in(:,:,10) + ...
```

```
    alphas(time_no,11)*K_in(:,:,11) + alphas(time_no,12)*K_in(:,:,12);
K_damaged = K + K_delta;

% ------------------------------------------------------------------------------------
% compute difference (error) between the original and damaged
acc = acc'; vel = vel'; dis = dis'; F_in = F_in';
y_out = F_in - M*acc - C*vel - K*dis;

% to verify the simulation data (y_zero should be zero)
y_zero = F_in - M*acc - C*vel - K_damaged*dis;



% ------------------------------------------------------------------------------------
% Adaptive LMS filtering, for n weights, m taps
% ------------------------------------------------------------------------------------

filter_no = dof;        % number of signal (filter) = number of DOF
tap_no = 5;             % number of weights for each signal
mu = 0.3;               % LMS factor: controls stability and rate of convergence
w(dof,tap_no,tap_no+time_no) = 0;  % initialise weight matrix

% redefine x and y
% add m-1 zero vectors for prior time step calculation
x_temp = dis;
y_temp = y_out;

x = zeros(dof,tap_no+time_no);
y = zeros(dof,tap_no+time_no);

x(:,tap_no+1:tap_no+time_no) = x_temp;
y(:,tap_no+1:tap_no+time_no) = y_temp;

% adaptive LMS weight updating, One Step method
for i1 = tap_no+1:time_no,

    n_temp = zeros(dof,dof);
    sum_n = 0;
    Q = 0;
    G = zeros(dof,tap_no);

    for j = 1:tap_no,
        for i2 = 1:dof

            % i) exact w/ prior time step disps
            % n_temp = w(i2,j,i1)*K_in(:,:,i2)*x(:,i1-j+1);

            % ii) w/o prior time step disps (main method)
            n_temp = w(i2,j,i1)*K_in(:,:,i2)*x(:,i1);

            sum_n = sum_n + n_temp;
        end        % for i
    end            % for j

    Q = sum_n;

    e(:,i1) = y(:,i1) - Q;
```

```
    for i3 = 1:dof,
        for j2 = 1:tap_no,

            % i) exact with prior time displacements
            %G(i3,j2) = e(:,i1)'*K_in(:,:,i3)*x(:,i1-j2+1);

            % ii) exact, current disps only
            % G(i3,j2) = e(:,i1)'*K_in(:,:,i3)*x(:,i1);

            % iii) w/o coupling and prior disps (main method)
            G(i3,j2) = e(i3,i1)*x(i3,i1-j2+1);

            % iv) w/o coupling w/o prior disps
            % G(i3,j2) = e(i3,i1)*x(i3,i1);

        end
    end

    G = 2*mu*G;

    w(:,:,i1+1) = w(:,:,i1) + G;

end

% sum up for m elements of a weight
w_temp = 0;
for j = 1:tap_no,
    w_temp = w_temp + w(:,j,:);
end

% redefine index for plotting
for i = 1:time_no-tap_no+1,
    w1(:,:,i) = w_temp(:,:,i+tap_no-1);
end


% ------------------------------------------------------------------------------------
% plot results (LMS and actual alphas vs time)
% ------------------------------------------------------------------------------------
%t_axis = 0:t_step:t_final;
t_axis = 0:t_step:t_final-t_step*(tap_no-1);
figure,
%plot(t_axis,w(:,1:time_no),':',t_axis,alphas)
plot(t_axis,w1(:,1:time_no-tap_no+1),':',t_axis,alphas(1:time_no-tap_no+1,:))
title(['Benchmark Structure, 12 DOF: ΔK, \mu = ' num2str(mu(1)), ...
        ', Time step = ' num2str(t_step) ' (sec), ' num2str(tap_no) ' taps']), grid
ylabel('ΔK, Change in \alphas'), xlabel('Time (sec)')
text(12,-0.4e7,['Case ' num2str(case_no) ', Damage Pattern ' num2str(damage_no)], ...
    'fontsize',12,'fontweight','bold')


%************************************************************************************
%************************** End of LMS_benchmark12_new.m ***************************
%************************************************************************************
```

A5. Matlab Code: S-function

```
%*************************************************************************************
% FILE NAME: S_DELTA_BENCHMARK12.M
%       changes in stiffness coefficent as the simulation runs
%       this function called by EOM_benchmark12.mdl
%       Created by Leo Hwang
%       Department of Mechanical Engineering, Unveristy of Canterbury, New Zealand
%*************************************************************************************

function [sys,x0,str,ts] = s_delta_benchmark12(t,x,u,flag,t_change,damage_no)
% This s-function generates varying alphas with time, and
% is called from 'EOM_benchmark12.mdl', which is called from 'LMS_benchmark12.m'

switch flag,
    % Initialization %
    case 0,
        [sys,x0,str,ts]=mdlInitializeSizes;
        % Derivatives %
    case 1,
        sys=mdlDerivatives(t,x,u,t_change,damage_no);
        % Update %
    case 2,
        sys=mdlUpdate(t,x,u,t_change,damage_no);
        % Outputs %
    case 3,
        sys=mdlOutputs(t,x,u,t_change,damage_no);
        % GetTimeOfNextVarHit %
    case 4,
        sys=mdlGetTimeOfNextVarHit(t,x,u,t_change,damage_no);
        % Terminate %
    case 9,
        sys=mdlTerminate(t,x,u,t_change,damage_no);
        % Unexpected flags %
    otherwise
        error(['Unhandled flag = ',num2str(flag)]);
end

%==============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%==============================================================================
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 12;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;   % at least one sample time is needed
sys = simsizes(sizes);
% initialize the initial conditions
x0  = [];
% str is always an empty matrix
```

```
str = [];
% initialize the array of sample times
ts  = [0 0];
% end mdlInitializeSizes

%==========================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%==========================================================================
function sys=mdlDerivatives(t,x,u,t_change,damage_no)
sys = [];
% end mdlDerivatives

%==========================================================================
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time step
% requirements.
%==========================================================================
function sys=mdlUpdate(t,x,u,t_change,damage_no)
sys = [];
% end mdlUpdate

%==========================================================================
% mdlOutputs
% Return the block outputs.
%==========================================================================
function sys=mdlOutputs(t,x,u,t_change,damage_no)

% changing in alphas
if damage_no == 0,          % undamaged
    alpha = 1e8 * [0 0 0 0 0 0 0 0 0 0 0 0];

elseif damage_no == 1,      % damage pattern 1
    alpha = 1e8 * [-0.4823 -0.4822 -1.5072 0 0 0 0 0 0 0 0 0];

elseif damage_no == 2,      % damage pattern 2
    alpha = 1e8 * [-0.4823 -0.4822 -1.5072 0 0 0 -0.4823 -0.4822 -1.5072 0 0 0];

elseif damage_no == 3,      % damage pattern 3
    alpha = 1e8 * [0 -0.1205 -0.1884 0 0 0 0 0 0 0 0 0];

elseif damage_no == 4,      % damage pattern 4
    alpha = 1e8 * [0 -0.1205 -0.1884 0 0 0 -0.1205 0 -0.1884 0 0 0];

end

% damage profile

% i) sudden failure (step change)
% for i = 1:12,
%     if u(1) < t_change,
%         sys(i) = 0;
%     else
%         sys(i) = alpha(i);
%     end
% end
```

```
% ii) gradual failure (ramp change)
grad = 2e7 * [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1];
for i = 1:12
    if u <= t_change,
        sys(i) = 0;
    elseif u <= t_change + alpha(i)/grad(i),
        sys(i) = grad(i)*u - grad(i)*t_change;
    else
        sys(i) = alpha(i);
    end
end

% end mdlOutputs

%=============================================================================
% mdlGetTimeOfNextVarHit
% Return the time of the next hit for this block.  Note that the result is
% absolute time.  Note that this function is only used when you specify a
% variable discrete-time sample time [-2 0] in the sample time array in
% mdlInitializeSizes.
%=============================================================================
function sys=mdlGetTimeOfNextVarHit(t,x,u,t_change,damage_no)
sys = [];
% end mdlGetTimeOfNextVarHit

%=============================================================================
% mdlTerminate
% Perform any end of simulation tasks.
%=============================================================================
function sys=mdlTerminate(t,x,u,t_change,damage_no)
sys = [];
% end mdlTerminate

%************************************************************************************
%************************** End of s_delta_benchmark12.m ****************************
%************************************************************************************
```

A6. Simulink Model: 12 DOF model

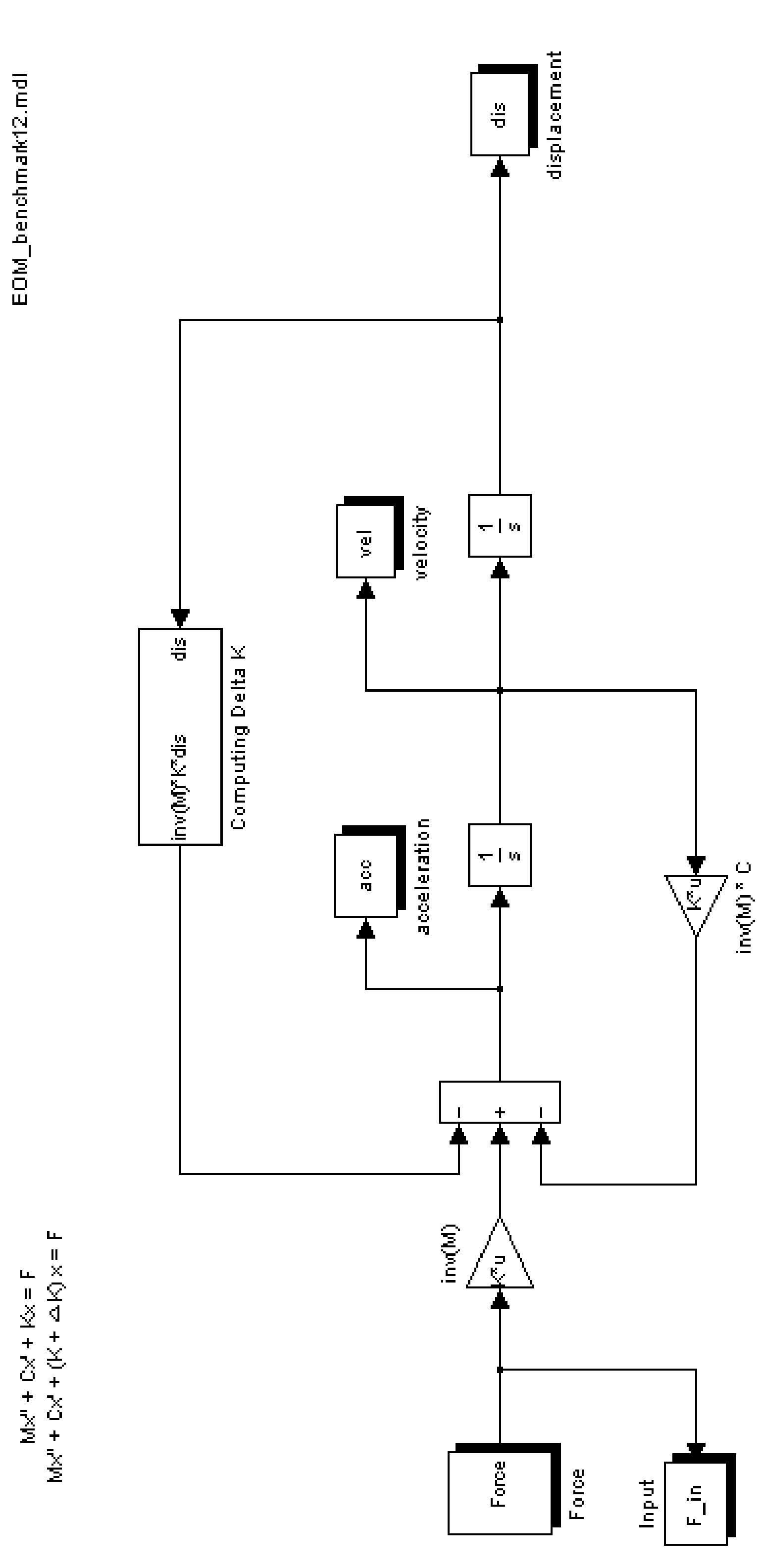


Figure 83 Simulink representation for 12 DOF model of Benchmark structure: `EOM_benchmark12.mdl`

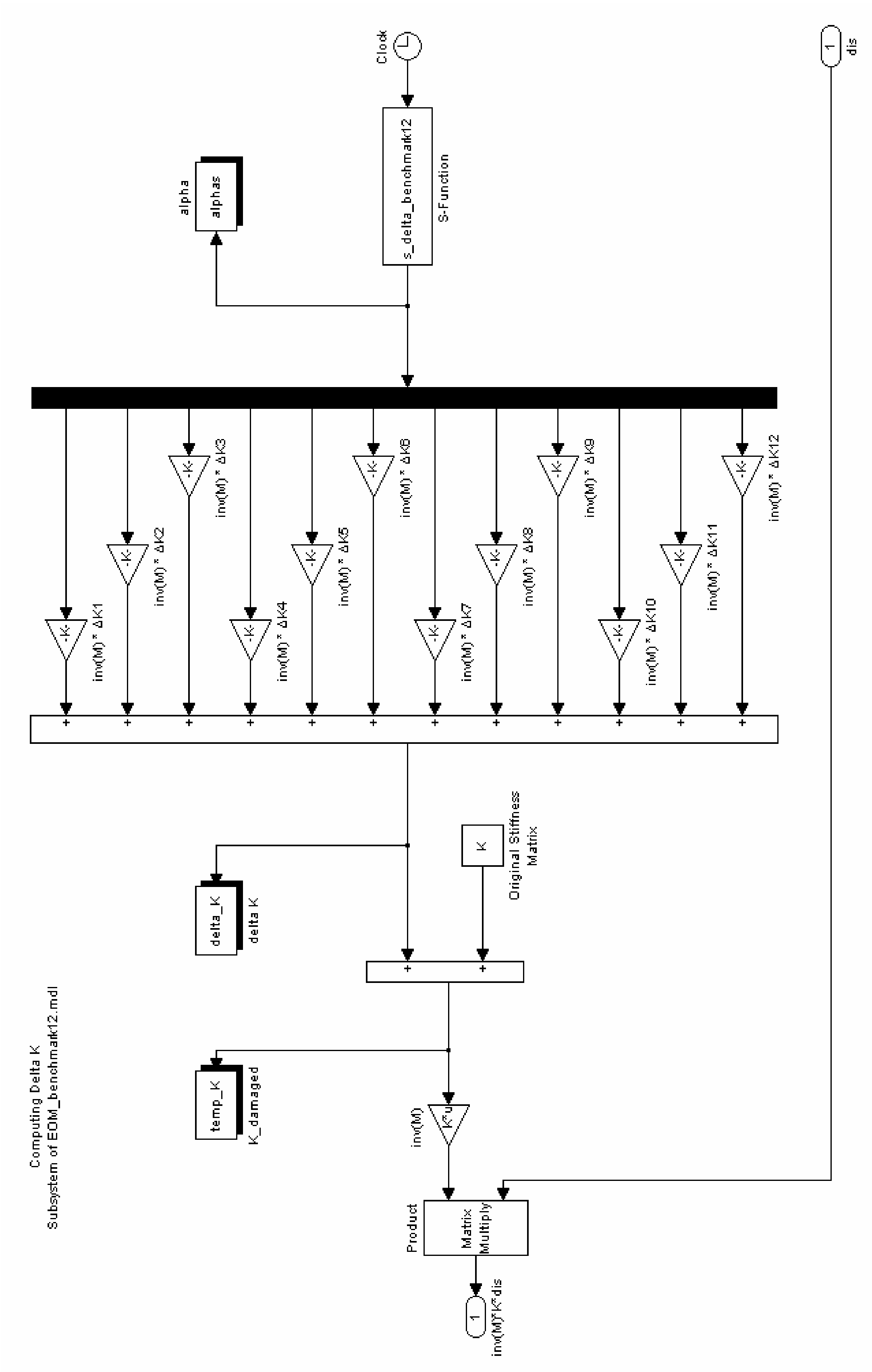


Figure 84 `Computing Delata K`: Subsystem of `EOM_benchmark12.mdl`, creating damage by changing α values