ONLINE LEARNING ALGORITHM FOR STRUCTURAL CONTROL USING MAGNETORHEOLOGICAL ACTUATORS

By

SIMON LAFLAMME

Bachelor of Engineering, Civil Engineering and Applied Mechanics
McGill University, 2006

Bachelor of Commerce, Economics and Finance
McGill University, 2003

submitted to the Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Civil and Environmental Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHONOLOGY

JUNE 2007

Signature of Author: ____
Department of Civil and Environmental Engineering
May 11, 2007

Certified by: _____
Jerome J. Connor
Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by: _____
Daniele Veneziano
Chairman, Departmental Committee on Graduate Studies

ONLINE LEARNING ALGORITHM FOR STRUCTURAL CONTROL USING MAGNETORHEOLOGICAL ACTUATORS

By

SIMON LAFLAMME

Submitted to the Department of Civil and Environmental Engineering
on May 11th, 2007, in partial fulfillment of the
requirements for the Degree of Master of Engineering in
Civil and Environmental Engineering

Abstract

Magnetorheological actuators are promising devices for mitigating vibrations because they only require a fraction of energy for a similar performance to active control. Conversely, these semi-active devices have limited maximum forces and are hard to model due to the rheological properties of their fluid. When considering structural control, classical theories necessitate full knowledge of the structural dynamic states and properties most of which can only be estimated when considering large-scale control, which may be difficult or inaccurate for complicated geometries due to the non-linear behaviour of structures. Additionally, most of these theories do not take into account the response delay of the actuators which may result in structural instabilities.

To address the problem, learning algorithms using offline learning have been proposed in order to have the structure learn its behaviour, but they can be perceived as unrealistic because earthquake data can hardly be produced to train these schemes. Here, an algorithm using online learning feedback is proposed to address this problem where the structure observes, compares and adapts its performance at each time step, analogous to a child learning his or her motor functions. The algorithm uses a machine learning technique, Gaussian kernels, to prescribe forces upon structural states, where states are evaluated strictly based on displacement and acceleration feedback.

The algorithm has been simulated and performances assessed by comparing it with two classical control theories: clipped-optimal and passive controls. The proposed scheme is found to be stable and performs well in mitigating vibrations for a low energy input, but does not perform as well compared to clipped-optimal case. This relative performance would be expected to be better for large-scale structures because of the adaptability of the proposed algorithm.

Thesis Supervisor: Jerome J. Connor
Title: Professor of Civil and Environmental Engineering

Acknowledgements

I would like to extend my appreciation to Professor Jerome J. Connor for his expert guidance and mentorship, and for his encouragement and support throughout the preparation of this thesis.

Also, thanks to Professor Tomaso Poggio and Jake Bouvrie for their technical advice regarding machine learning.

I would furthermore like to show gratitude to my family and friends who have supported and encouraged me along this task, with special thanks to Dustin Smith, whose scientific discussions have encouraged me to research the field of artificial intelligence to find practical tools in order to solve the addressed problem and Ann Walker, whose emotional support throughout my degree certainly made a difference in the quality of my research.

Table of Contents

1.0 INTRODUCTION

Mitigating vibrations in building is critical whether it is for small or large excitations in order to prevent serviceability issues or worse, structural damages. For instance, occupants may experience acute health effects under constant small vibrations, or walls may crack under larger vibrations. Keeping the structure serviceable and more importantly, protecting its occupants, is what motivates the field of structural control.

Structural control can be achieved with a passive scheme, which requires no energy input, or by creating an adaptive structure. Adaptive structures are defined as structures that are capable of responding to environmental changes, such as strong winds and earthquakes, and also to internal changes such as a failure of a primary member. This adaptation is achieved by a modification of the geometric configuration or of certain structural properties.

The problem with adaptive structures is that they require a complex adaptive control system which includes a monitoring system, a decision system and an actuator system. There is a need for further research and technology development of these components. Actuators, for instance, are not yet economically viable to be installed on large scale buildings as they need a significant quantity of energy to operate. Semi-active actuators, particularly magnetorheological (MR) fluid actuators, are very efficient on energy consumption as they need only a small fraction of the power input required for traditional actuators. However, they provide at most a force of the order of 200 kN to the structure, which is insufficient for tall buildings exposed to high winds. Despite this force issue, the low energy requirement of MR actuators is a significant asset and they are regarded as promising devices when it comes to vibration mitigation.

Conversely, MR actuators are not broadly implemented. It is due to the fact that their installation requires multiple actuators which can results in modeling complications because of the coupling action between the actuators and the plant as well as the non-linear behaviour of the structure.

The objective of this thesis is to propose a learning scheme that would overcome these difficulties. By learning, it is meant that the structure would learn how it is behaving upon the application of forces by the semi-active actuator, which is analogous to a child learning his motor functions. The challenge here is to use an algorithm that would learn and try to improve following each time step (online learning) rather than having a complete set of inputs-outputs data and simultaneously learning the function that would represent the building behaviour (offline learning).

The algorithm proposed in this thesis is a new method to address the problem of training adaptive structures which is somewhat more realistic than current approaches using offline learning and easier of implementation.

The text is organized as follows. Sections 2 and 3 give theoretical information about MR actuators and machine learning respectively. Section 4 describes the proposed algorithm using the theory exposed in the previous sections. A simulation assessing the performance of the algorithm is presented in section 5 and finally, section 6 makes concluding remarks.

2.0 MAGNETORHEOLOGICAL ACTUATORS

Magnetorheological actuators are defined by the rheological fluid that gives them their rheological properties. This particular fluid is made of polarizable and magnetizable particles that line up upon magnetic excitation, causing a change in the liquid's viscosity within a few milliseconds (Wu *et al.*, 2004). For a large-scale 180 kN MR damper, this response would be on the order of 60 milliseconds (Lord Corporation).

Their low power requirement, which is 50 W for a 200 kN damping force (Yang *et al.* 2002), also makes them very attractive as simply a battery is needed to drive their responses. This would allow the structure to remain stable under a global power failure which is probable during a natural disaster. Moreover, if a local power failure is to occur, MR actuators will act as passive dampers, providing a minimum damping to the structure.

Magnetorheological actuators are commonly used on mechanical devices in the transportation industry such as cars, planes and trains. Their first large-scale application to civil engineering structures only took place in 2001 (Spencer and Nagarajajah, 2003) to mitigate vibrations of stay cables on a cable-stayed bridge, the Dongting Lake Bridge, as well as on the Tokyo National Museum of Emerging Science and Innovation. The main challenge faced by MR actuators is their low force capacity compared to active devices. So far, the largest MR device manufactured by Lord Corporation has a capacity of 200 kN. Its characteristics are listed in table 2.1.

Stroke	±8 cm
F_{max}/F_{min}	10.1@10 cm/s
Cylinder bore (ID)	20.32 cm
Max. input power	<50 W
Max. force (nominal)	200,000 N
Effective axial pole length	8.4 cm
Coils	3×1050 turns
Fluid $\eta/\tau^2_{0(field)}$	2×10^{-10} s/Pa
Apparent fluid η	1.3 Pa-s
Fluid $\tau_{0(field)}$max	62 kPa
Gap	2 mm
Active fluid volume	~90 cm^3
Wire	16 gauge
Inductance (L)	~6.6 henries
Coil resistance (R)	3×7.3 ohms

Table 2.1: Characteristics of a 200 kN MR damper (Yang *et al.*, 2002)

This limited force per actuator results in the need for multiple devices for a large structure as the prescribed forces are likely to be higher than 200 kN. In the case of the Dongting Lake Bridge, a single MR actuator was installed on each of the stay cable, providing each of them with independent control. A multiple installation would require using a central control scheme, selecting efficient locations and determining a sufficient number of devices.

2.1 CONTROL OF MR ACTUATORS

Effective controls are hard to achieve for MR actuators because some of their rheological properties are problematic to model. One of these properties is the stiction phenomenon, represented in figure 2.1. It happens when the liquid changes from the pre-yield to post-yield state at maximum stress. The liquid then starts to flow more rapidly and causes a sudden decrease in force and a displacement lag. There is also the shear tinning effect which is observed when displacement oriented servo-controllers are used: as the velocity approaches zero, there is a rapid decrease of the plastic force (Yang *et al.*, 2004). These phenomena are shown in figure 2.2.

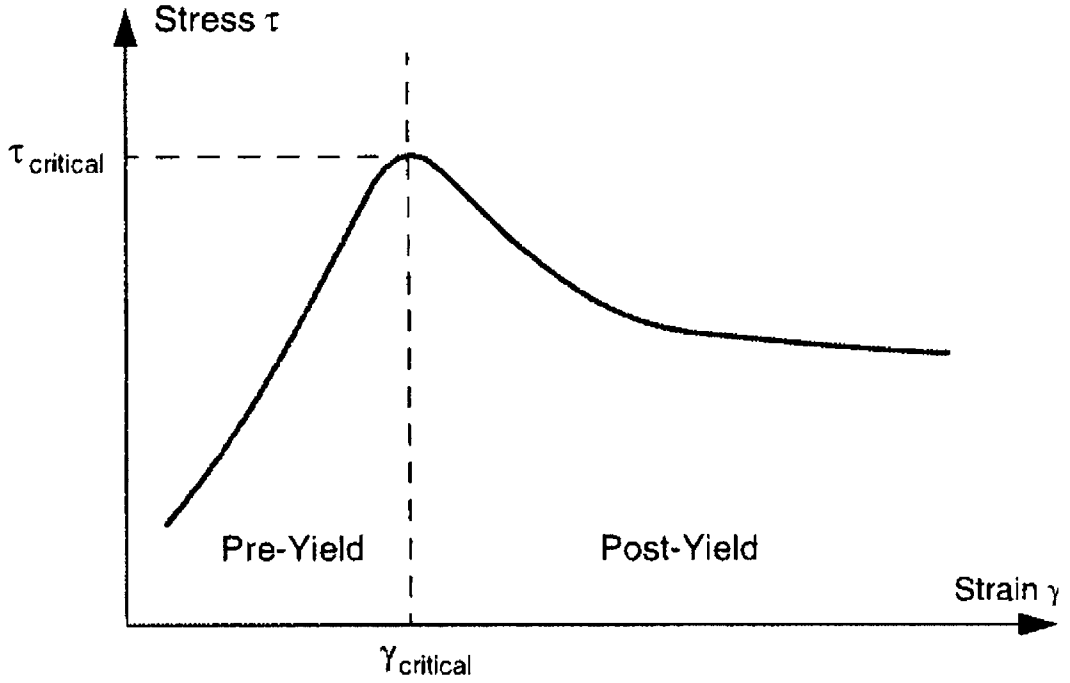

Figure 2.1: Stiction phenomenon (Yang *et al.* 2004)

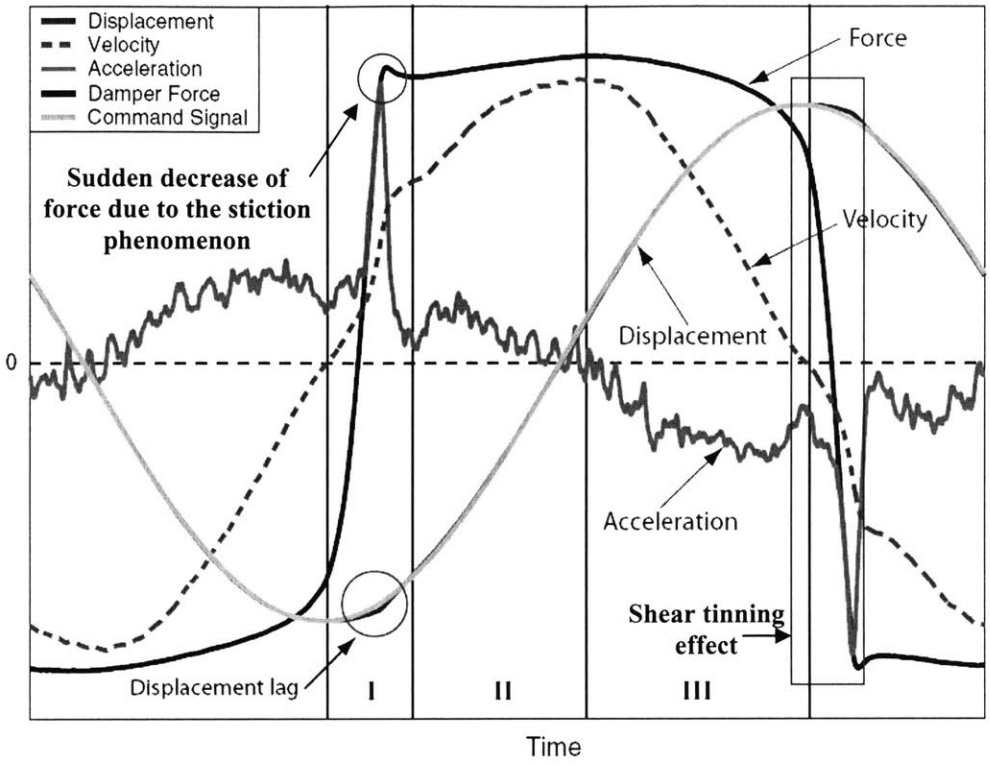

Figure 2.2: Force-Displacement relationship (adapted from Yang *et al.* 2004)

It is possible to neglect the shear tinning effect by using a velocity or acceleration feedback controller since the control will not risk of prescribing "no control force" as being the optimal control over a small time step (Ribakov and Gluck 2001). On the other hand, a pure displacement feedback is likely to cause instability in the system (Connor, 2003). Additionally, it is hard to measure displacements during an earthquake as there is no absolute reference.

2.1.1 Fuzzy Control

Liu *et al* (2005) made and experiment on a small scale bridge in order to assess the performances of four instantaneous control schemes, namely: energy minimization, Lyapunov, fuzzy logic and variable structure system fuzzy logic. The experimented bridge was equipped with two MR dampers in parallel between the bridge deck and abutment. This specific installation in parallel only necessitates a single response of both

dampers as their forces are added. Hence, the bridge is modeled as a single-degree-of-freedom (SDOF) system.

The experiment has shown that all control schemes have about the same performances when it comes to displacement and acceleration control. However, the fuzzy control scheme was significantly optimal for the energy input prescribed (figure 2.3).

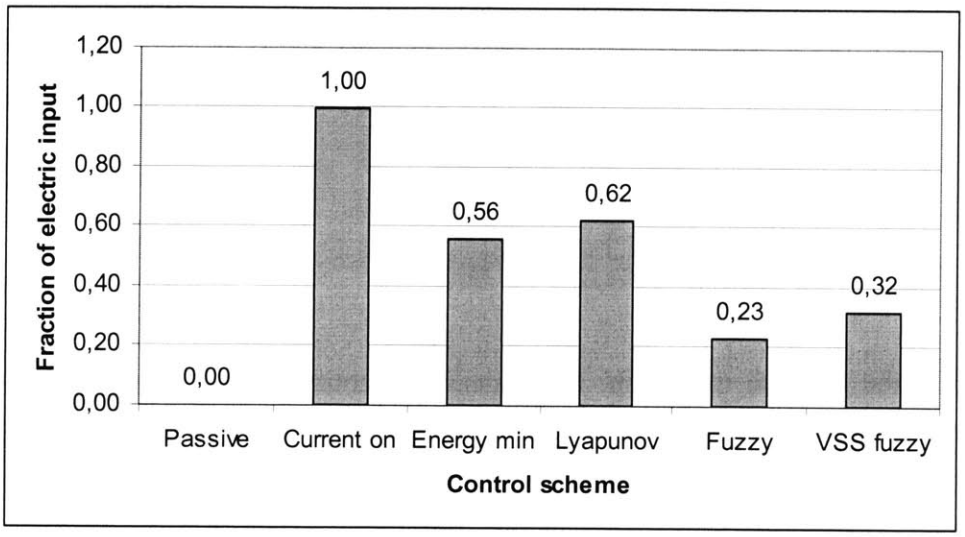

Figure 2.3: Fraction of electric input used by each control scheme (adapted from Liu *et al.* 2005)

These control schemes give a good idea of the relative power requested by a logic control such as the fuzzy logic controller. Fuzzy logic is used with a table of state-output controls: if x_1 and x_2 happen, then do y. A weakness of fuzzy logic is its application to multi-degree-of-freedom (MDOF) systems. While the logic rules are easy to imagine for a single DOF being controlled, the decision table would contain as many dimension as the number of DOF observed. Furthermore, such scheme becomes extremely complex for a decentralized control scheme.

2.1.2 Clipped-Optimal Control

Jansen and Dyke (2000) have studied the performance of five control schemes, namely Lyapunov, decentralized bang-bang, maximum energy dissipation, clipped-optimal

control and modulated homogeneous friction, on a six storeys small scale building with two MR dampers installed in series, testing control schemes for a MDOF system. The experiment resulted in the clipped-optimal controller being optimal when it comes to reducing the acceleration response and, interestingly, it was able to reduce the acceleration on each storey despite that the MR actuators were only installed on the first two storeys.

The clipped-optimal strategy is to find a linear optimal controller K_c based on the local structural response and force to determine desired forces (Jansen and Dyke (2000)):

$$f_c = L^{-1}\left\{-K_c(t)L\left\{\begin{matrix} y \\ f \end{matrix}\right\}\right\} \tag{2-1}$$

where L is the Laplace transform, f the measured control force vector, f_c the desired control force vector, y the observed structural responses.

As the MR actuator's response force is dependant on the local force, the control scheme can be executed by having a voltage response instead of a direct force, which voltage controls the actuator's force. Hence, if the force required is equal to the local force, then the actual voltage remains stable. If the force required is greater and of the same sign as the local force, then the voltage is increased to the maximum voltage. Otherwise, the voltage is turned off. This can be written as the following equation:

$$v_i = V_{\max}\left\lfloor H(f_C - f\,)f\,\right\rfloor \tag{2-2}$$

where H is the heaviside step function, V_{max} the maximum voltage (property of the MR actuator).

Lyapunov and clipped-optimal controls are superior algorithms when it comes to efficiency in controls (Heo *et al.*, 2006b). In an experiment to develop an algorithm for a Squeeze Mode MR actuator, Heo *et al.* (2006b) developed two unified control

algorithms: one based on a Lyapunov Control, the other based on the clipped-optimal control. A unified control has the advantage to be capable of controlling more than the first few modes of the structure and to avoid spillover problems. Both algorithms have shown good and similar performances in the specific experiment and achieved a reduction of about 30% of energy consumption compared with the passive-on case. The clipped-optimal, however, was found to be the most effective in displacement reduction and had an acceleration feedback strategy.

This clipped-optimal control has been used by Cho *et al.* (2005a) while trying to control the first few modes of a structure. In their study, the authors are using the H_2/LQG approach to obtain a control matrix to solve for the modal shapes of these first few modes. The study showed that modal control is an efficient approach to control a structure with multiple MR actuators, and that it can be efficiently done by only controlling the first two modes, using acceleration feedback. The challenge faced by using modal control is to estimate modal shapes of the structure. It would be expensive to use many sensors to account for the full shape, and using only a few sensors would fail to derive the full modal shape. It is possible to evaluate these modal shapes by using an observer. Luenberger observers can be utilized for low noise-to-signal ratios while a Kalman-Bucy filter is more suitable for high noise-to-signal ratios (Cho *et al.*, 2005a).

2.1.3 Discrete Time Invariant Systems

Describing dynamic systems as discrete time invariant systems can be quite handy when considering computer programming. A discrete formulation can be derived from the state-space formulation of the equation of motion:

$$\dot{X} = AX + B_f F + B_g a_g + B_p p \tag{2-3}$$

where:

$$X = \begin{Bmatrix} u \\ \dot{u} \end{Bmatrix} \; ; \; A = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \; ; \; B_f = B_p = \begin{bmatrix} 0 \\ M^{-1}E_f \end{bmatrix} \; ; \; B_g = \begin{bmatrix} 0 \\ -E \end{bmatrix}$$

and E is a vector of 1's, E_f is a matrix representing the position of the actuators, M the mass matrix, C the damping matrix and K the stiffness matrix; F the control force, p an external force, a_g the ground acceleration.

The solution of equation (2-3) can be found using Duhamel integrals where its general solution has the form:

$$X(t) = e^{A(t-t_0)}X_0 + \int_{t_0}^{t} e^{A(t-\tau)}G(\tau)d\tau \tag{2-4}$$

where $G(t) = B_f F + B_g a_g + B_p p$.

Considering two times, t_{j+1} and t_j, and assuming the system parameters are time invariant (constant), equation (2-4) is approximated by the following algebraic form:

$$X_{j+1} = e^{A\Delta t}X_j + A^{-1}\left(e^{A\Delta t} - I\right)\left[B_g a_{g,j} + B_f F_j\right] \tag{2-5}$$

where Δt is the sampling time step.

Equation (2-5) is useful when evaluating time step responses. Connor (2003) provides a more complete derivation of the time invariant system discrete formulation.

It follows that actuators can be controlled using the discrete formulation. A standard performance index for optimal linear feedback control is:

$$J = \frac{1}{2}\sum_{j=0}^{\infty} X_j^T (Q + K_f^T R K_f) X_j \tag{2-6}$$

where X_{f+1} is evaluated using equation (2-5), Q is a diagonal matrix weighting the displacements, R is a diagonal matrix weighting the forces, K_f is the linear state feedback matrix. One evaluates K_f by solving the discrete time algebraic Riccati equation based on the performance index J. Connor (2003) presents an elegant derivation of the final form of the Riccati equation. This may also be solved using the function DARE in MATLAB (Connor, 2003).

2.1.4 Predictive Control

It is possible that the rheological properties, combined with the short delay in response, cause instability into the mitigated system that can be worse than for the case of an undamped system (Ribakov and Gluck, 2001). Therefore, control systems must be implemented with care. Ribakov and Gluck (2001) introduced a predictive control that tries to predict the future response of the system. The algorithm observes the actual response measured by the sensors at time t for a force that has been applied at time $t-1$, and evaluates the future structural response at time $t+1$ for a force applied at time t. This process has been developed for a MDOF system where MR actuators were installed in series in a high-rise structure, which installation in series is highly sensitive to differences in delay of response as the errors can be amplified. The experiment has resulted in predictive control to give similar but improved performance as for the instantaneous control schemes.

It can be noted that the time invariant discrete formulation can be used in a predictive control scheme. A force can be assumed to be ordered by the control scheme at time t and applied at time $t + \Delta t$ where consequently one would need to find a good displacement predictor. For instance, Beaver (1999), in order to control a semi-active actuator for a SDOF system, used a displacement predictor based on the Central Difference Method:

$$u_{j+1} = \left(\frac{\frac{2m}{\Delta t^2} - k}{\frac{m}{\Delta t^2} + \frac{c}{2\Delta t}} \right) u_j + \left(\frac{\frac{c}{2\Delta t} - \frac{m}{\Delta t^2}}{\frac{m}{\Delta t^2} + \frac{c}{2\Delta t}} \right) u_{j-1} + \left(\frac{1}{\frac{m}{\Delta t^2} + \frac{c}{2\Delta t}} \right) (-ma_{g,j} + F_j) \qquad (2\text{-}7)$$

2.1.5 Stability

Stability is a critical issue when it comes to structural control. The previous subsection highlighted the importance of taking response delays into consideration, but instabilities may also arise from the sampling time step Δt and types of feedback. The stability of the system can be assessed by investigating the eigenvalues of the matrix A in the state space representation (equation 2-3). To be stable, the real part of these eigenvalues must be negative.

Connor (2003) shows that pure displacement feedback produces an unstable behaviour for an undamped system since the real part of the eigenvalues is always positive for any Δt. This does not hold for pure velocity feedback, but the real part of the eigenvalues converges towards a positive value with increasing Δt and decreasing active damping. The author gives a good discussion about the stability for MDOF systems.

2.2 SYSTEM DESIGN

2.2.1 Power Input

MR actuators, as stated previously, have the advantage of requiring a small power input which is quite beneficial during a general power failure, as the device could rely on a battery.

Cho *et al.* (2005b) implemented an electromagnetic induction (EMI) on a 3 kN MR actuator. The EMI produces voltage by changing kinetic energy into potential energy. It is capable of producing a voltage output greater than the required input from the actuator (maximum of 2.5 V in this specific experiment), which would allow the MR actuator to be used as a smart passive system as no external power source would be needed. Despite that this may be feasible for a large scale MR actuator, it would be difficult, if not impossible, to think of utilizing this passive scheme to control such devices in series in order to create a global force meant to control a large structure.

Moreover, a voltage input is not the optimal input to drive a MR actuator. Current driven MR devices are optimal as they will induce a response significantly quicker than in the case of a voltage driven power supply (Yang *et al.* 2002). As shown in figure 2.4 for the case of a 20 ton MR damper, a voltage driven power supply will achieve 95% of its final value in 1 second while it will take 0.06 second for the current driven supply to fall into the 5% error range.

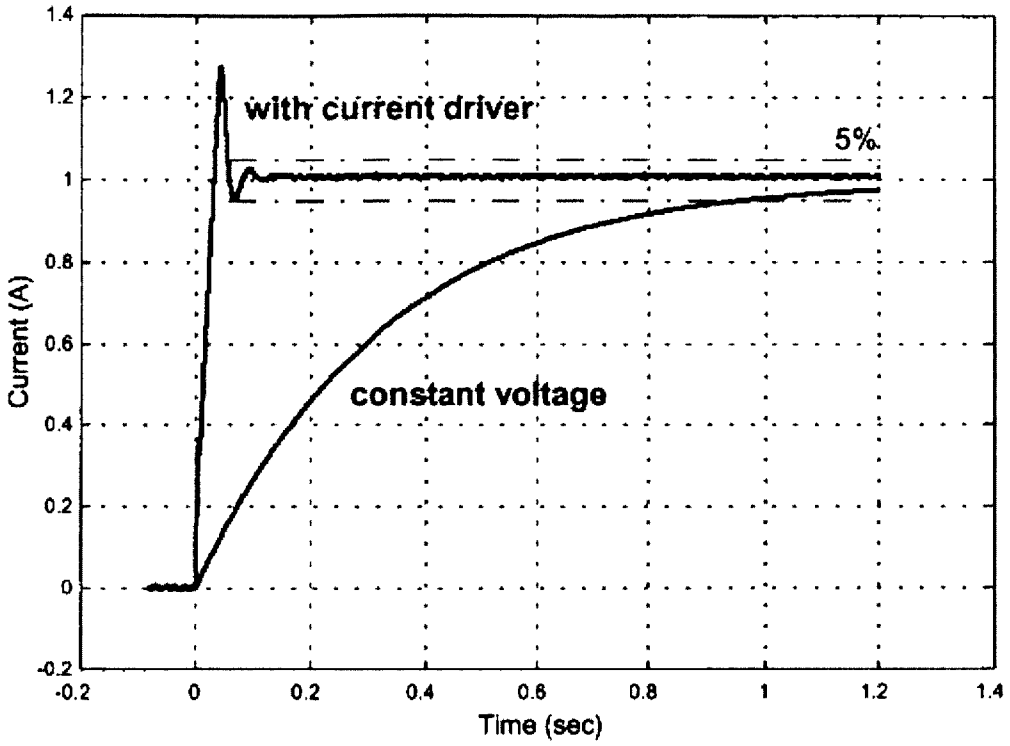

Figure 2.4 Comparison of current in coils between current driven and voltage driven devices (Yang *et al.* 2002)

2.2.2 Placement of Devices

The optimal number of actuators and positioning can be done heuristically by using computer simulations. Such a method would need to use as inputs the required damping forces in every axis (or DOF) and the allowable damping force per MR actuator. This would give, in a simple fashion, the number of MR actuators needed in each direction. This has the limitation of providing only the best positioning result among the different trials.

There exist, however, a more rigorous approach based on motion-based design. This approach requires to evaluate the mode shapes of the structure and to design the actuating strategy accordingly. The first mode is a rational control target as it generally accounts for most of the structural response. The mode shapes can be calculated by determining

the eigenvectors of the system or one can simply assume a linear shape for the first mode (not without a loss of accuracy).

The actuator efficiency is based on its location. The effect of an actuator on the global control of a structure is increasing with the relative displacement of the DOF where it is acting. The equation of motion for a particular mode of a system with actuators can be written as (Connor, 2003):

$$\tilde{m}_j \ddot{q}_j + \tilde{c}_j \dot{q}_j + \tilde{k}_j q_j = \Phi_j{}^T EF - \tilde{m}_j E_f \ddot{u}_g \tag{2-8}$$

where the left hand-side is the equation of motion of mode j, Φ is the mode shape of mode j, E is the positioning matrix of the force actuators, F the vector of actuating forces, E_f is a vector of all ones, $\ddot{u}_g$ is the ground acceleration.

It follows that the optimal locations for the actuators will be at the maximum elements of the mode shape. Conversely, other factors may drive the choice such as the size of one actuator and its weight where one might one to limit the quantity of devices per DOF.

3.0 MACHINE LEARNING

3.1 OVERVIEW

Machine learning can be defined as algorithms capable of adapting to improve their performance throughout experiences (Langley, 1996). The experience is extracted from an *environment*, in which *knowledge* is acquired by *learning* and adapted by evaluating its *performance*. In this specific context, improvement of the algorithm performance, termed training, and experiences are presented as data containing inputs-outputs pairs. This relationship is shown in figure 3.1.

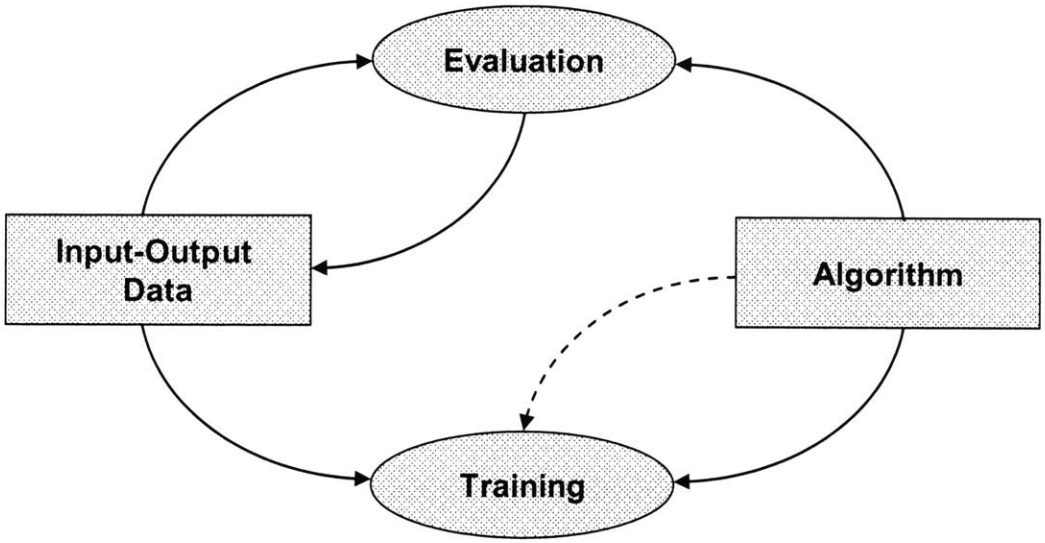

Figure 3.1: Interaction diagram for machine learning. The dashed line indicates an optional link (adapted from Langley, 1996)

3.1.1 Online versus Offline Learning

In machine learning, the way training is performed can be distinguished into two classes: online and offline learning. Online learning refers to training algorithms at subsequent time steps, often associated with time series problems. In opposition, offline learning is simultaneously training algorithms with already available data sets. Figure 3.2 is a modification of figure 3.1 expressing the difference between online and offline learning.

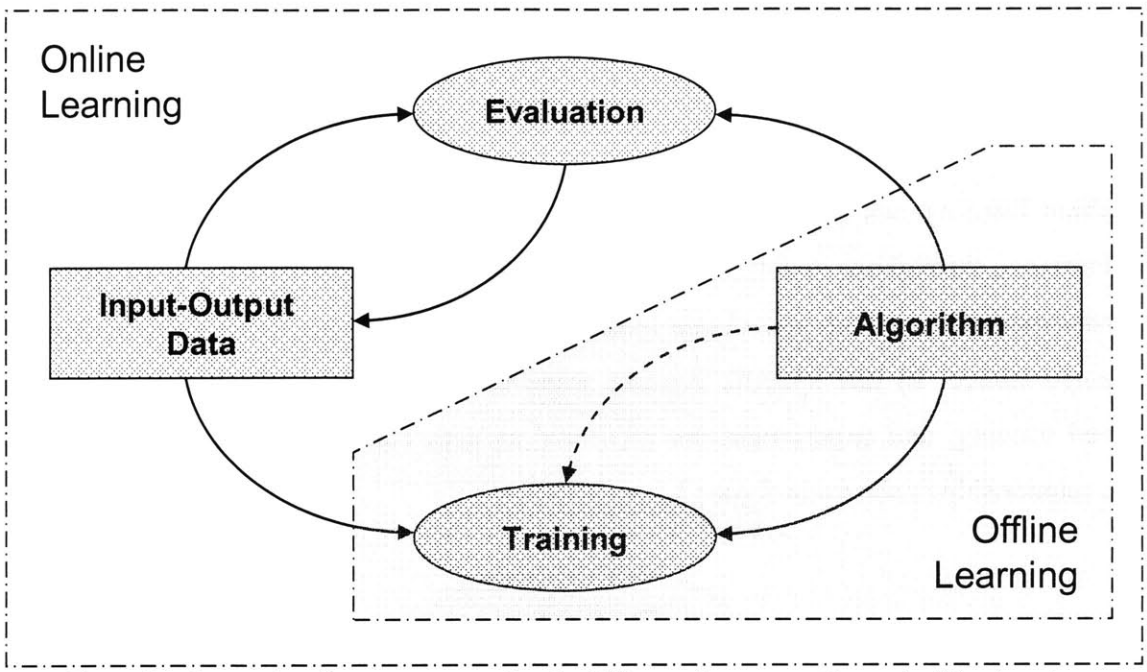

Figure 3.2: Interaction diagram representing the difference between online and offline learning

Most researches have focused on offline learning (Langley, 1996), resulting in abundant literature and applications using this scheme. However, the field of structural control uses many time series data. One can easily think of building excitations or responses with respect to time. In consideration, online learning merits particular attention.

Usually, algorithms using online learning require prior knowledge, such as an expected behaviour or a probability distribution, in order to attain a certain level of accuracy because there only exists a limited number of data. It results that a challenge in building an online learning algorithm is to achieve acceptable stability and performance at an early stage.

3.1.2 Supervised and Unsupervised Learning

The way the algorithms are evaluated depends on the degree of supervision which can be categorized as supervised and unsupervised learning.

Supervised learning uses inputs-outputs to evaluate performances. Hence, the correct step is known for each observation, which is quite handy for function approximation or classification problems. This type of feedback is used in the majority of researches (Langley, 1996).

Unsupervised learning uses outputs as the only available data. This type of feedback commonly addresses classification problems whereas the algorithm will try to establish different patterns in order to achieve classification.

Structural control can provide inputs-outputs pairs such as forces-displacements. Hence, it is appropriate to use supervised learning feedback. For this reason, this section presents two machine learning techniques, feed-forward neural networks and kernels, under the supervised learning paradigm. They could, however, be adapted for unsupervised learning.

3.2 ARTIFICIAL NEURAL NETWORKS

As the name suggests, artificial neural networks (ANNs) are inspired by biology, mimicking the human neurons. For the purpose of comparing ANNs with biological neurons, figure 3.3 shows a simplified schematic. A biological neuron can send a signal to other neurons only if the sum of the excitation received from its dendrites is above a certain threshold. If it does send a signal, the neuron is said to fire. The signal then proceeds from the axon to dendrites of adjacent neurons through synapses (Russell and Norvig, 2003).

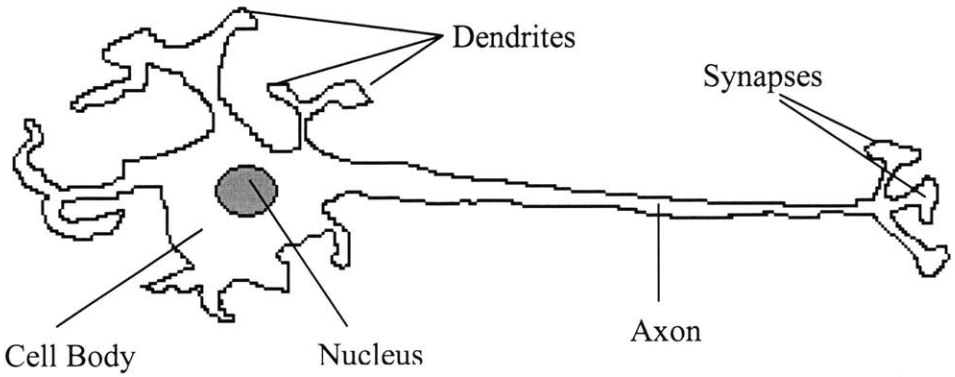

Figure 3.3: Simplified schematic of a human neuron

An artificial node is based on this biological knowledge (figure 3.4). The input links (dendrites) are multiplied by connection weights and added up together with a bias weight (the activation threshold) as an input function. The artificial neuron will process the information with respect to an activation function (cell body) and generate an output that will be directed to other neurons (axon) (Winston, 1992).

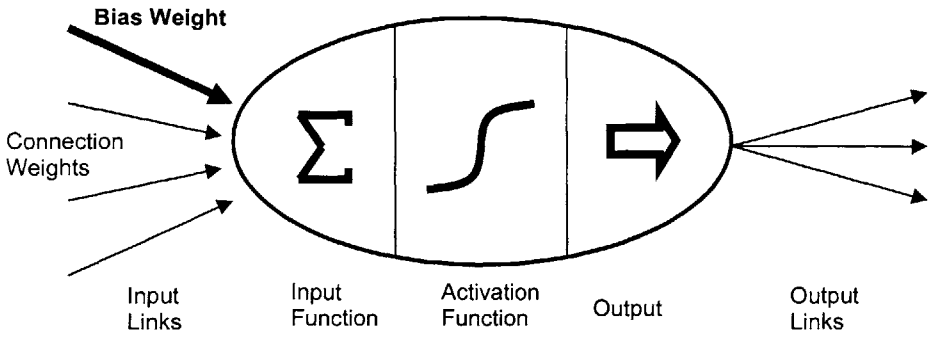

Figure 3.4: An artificial neuron (adapted from Russel and Norvig, 2003)

It follows that artificial neurons are set into a network. The network is built in order to receive inputs, process these inputs internally and finally generate outputs. In other words, ANNs are algorithms used to fit an unknown function and generate predictions based on inputs.

3.2.1 ANNs Architecture

ANNs were originally composed of a single layer. In the 1980's, intensive research has resulted in the discovery of feed-forward multilayer neural networks (Annema, 1995). Feed-forward multilayer neural networks are composed of several hidden layers comprised between an input and an output layer. They are called feed-forward because the input in processed forwardly until the output. There exist, however, many kinds of ANNs, among which the recurrent networks, where the outputs are proceeded back into the system. These last are currently attracting significant attention in research for their ability to handle time series data. A complete description of the types of neural networks is given in Hagan *et al.* (1996). Figure 3.5 shows a two-layer (hidden) neural network.

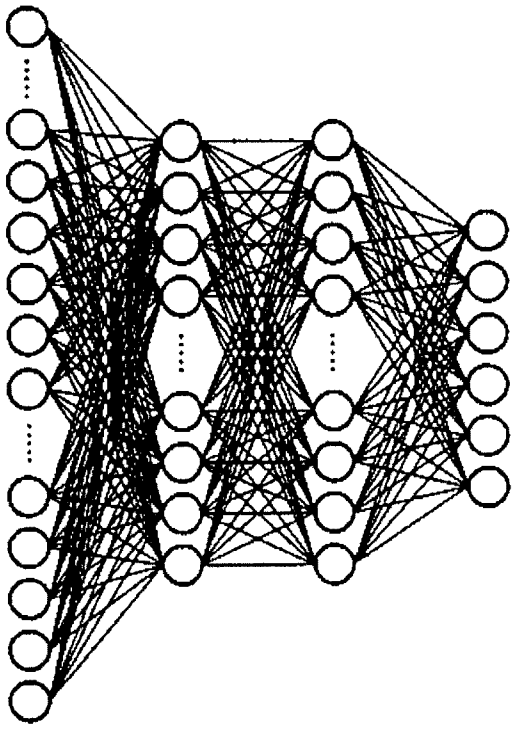

Figure 3.5: A two-layer feed-forward neural network (Jung and Kwon, 2004)

The challenge in using ANNs resides in the ability to choose consistent activation functions, to design a correct number of nodes (neurons) and layers, and to assign proper connection weights. All of these are, of course, interconnected.

Activation functions are typically the sigmoid function $\frac{1}{(1+e^{-x})}$ or a simple linear function. The sigmoid function is generally preferred over a step function because it is smooth and does not necessitate an "if" function. In the case of the sigmoid function, the bias weight represents the threshold as its purpose is to shift the function, setting the value above which the node will fire. The linear function will allow the output to match the desired order of response. There exists, however, several other useful activation functions and they can be found in Hagan *et al.* 1996.

The number of nodes and layers can be established upon ANNs theorems. One of the most powerful of these theorems is the Kolmogorov theorem establishing that a two-layer neural network with 2N+1 nodes in the first layer can exactly fit any function of a

N-dimensional input space, provided suitable activation functions and N>2 (Annema, 1995). Consequently, most of the designed neural networks use two hidden layers, which simplifies computations and response time. However, there is the risk of over-fitting a function that would result in inaccurate prediction results (Winston, 1992).

Assigning proper weights to a neural net can be tricky but is certainly the most attractive feature of the ANNs. ANNs are used for their ability to learn from past experience, an already existing set of inputs-outputs for the system, and adapt its connection weights as well as its bias weights to imitate the environment. Hence, the user task is not to assign these weights but to properly train the network with efficient algorithms.

Because of this ability to adapt through training and to mimic functions, neural nets are intelligent systems that may be effectively used to control structures: there is no need to evaluate the system properties such as mass and stiffness, and the ANN will take care of the system non-linear behaviour provided proper discretization of the structure and tuning of the network. This network, once designed, has to be properly trained. Considering a supervised learning feedback scheme, the most common and efficient way of training networks is called back-propagation training.

3.2.2 Back-Propagation Training

Back-propagation training of feed-forward networks is extensively used in training neural nets. As the name suggests, the connection weights on the last layer are upgraded and carried over to the next layer backward until the first layer. The network tries to minimize a performance function such as (typically):

$$J = \sum (Out_{ANN} - Out_d)^2 \tag{3-1}$$

where the outputs of the ANN are compared with the desired outputs from the training sample.

The influence of each connection weights of a layer on the global network outputs is computed and modified in order to achieve a better performance. This is done by taking the derivative of the performance function with respect to the connection weights. Hence, the update of a connection weight between node *i* of layer *m* to node *j* of layer *n* will be:

$$\Delta W_{im \to jn} = -\eta \frac{\delta J}{\delta W_{im \to jn}} \tag{3-2}$$

where η is the learning rate. This method is called the steepest gradient descent.

One of the main issues about back-propagation is the convergence of the model (Winston, 1992). There is no guarantee whatsoever that the neural net will hit a global minimum, nor will converge to any minima. The convergence issue can be minimized by proper architecture of nodes and layers and ensuring a well-posed problem. A problem is considered as well-posed if a unique solution exists and this solution depends on the continuity of the data, a famous principle formulated by Hadamard. Moreover, to avoid local minima, a commonly used strategy is to train the ANN a few times with different initial connection weights. Figure 3.6 illustrates this phenomenon: two starting points will lead to a local minimum while three other starting points will lead to the global minimum.

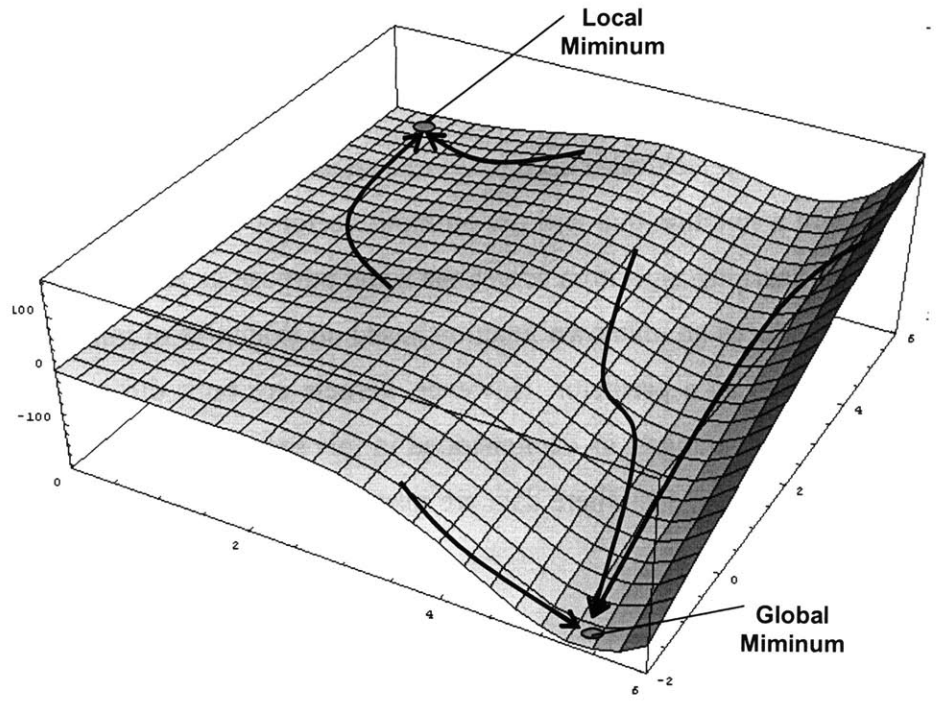

Figure 3.6: Representation of a convergence to a local minimum and a global minimum

3.3 KERNELS

A kernel can be seen as a technique used to do regressions in high dimensions, where its objective is to learn a function that will use an input vector X to predict an output vector Y. In fact, a kernel learning algorithm serves the same purpose as an ANN: it will find a black box that reproduces an unknown function based on examples of inputs-outputs and will thereafter be used as a predictor. However, kernels are built differently and offer many computational advantages, as demonstrated in this section. The kernel, by itself, is a dot product in a feature space where data are mapped by a certain function (Schölkopf and Smola, 2001):

$$\Phi : \aleph \rightarrow \Re^n \tag{3-3}$$

where Φ is the mapping function. Note that in the context of this research, outputs are scalars. Consequently, for simplicity of notations, this section derives the kernel theory assuming scalar outputs ($n = 1$).

A kernel can therefore be written as:

$$k(x_i, x_j) = \left\langle \Phi(x_i), \Phi(x_j) \right\rangle \tag{3-4}$$

The kernel will be defined by the type of mapping function used. Examples of widely used kernels comprise:

- Linear kernel: $k(x_i, x_j) = X_i^T X_j$

- Polynomial kernel: $k(x_i, x_j) = \left(X_i^T X_j + 1\right)^n$

- Gaussian kernel: $k(x_i, x_j) = e^{\frac{-\|X_i - X_j\|^2}{\sigma^2}}$

The term $k(x_i, x_j)$ is frequently written as $K_{i,j}$. From the listed common kernels, one can observe than the kernel matrix has the property to be positive semi-definite and symmetric, which saves computation time as the matrix can be decomposed using Cholesky factorization. In addition, this matrix can be imagined as several measures of vector proximity (Schölkopf and Smola, 2001), which similarities, multiplied by weights, will give an approximation of the output. Hence, any function can be written as (called the representer theorem):

$$f(\cdot) = \sum_{i=1}^{n} c_i k(\cdot, x_i) \tag{3-5}$$

It follows that the objective is to solve for the coefficients c_i which will best fit the function, fitness subjected to smoothness constraints. The next subsection shows how it is done using regularized least squares.

3.3.1 Regularized Least Squares

A regression can usually be achieved by minimizing with respect to a cost function v: $\sum_{i=1}^{n} v(f(X_i), Y_i)$. However, when regularization is not used, the problem becomes directly ill-posed and there is a risk of overfitting data (Rifkin and Lipert (2007)). In order to reinstate the well-posedness of the problem, a regularizer is introduced to the minimization which penalizes terms that are non-smooth or too complex:

$$\frac{\lambda}{2}\|f\|^2 \text{ , called the Tikhonov regularizer} \tag{3-6}$$

where λ is a coefficient setting the required smoothness. This leads to the Tikhonov regularization problem:

$$\inf_{f \in \mathrm{H}} \left\{ \sum_{i=1}^{n} v(f(X_i), Y_i) + \frac{\lambda}{2} \|f\|_k^2 \right\} \tag{3-7}$$

where H is the reproducing kernel Hilbert space (RKHS) and k the kernel function. A RKHS is a space where, in particular, a function k exists such that:

$$\langle f, k(x_i, \cdot) \rangle = f(x) \text{ for all } f \in \mathrm{H} \tag{3-8}$$

and

$$\langle k(x_i, \cdot), k(x_j, \cdot) \rangle = k(x_i, x_j) \tag{3-9}$$

The general solution for the Tikhonov regularization problem is:

$$c = [K + \lambda I]^{-1} Y \tag{3-10}$$

which derivation is shown in Evgeniou *et al.*, 2000.

3.3.2 Kernels Architecture

When using a kernel, it is useful for the user to have prior knowledge of the type of relationship that has to be learned. For instance, a linear kernel will reflect a linear black-box while the Gaussian kernel will give nonlinear decision boundaries in the original data space. Conversely, for the Gaussian kernel, the estimation of an output from given inputs that are far away from the known examples may lead to a high inaccuracy.

Just like ANNs, kernels have to be tuned. The elements of the input vectors may need to be properly scaled to obtain consistent comparisons. Also, the user must select a good value for λ which, in practice, is often established heuristically or by cross-validation. In addition, if one uses a Gaussian kernel, the parameter σ has to be wisely chosen. A very large σ will produce a kernel matrix of all ones while a small σ will give a kernel that looks like the identity matrix.

To show the importance of selection and tuning, a simulation has been done in MATLAB. The simulation consists of a 5 nodes shear beam with assigned arbitrary stiffnesses. 400 random forces, between 0 and 1000 N, have been applied on each node and the real displacement obtained using the inversion of the stiffness matrix. A significant error has been added to the results to show the difference in kernels. Two kernels have been used: a linear and a Gaussian kernel. Figure 3.7 shows the original data as well as both kernels trying to reproduce these data. Figure 3.8 shows the same relationship, but with λ being 10 times smaller.

It can be observed from both figures that the Gaussian kernel is close to the obtained values while the linear kernel reproduces a plane in the 3D space. Also, an increase in λ does not significantly affect much the linear kernel but has a high impact on the smoothness of the Gaussian kernel. Hence, in this situation, if the user knew the linear relationship of the system and the occurrence of some noise, the linear kernel would have been a wise choice. However, if that relationship was unknown but the user was still aware of the noise, a Gaussian kernel with a λ of the same magnitude as the input is a conservative procedure for a good smoothness. Otherwise, if it is known that no noise exists, a Gaussian with a small λ would better fit the known data.

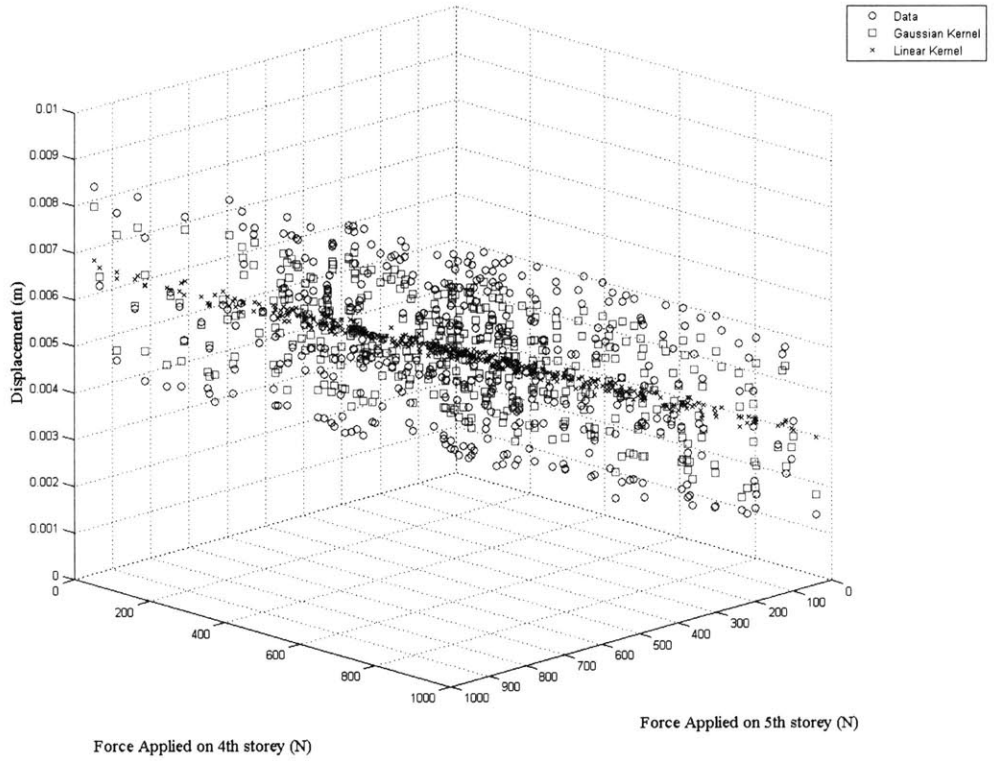

Figure 3.7: Displacement of the 5th storey, $\lambda = 0.1$

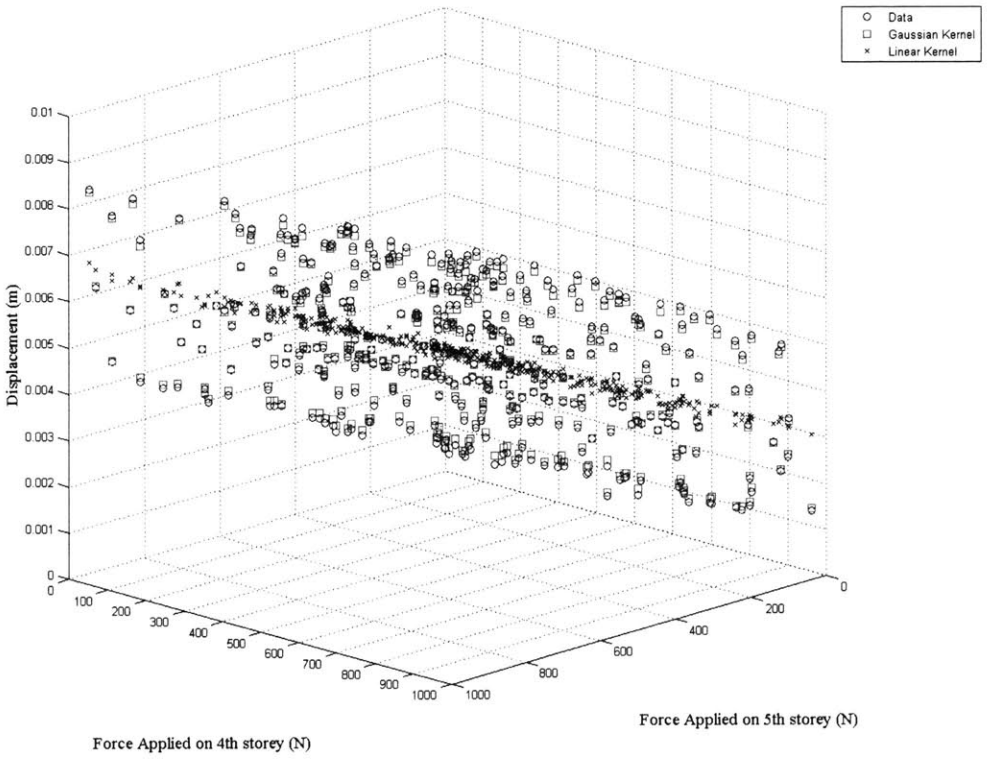

Figure 3.8: Displacement of the 5th storey, $\lambda = 0.01$

4.0 PROPOSED ALGORITHM

4.1 PROBLEM

One of the main difficulties for semi-active and active control arises from the non-linear dynamic behaviour of structures and mutual dependency of the plant state and the control force, represented in figure 4.1.

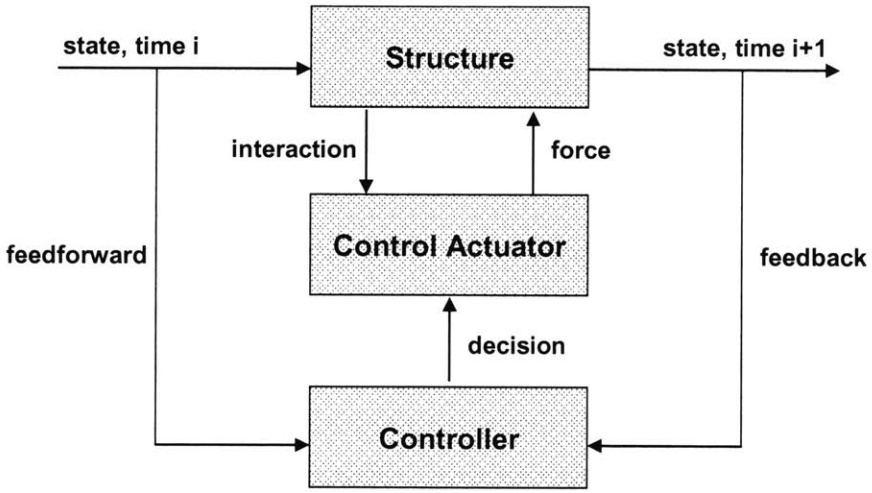

Figure 4.1: Interaction diagram for an active/semi-active control scheme (adapted from Dyke et al., 1995)

In addition, to control MR actuators, most of the suggested control schemes do not model the actuator's response delay, and it is known that a lag in the actuator response can lead to a significant inefficiency in control and can even lead to the instability of the system. It goes without saying that MR actuators are hard to model which difficulty arises from their rheological fluid properties. Many attempts have been made to properly model the actuator, but the stiction phenomena as well as the shear tinning effect (section 2.1) still cause estimation issues.

Also, control schemes assume full knowledge of the structure state and properties. It means that the dynamic states (accelerations, velocities and displacements) of all degrees of freedom are known, and that the masses, damping and stiffnesses are appropriately

estimated. In real situations, accelerations and displacements data can be obtained with the use of accelerometers and strain gauges (provided that the strain gauge is located in a bracing) respectively. Conversely, velocities are unknown and can only be estimated through the integration of the acceleration which may lead to an inaccurate estimator. Regarding the system properties, they can be properly estimated for normal geometries, but they may be inaccurate for complicated ones.

For these reasons, it is attempted here to propose a control scheme where only the accelerations and displacements are known at a specific time. The idea of using machine learning is to have the structure "learn" its behaviour upon some excitations. Hence, modeling rheological properties would not be necessary as the system can "know" the reaction of the plant with respect to the input of a certain voltage.

Some attempts have been made using machine learning strategies, such as artificial neural networks and kernels, but they have the weakness of using offline learning. Offline learning is feasible when one thinks of computer simulations and small scale models. However, for large scale structures, the structural behaviour upon an earthquake is unknown until a first earthquake happens, whereas the control scheme will be untrained and inefficient. It is possible, however, to build a similar plant and run destructive tests in order to train the network, which involves tremendous economic expenses.

Therefore, the proposed scheme will use online learning. Online learning, when considering earthquakes, is difficult to achieve as earthquakes usually strike with high intensities in a very short period of time whereas the algorithm must perform well in order to prevent structural damages. The following subsection describes the proposed algorithm and the next section assesses its performances.

4.2 ALGORITHM

4.2.1 Selecting Forces

Online learning can be achieved through the use of many machine learning techniques such as artificial neural networks and kernels. For the addressed problem, kernels are preferred over ANNs for practicality. ANNs can be used accurately for linear continuous systems provided that the structure is properly discretized, but it is not necessarily the case for nonlinear structures (Sanner and Slotine, 1992). Moreover, the gradient descent method gives no convergence guarantees, as discussed previously, as the ANN would necessitate proper online tuning of its parameters, such as the number of hidden layers, nodes and connections.

Classical control can be seen as the application of a force upon a certain state of the structure:

$$U_{t+1} = Cx + Df + \varepsilon \tag{4-1}$$

where U_{t+1} is the output vector, C and D are coefficient matrices, x is the state of the structure, f is the applied force and ϵ is the noise. The state x comes from the state-space representation of the equation of motion expressed previously (equation 2-3) and rewritten here for convenience:

$$\dot{x} = Ax + B_f f + B_g \ddot{u}_g \tag{4-2}$$

The idea of the control scheme is to have the kernel learning algorithm to estimate the state of the structure at time t and predict a force based on this state. The state is estimated using the following data:

$$X_t = \begin{Bmatrix} u_{i,t-j} \\ u_{i,t} - u_{i,t-1} \\ \ddot{u}_{e,t-k} \end{Bmatrix}_{3xDOF+3} \tag{4-3}$$

with the associated applied forces:

$$Y_t = \{F_{i,t}\}$$

for i = 1 to the number of degrees of freedom; j = 0,1, k =0,1,2, $\ddot{u}_e$ is the acceleration of the node where the force is applied.

The elements of the X matrix are scaled such that all the elements are of about the same order of magnitude, which will facilitate tuning of the kernel. This scaling rule is rationalized by an equal importance of each input element. As the two first groups of terms of the input vector are generally of the same order of magnitude, only the acceleration has to be scaled which can be done approximately.

The mapping function of the kernel is a Gaussian function as the structural behaviour is known to be non-linear:

$$K_{i,j} = e^{\frac{-\|X_i - X_j\|}{\sigma^2}} \tag{4-4}$$

where i,j = 1 to number of observations, σ is chosen appropriately in accordance with the magnitude of the inputs in order to obtain a good distribution of the values in the kernel matrix. A good σ would give an exponent of the order of $\sim 10^{-1}$ to 10^{-2}.

As discussed previously, the solution of the Kernel is

$$c = [K - \lambda I]^{-1} Y \tag{4-5}$$

In this case, a wisely chose λ would be such that it gives good, but not too sensitive, results. Consequently, a correct λ would be of the same order of magnitude as the inputs.

The issue here is that the kernel has only one null force as the output example at time 1 and will therefore always output a force F = 0 for any state, unless the forces are adapted with some learning rules whereas the algorithm is not expected to converge quickly enough to give a descent performance. In order to solve this issue, a synthetic observation, here termed upper bound, is introduced among the set of real observations. This upper bound uses input values that would necessitate an optimal force:

$$X_{t+1} = \begin{Bmatrix} u_{i,t-j}\big|_{up} \\ \left(u_{i,t} - u_{i,t-1}\right)\big|_{up} \\ \ddot{u}_{1,t}\big|_{up} \end{Bmatrix} \qquad Y_{t+1} = \left\{F_{\max}\right\} \tag{4-6}$$

where the upper bounds are user-determined and, to make sense, have to be large but do not necessarily have to be larger than the maximum values (as they are considered as unknowns), and F_{max} is a physical property of the MR actuator used. It will be shown later that the upper bound inputs can be any values, as long as F_{max} is used. The upper bound will not provoke inaccuracy if ever it falls outside the trends of data because of the well-posedness of the problem and the λ parameter. Additionally, the influence of the synthetic observation is limited in a high dimension space as it represents only one specific point of the space.

It follows that the kernel learning algorithm will be able to determine the Euclidian distance between two states and derive an output that will be included between 0 and F_{max} (as F_{max} is invariably set as the maximum allowable force) for its first and subsequent approximations.

Using kernels comes at the expense of computation time. To be consistent, running the algorithm should take less time than the sampling time step itself. Dealing with a 50x50

kernel matrix is close to this limit (for a code not written by an expert in computer programming). For this reason, a "running" kernel is introduced. This kernel will only take the last 50 entries of the system, except for the first 50 time steps where it will use all of them, plus the upper bound. This also increases the controller accuracy as the kernel only keeps the latest updated results.

It is important to point out that in a real experiment, the voltage would be selected rather than the force, which would prevent the user from evaluating the needed voltage with respect to the prescribed force. For simplicity, only the reaction of the actuator with respect to the voltage, which is the force, has been modeled.

4.2.2 Applying Forces

Once a force is selected by the kernel learning algorithm, it is applied to the system following some physical rules. A distinction has to be made between the targeted DOF u_c, which is what is meant to be globally controlled, and the u_e, the DOF where the force is applied. It must be said that the decision whether to apply a force or not will be made with respect to u_e rather than u_c[1]. The reason is that it is preferable to directly control for this DOF because the earthquake can excite all modes of the structure. Hence, a 3 DOF shear beam could find its second and third modes more excited than the first one. Trying to directly control for the top node while having the actuator at the first node might be problematic in this case as the direction of displacements and velocities of these two nodes might not coincide. In the algorithm, the targeted DOF u_c will be passively controlled by serving as the performance index.

[1] This statement has been briefly tested in the simulation and does give better results than trying to control directly for the targeted node. This could be investigated in further works.

The force application rules are as follows:

- Apply the force if the acceleration goes in the same direction as the velocity.
- This force is opposite to the direction of the degree of freedom displacement where the force is applied, u_e.
- For simulation purposes, the force cannot be larger than the mass of the building times the acceleration of the degree of freedom where the force is applied (specific to a MR actuator since it can only resist a force).
- Apply the force only if the structure is found to be in an earthquake state (which could be evaluated with prior accelerations).

4.2.3 Adapting Forces

Once the force has been applied, the learning algorithm has to evaluate its performance and adjust the predicted force with a better value that could have been predicted in order to increase the accuracy of the controller.

The learning rule follows that the prediction will be updated only if a force has been applied. If this is the case and the acceleration of the controlled DOF has increased, the prediction will be reduced using the following rule:

$$F_{prediction,t} = F_{prediction,t} \cdot \left(1 - \frac{\left|u_{c,t+1}\right|}{\eta}\right) \tag{4.7}$$

where η is the learning rate, u_c the targeted DOF displacement. The learning rate may be small and increase if the performance, here the absolute displacement of the targeted DOF, comes close to its value. This will ensure that the predictor does not fall into a

negative value or above F_{max}. Conversely, if the acceleration has decreased, the predictor will be augmented using the following rule:

$$F_{prediction,t} = F_{prediction,t} + \left(1 - \frac{F_{prediction}}{F_{\max}}\right) \cdot F_{\max} \cdot \frac{\left|u_{c,t+1}\right|}{\eta} \qquad (4\text{-}8)$$

4.2.4 Algorithm Diagram

Figure 4.2 shows the algorithm in the form of a diagram.

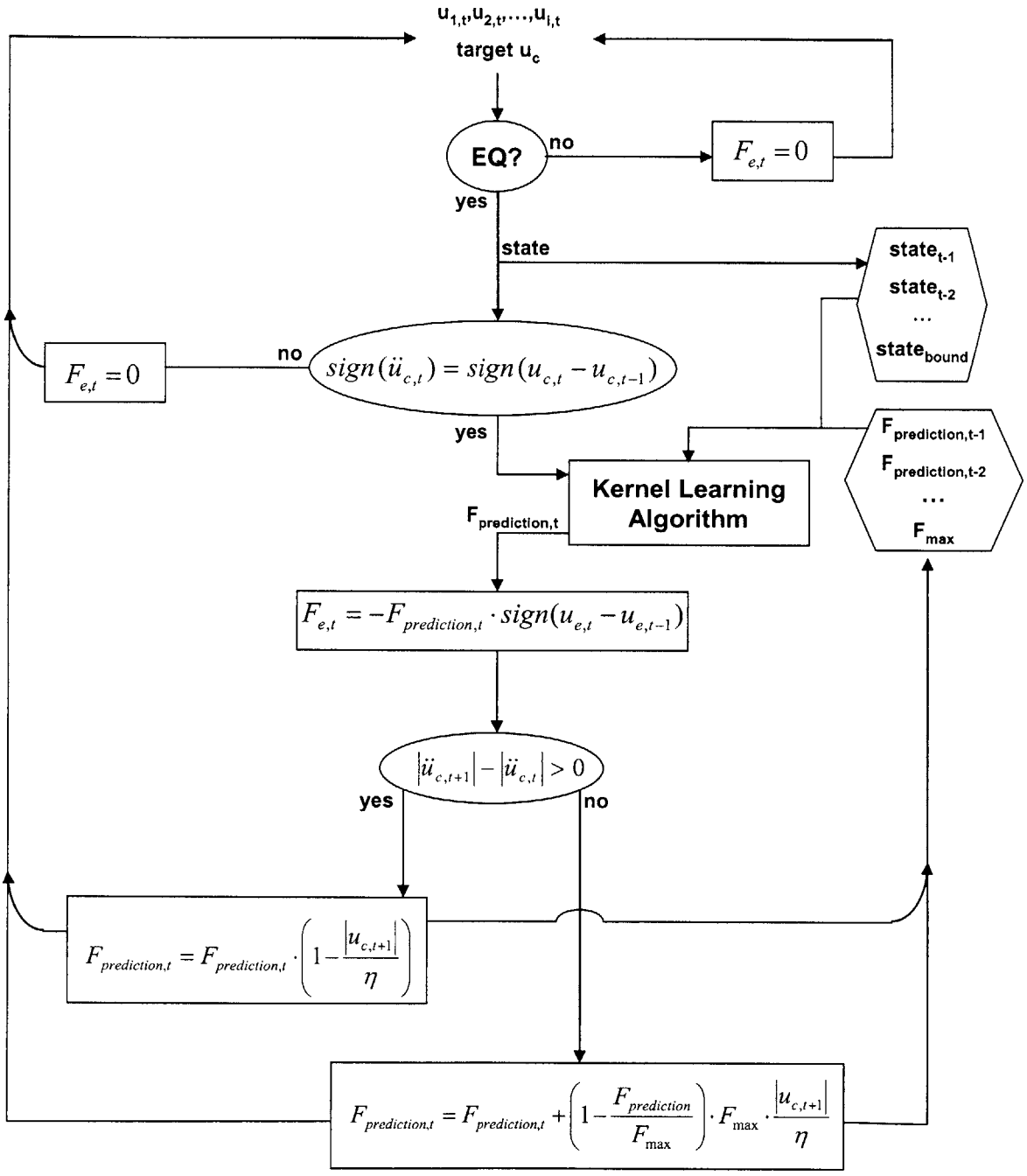

Figure 4.2: Algorithm diagram (relative to time *t*)

5.0 SIMULATION

5.1 METHODOLOGY

In order to assess its efficiency, the algorithm has been compared to the experiment made by Dyke *et al.,* 1996. In this well known experiment, a MR actuator is used to control a 3 storey scaled building and is controlled by a clipped-optimal control strategy. Figure 5.1 shows the setup for the experiment.

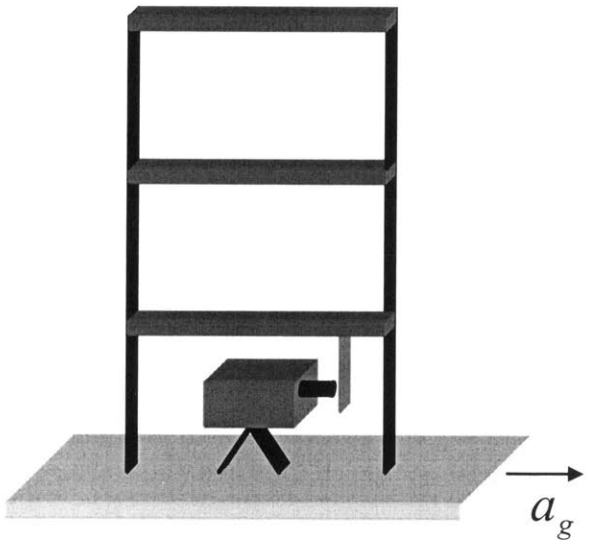

Figure 5.1: Experiment setup (adapted from Dyke *et al.*, 1996)

Which structure has the following properties:

$$M = \begin{bmatrix} 98.3 & 0 & 0 \\ 0 & 98.3 & 0 \\ 0 & 0 & 98.3 \end{bmatrix} kg \; ; \qquad C = \begin{bmatrix} 175 & -50 & 0 \\ -50 & 100 & -50 \\ 0 & -50 & 50 \end{bmatrix} \frac{N \cdot s}{m} \; ;$$

$$K = 10^5 \begin{bmatrix} 12.0 & -6.84 & 0 \\ -6.84 & 13.7 & -6.84 \\ 0 & -6.84 & 6.84 \end{bmatrix} \frac{N}{m}$$

These properties have been implemented in a MATLAB code and the response has been obtained using the general solution for a time invariant MDOF system as described previously (equation 2-5) and rewritten here for convenience (setting $P = 0$).

$$X_{j+1} = e^{A\Delta t} X_j + A^{-1}\left(e^{A\Delta t} - I\right)\left[B_g a_{g,j} + B_f F_j\right] \tag{4-9}$$

where for this specific case F_j is the applied force defined from the predicted force.

The algorithm has been tuned using the following coefficients:

- $\ddot{u}$ scaled with a factor of 1000 in order to obtain a magnitude $\sim 10^{-3}$ for inputs
- $\sigma = 0.1$, following the selection rule explained previously
- $\lambda = 0.002$, following the selection rule explained previously
- $\eta = 0.001$ m, and is multiplied by 10 if $|\tilde{u}_c| > \eta/2$
- $F_{max} = 1000$ N, to be consistent with the experiment
- $X_{t+1} = \begin{Bmatrix} 0.01 \\ 0.001 \\ 0.005 \end{Bmatrix}$, arbitrarily and consistent with magnitudes

The MATLAB code used for the simulation is presented in appendix A.

5.2 RESULTS

The following subsections will show the performances of the algorithm for the three-storey building described previously.

5.2.1 Algorithm versus Classical Control (Extreme Scenario)

The El Centro earthquake has been simulated to be consistent with the experiment and scaled by a factor of 5 to be consistent with the structure size. El Centro is an impulsive earthquake. Therefore, it is an ultimate test to assess the performance of online learning as the building must "learn" its behaviour quickly.

Figure 5.2 shows the performance for storey displacements and compares with the third storey displacement obtained by Dyke *et al.* (figure 5.3[2]) while figure 5.4 shows the acceleration of each storey and compares the last storey acceleration with the control obtained by Dyke *et al.*. (figure 5.5[2]), over 5 seconds (equivalent to 25 seconds for large scale). In this subsection, the output forces are not compared since the algorithm prescribes point forces (bang-bang controller) while the clipped-optimal strategy uses continuous forces.

The results obtained from the simulation show a good performance of the algorithm. However, it does not perform as well as the clipped-optimal control for the displacements while this relationship is inverted for the accelerations of the two last storeys. Table 5.1 summarizes the optimal values to compare the simulation and the experiment. Note that with the algorithm there is no direct attempt to control for acceleration.

[2] there is a lag of ~0.5 seconds in the earthquake time in the experiment by Dyke *et al.* with respect to this simulation

DOF	Uncontrolled	Algorithm	Clipped-Optimal (Dyke et al., 1996)	Peak Reduction Algorithm (%)	Peak Reduction Clipped-Optimal (%)
u_1(cm)	0.55	0.182	0.114	66.9	79.3
u_2(cm)	0.837	0.264	0.185	68.5	77.9
u_3(cm)	0.974	0.318	0.212	67.4	78.2
$\ddot{u}_1$ (g)	1.06	1.26	0.71	-18.9	33.0
$\ddot{u}_2$ (g)	1.12	0.645	0.753	42.4	32.8
$\ddot{u}_3$ (g)	1.51	0.681	0.717	54.9	52.5

Table 5.1: Comparison of optimal values – classical control case

Conversely, the relative performance of the algorithm with respect to the experiment would be expected to be significantly better for a real structure because of all the unknowns and difficulties explained in the previous section. Additionally, the proposed algorithm is not sensitive to noise as it will adapt according to the entire system behaviour.

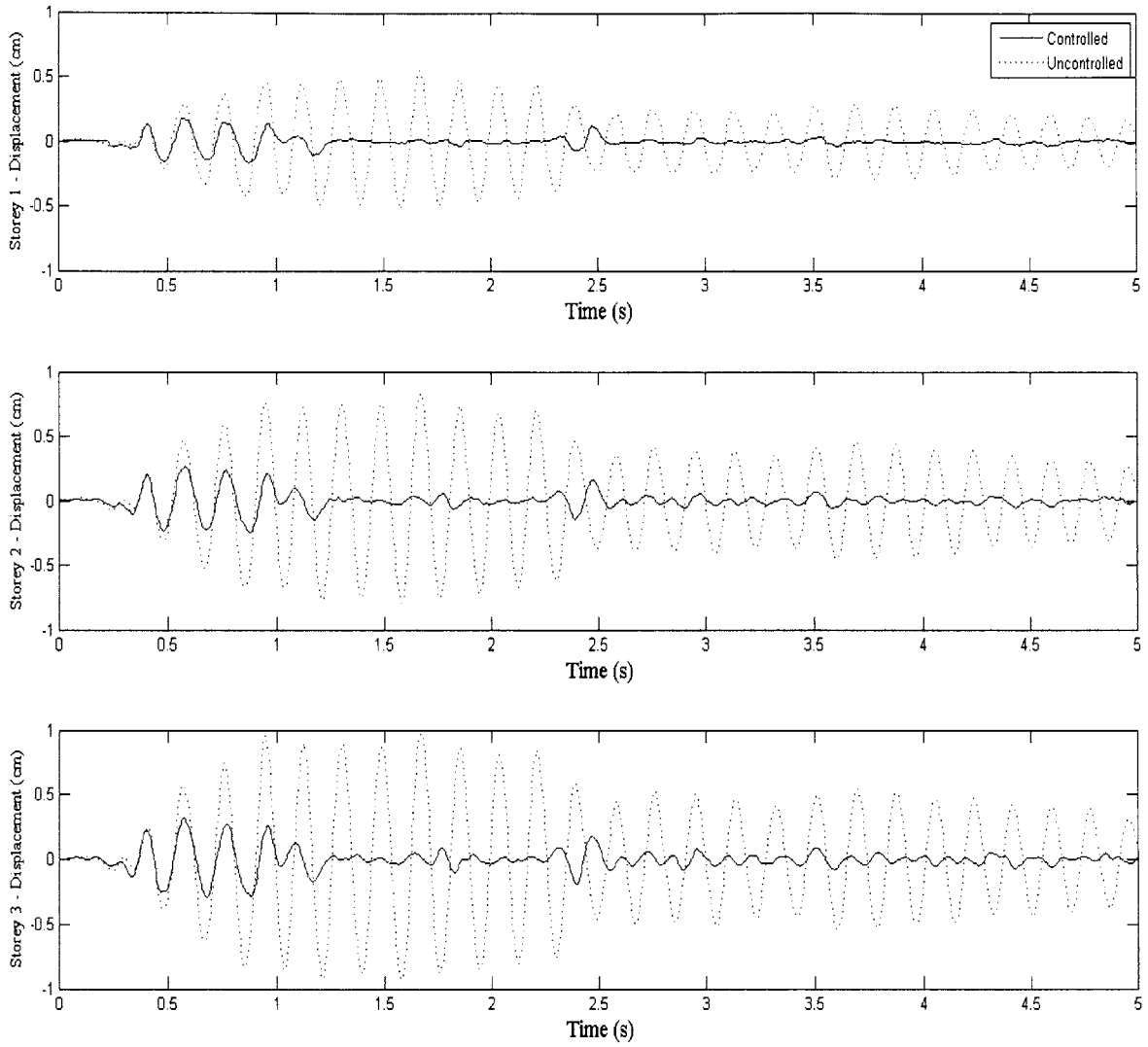

Figure 5.2: Storey displacements over 5 seconds, El Centro earthquake

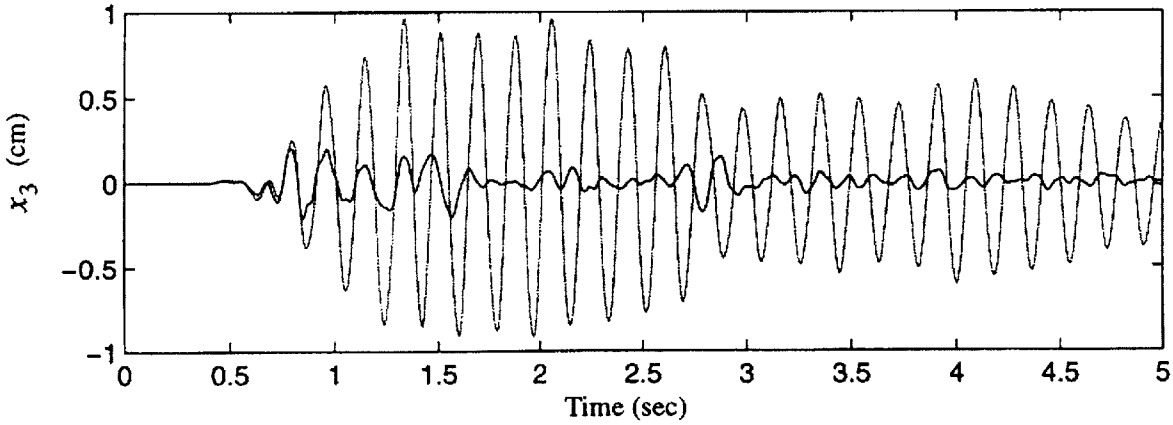

Figure 5.3: Dyke *et al.* (1996) results, third storey displacement

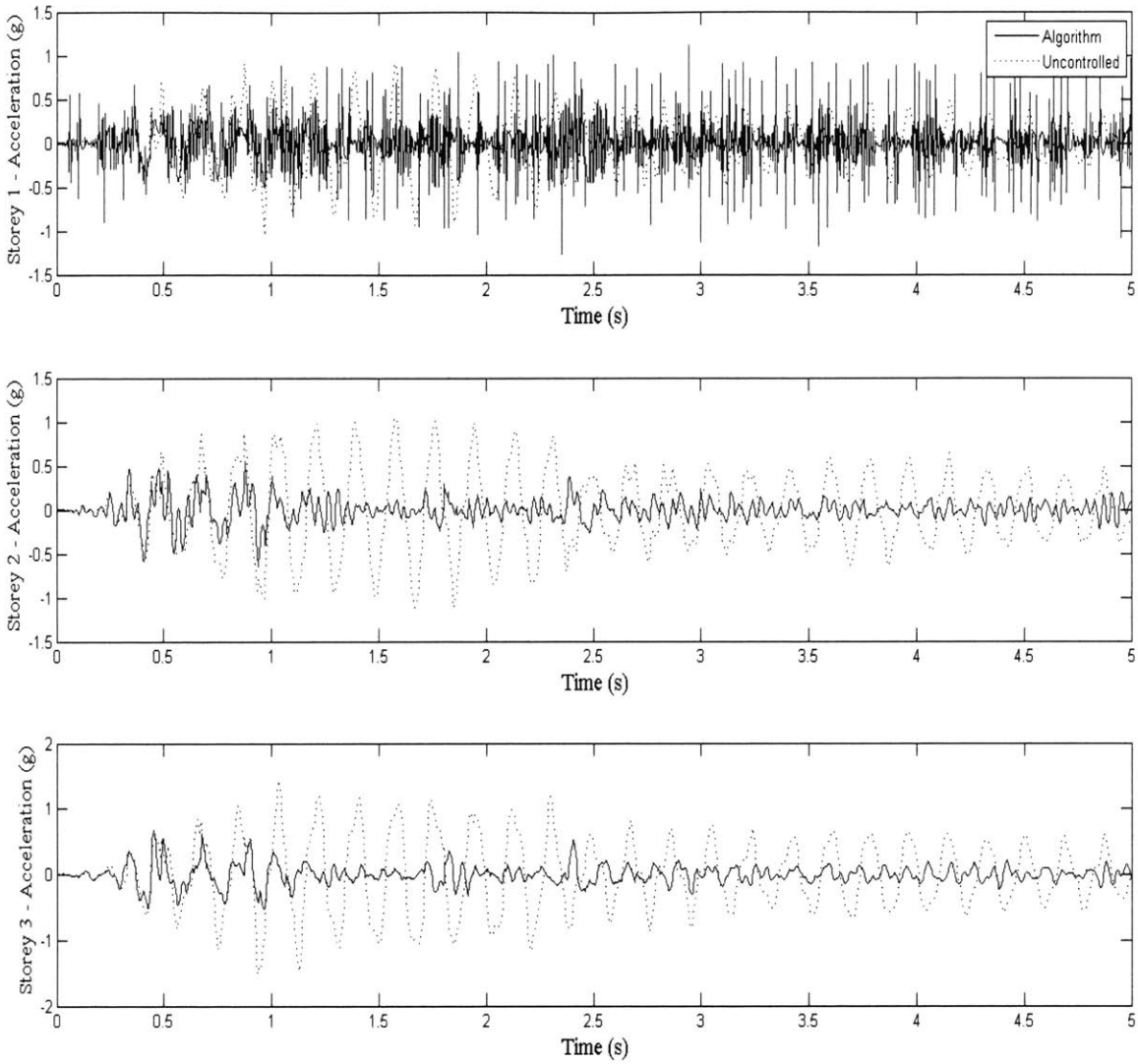

Figure 5.4: Storey accelerations over 5 seconds, El Centro earthquake

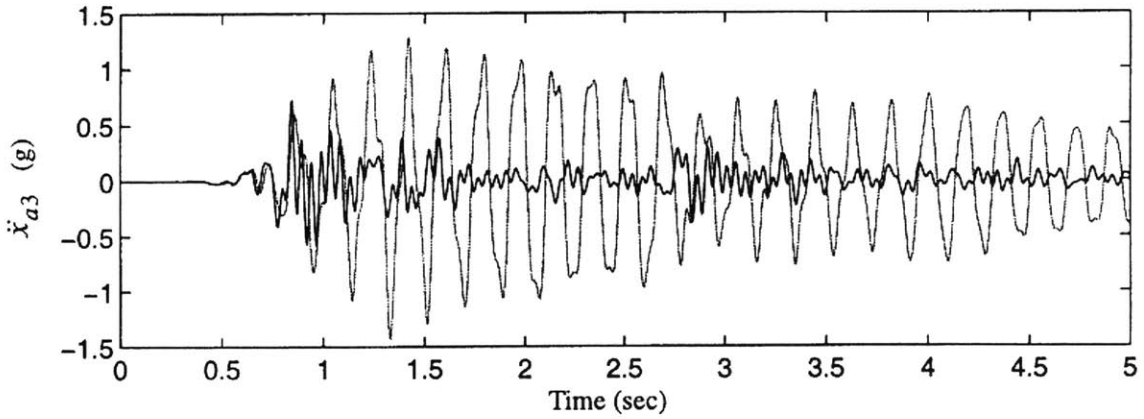

Figure 5.5: Dyke *et al.* (1996) results, third storey acceleration

5.2.2 Algorithm versus Passive

In what follows, the algorithm will be compared with two passive-on strategies, when the current is set to a constant value to provide a constant force reaction. The tested constant forces are 1000 N, the maximum force, and its half, 500 N.

Figures 5.6 to 5.9 compare results with the passive cases using the El Centro Earthquake. Table 5.2 summarizes the optimal values. When comparing the algorithm with the passive-on 1000 N case, displacement mitigation is quite similar (67.4% reduction of the third storey displacement obtained with the algorithm and 64.5% with passive-on 1000 N), but the passive-on requires 44.2% more energy for such performance. The passive-on 500 N case returns worse performances on displacements.

Regarding acceleration, since all strategies do not directly attempt to control for these degrees of freedom, it is not easy to compare them together. However, it can be observed that the passive-on 1000 N case exhibits significantly higher accelerations at all levels while the passive-on 500 N does significantly better at controlling the acceleration at the first floor.

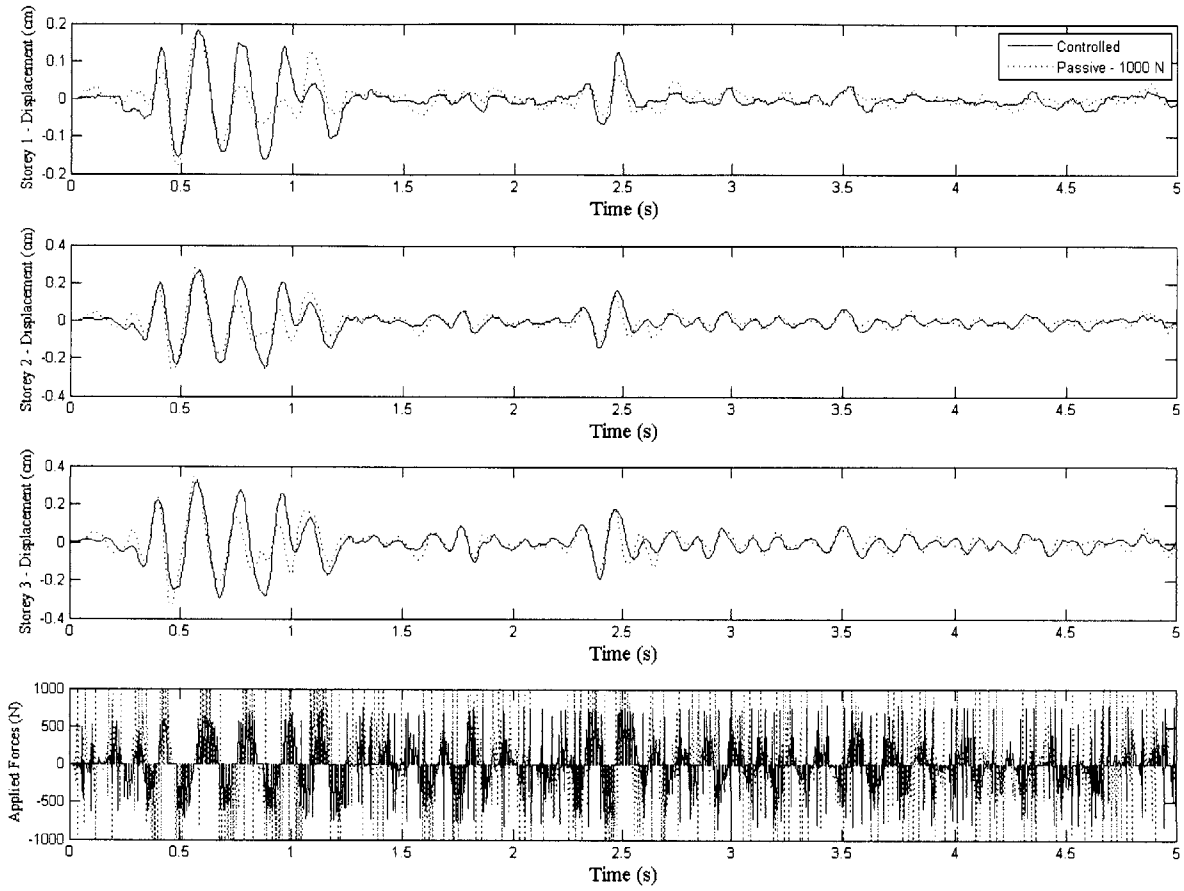

Figure 5.6: Comparison of storey displacements and applied forces - passive 1000 N

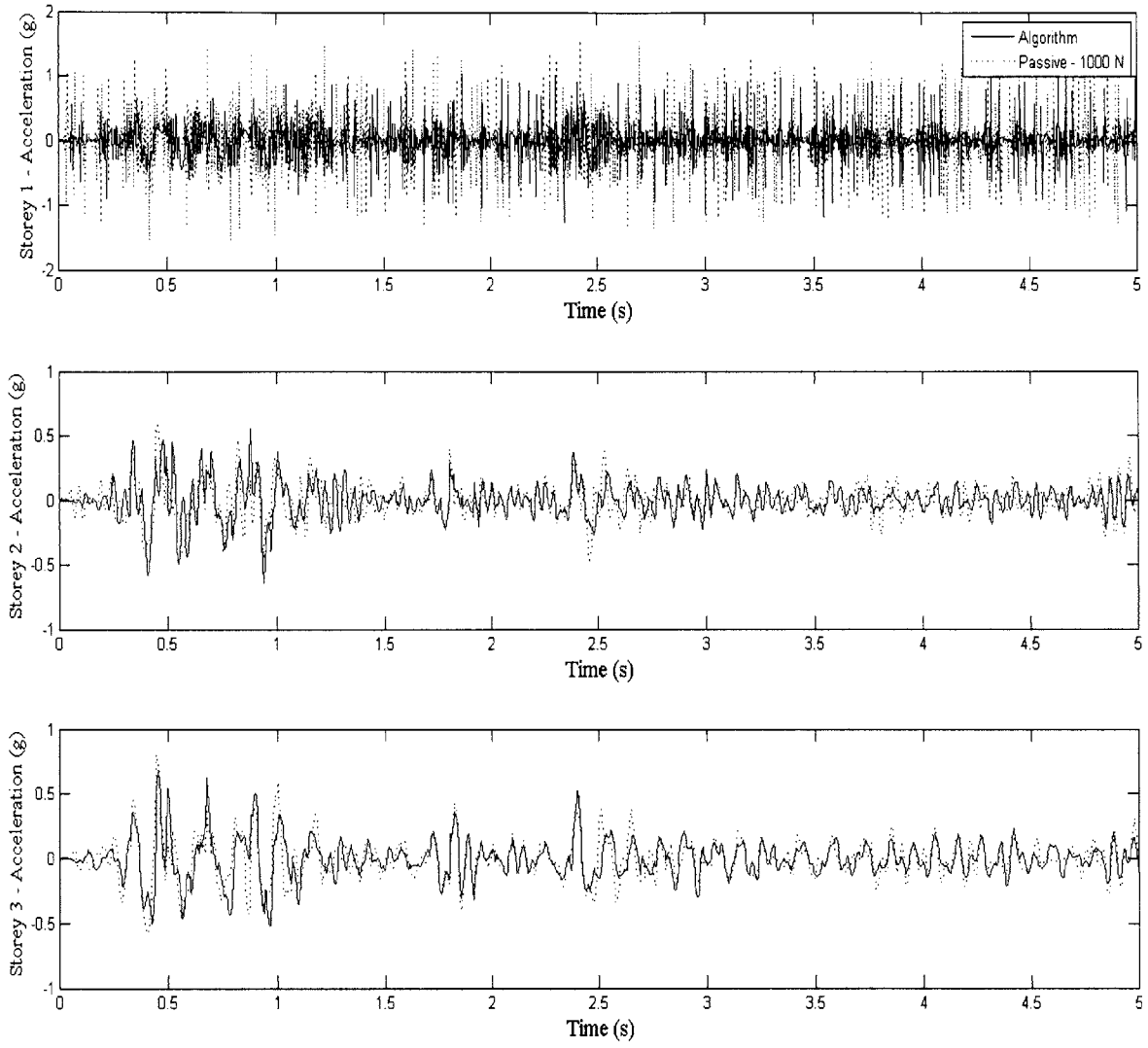

Figure 5.7: Comparison of storey accelerations - passive 1000 N

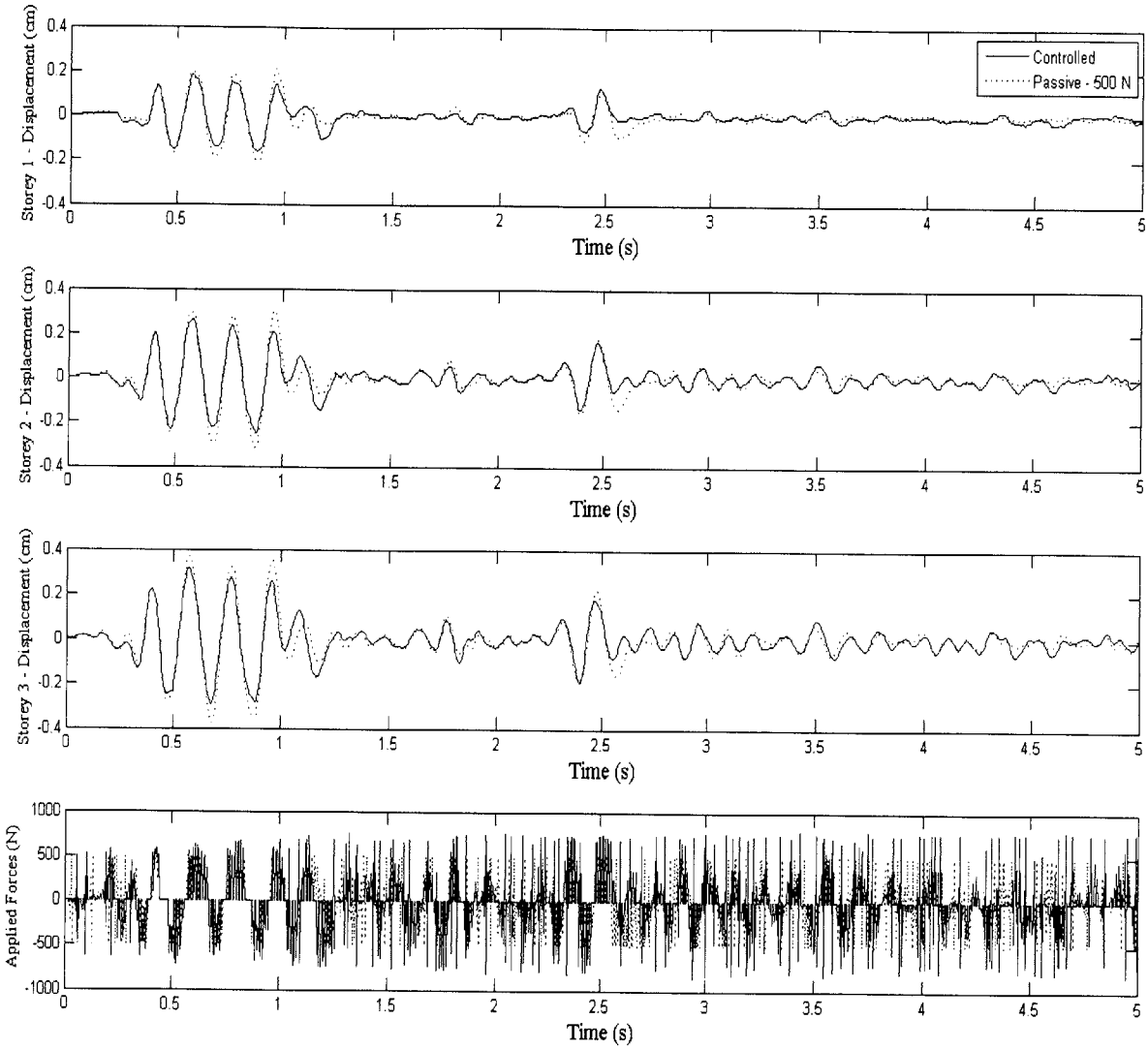

Figure 5.8: Comparison of storey displacements and applied forces - passive 500 N

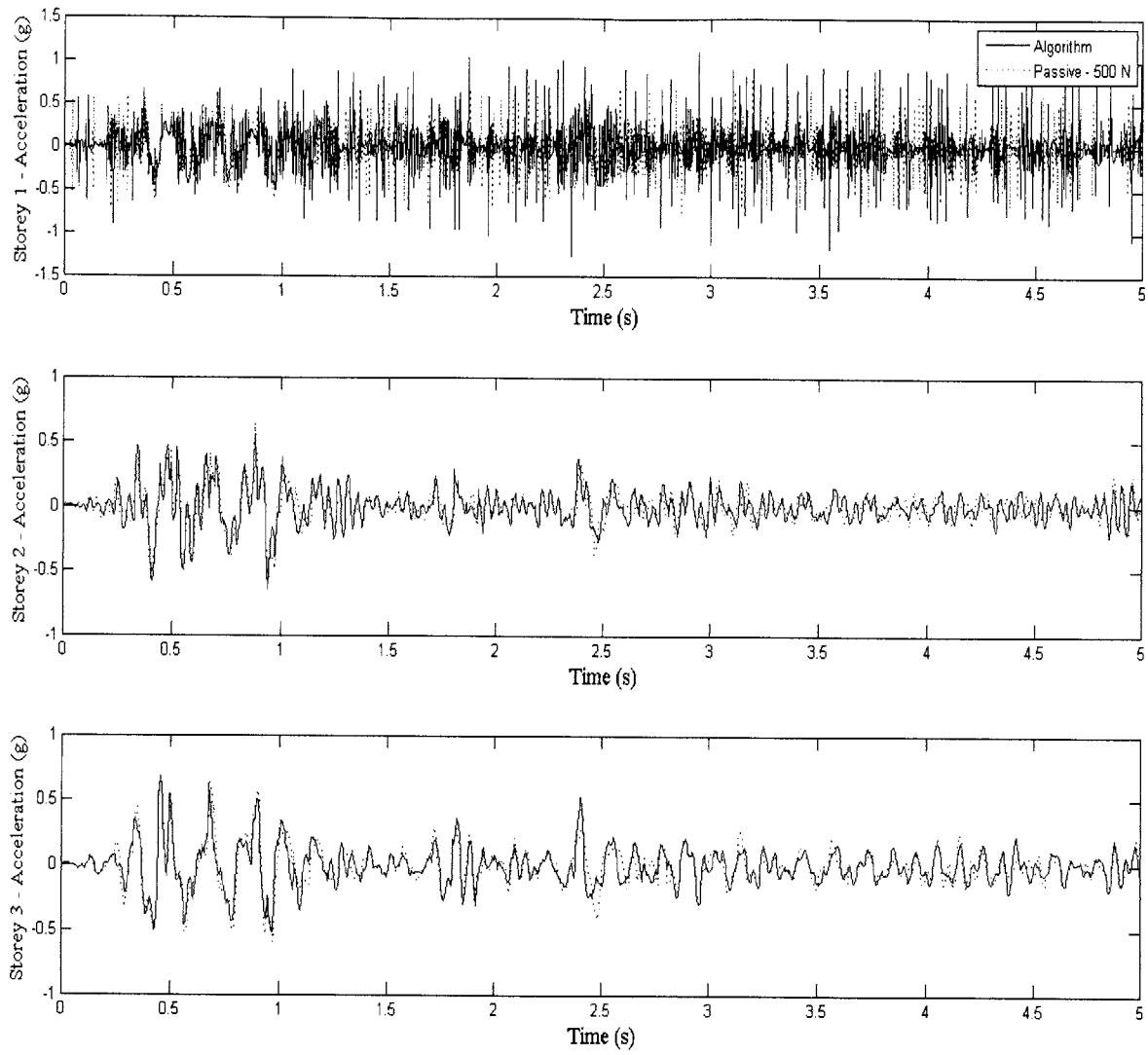

Figure 5.9: Comparison of storey accelerations - passive 500 N

DOF	Uncontrolled	Algorithm	Passive (1000 N)	Passive (500 N)
u_1 (cm)	0.55	0.182	0.176	0.210
u_2 (cm)	0.837	0.264	0.288	0.315
u_3 (cm)	0.974	0.318	0.346	0.376
$\ddot{u}_1$ (g)	1.06	1.26	1.571	0.811
$\ddot{u}_2$ (g)	1.12	0.645	0.581	0.637
$\ddot{u}_3$ (g)	1.51	0.681	0.806	0.662
Total Applied Forces (kN)*	0	267	385	222

* over 5 seconds

DOF	Peak Reduction Algorithm (%)	Peak Reduction Passive-1000N (%)	Peak Reduction Passive-500N (%)
u_1 (cm)	66.9	68.0	61.8
u_2 (cm)	68.5	65.6	62.4
u_3 (cm)	67.4	64.5	61.4
$\ddot{u}_1$ (g)	-18.9	-48.2	23.5
$\ddot{u}_2$ (g)	42.4	48.1	43.1
$\ddot{u}_3$ (g)	54.9	46.6	56.2

Table 5.2: Comparison of optimal values – passive cases

5.2.3 Algorithm for Different Earthquakes

This subsection compares the algorithm performance for five earthquakes. It is primordial for the algorithm to perform well over different kinds of earthquake because it is not tuned for any specific one. A good online learning algorithm is expected to be capable to adapt any external situation and succeeding in this task partly assesses its stability.

The five tested earthquakes are summarized in table 5.3 and their performances shown from figure 5.10 to figure 5.14. The time scale represents the scaled earthquake which would be 5 times longer for large scale structures. A comparison of the performances is shown in table 5.4.

Eartquake	Description	Max Acceleration (g)	Max Velocity (cm/s)	Max Displacement (cm)	Duration (seconds)
Imperial Valley	Imperial Valley, El Centro May 18, 1940, 180 degrees	0.296	23.6	13.3	39.99
Mexico City	Mexico City, Station 1 September 19, 1985, 180 degrees	0.161	57.4	21.9	180.1
Northridge	Northridge, Santa Monica, City Hall Grounds January 17, 1994, 90 degrees	0.753	41.7	8.88	39.98
San Fernando	San Fernando, Pocoima Dam February 9, 1971, 254 degrees	0.849	54.3	11.7	41.63
Kern County	Kern County, Taft Lincoln School July 21, 1952, 111 degrees	0.152	8.99	8.99	54.15

Table 5.3: Earthquakes summary

Earthquake	Imperial Valley	Mexico City	Northridge	San Fernando	Kern County
Peak Displacement Uncontrolled (cm)	0.974	0.347	0.482	1.92	0.542
Peak Reduction Algorithm (cm)	0.318	0.184	0.232	1.25	0.100
Reduced Peak Displacement Algorithm (%)	67.4	47.0	51.9	34.9	81.5

Table 5.4: Summary of results

The results show significant performances for the algorithm. Figure 5.11 illustrates an interesting feature of the controller. It can be observed that the algorithm continues to output large forces for low displacements and velocities. This is due to the fact that the algorithm is ran for a longer time (35 seconds) where learning can no longer take place since the controlled displacements are close to zero. It results that the controller must be turned off after the earthquake (rationalizing the force application rule about the earthquake state) and that the algorithm has to be reset to null values after the earthquake.

However, since these performances are significantly different for all earthquakes, the algorithm must be compared with the passive-on case to ensure that these differences are due to the earthquake types (clustered accelerations, early high peak...) and properties (maximum ground acceleration, maximum ground displacement...). This comparison is done in section 5.2.5. The next subsection investigates the sensitivities of the tuning parameters to address the algorithm stability.

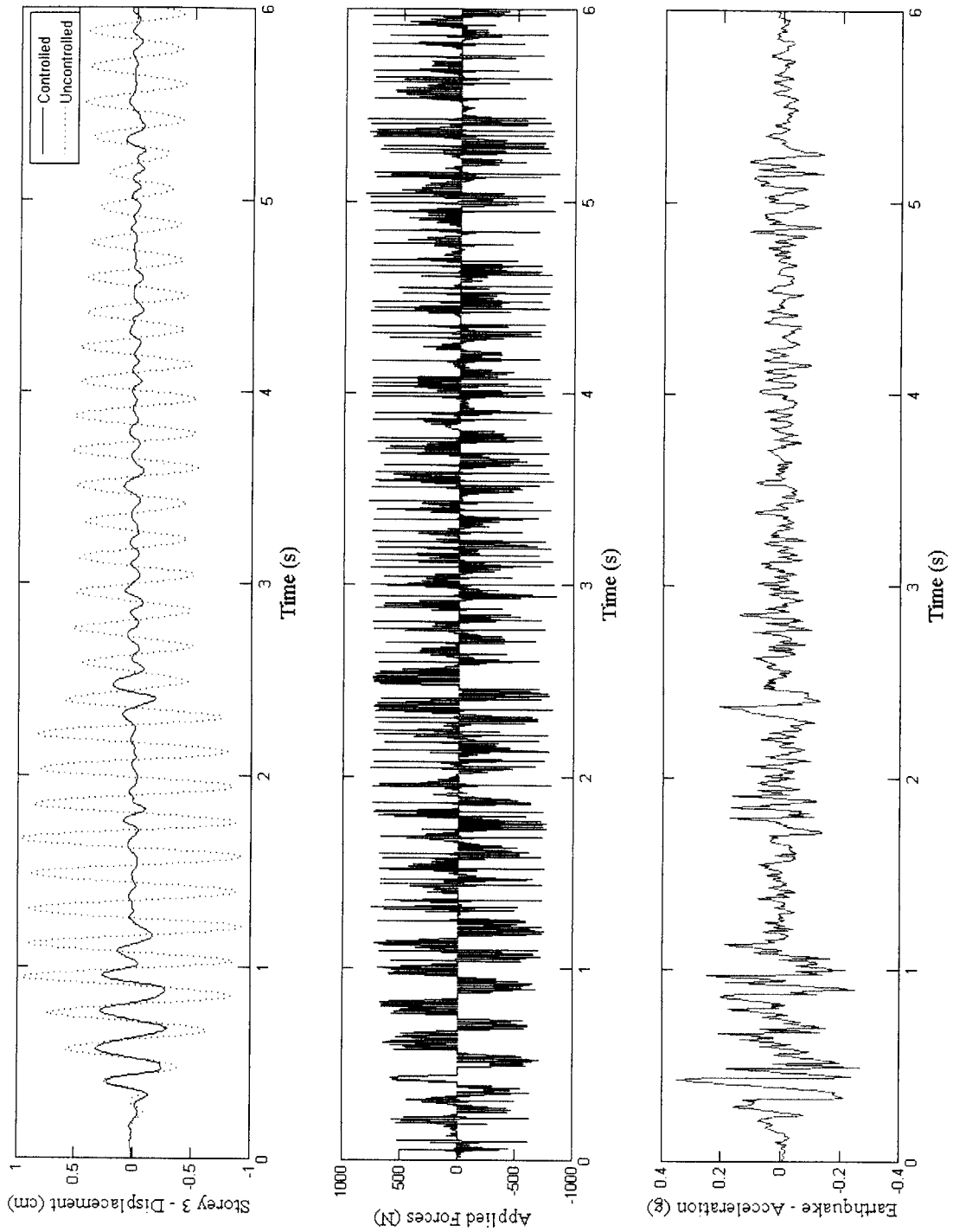

Figure 5.10: Performance for Imperial Valley earthquake (El Centro)

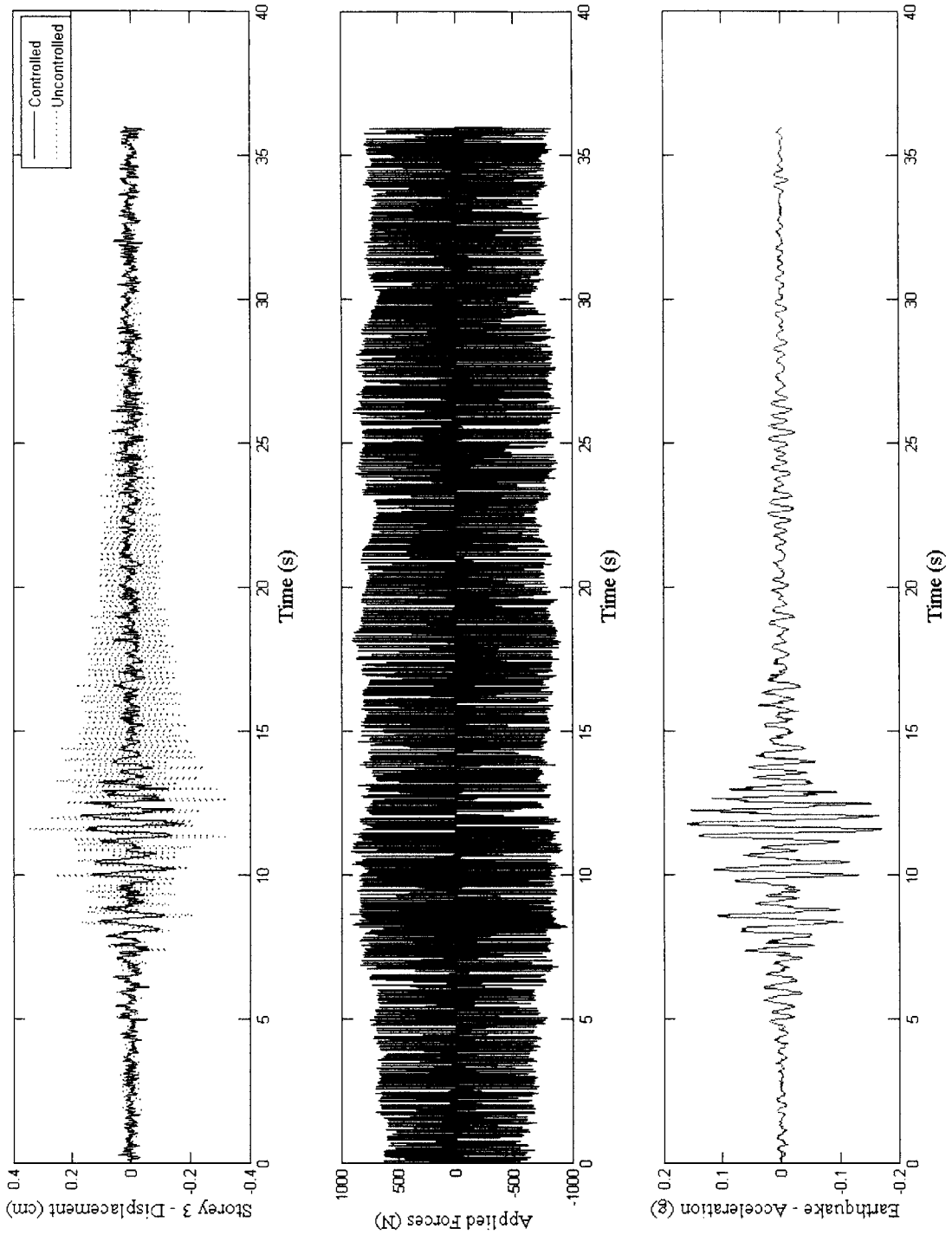

Figure 5.11: Performance for Mexico City earthquake

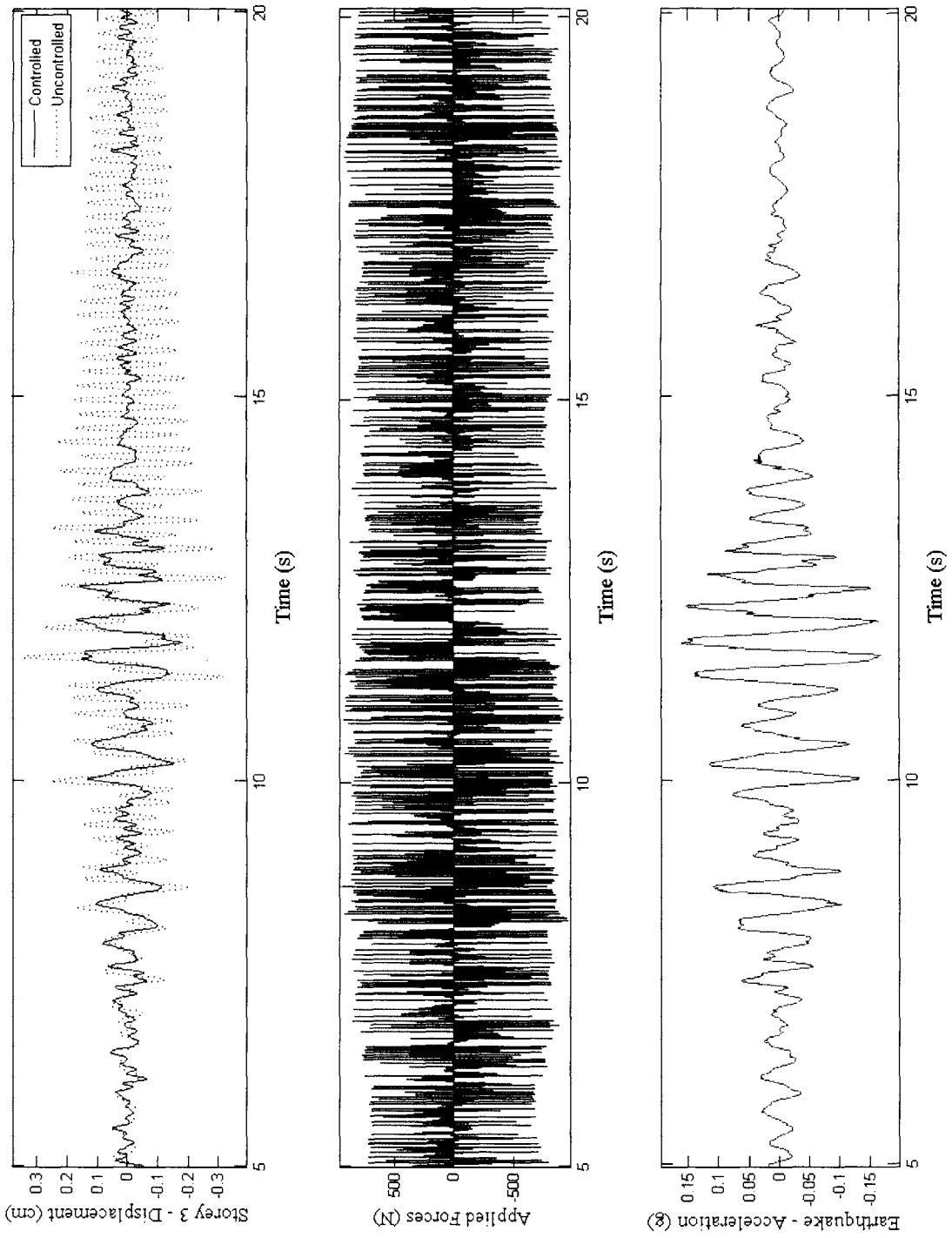

Figure 5.11a: Performance for Mexico City earthquake – Zoom over main earthquake excitation

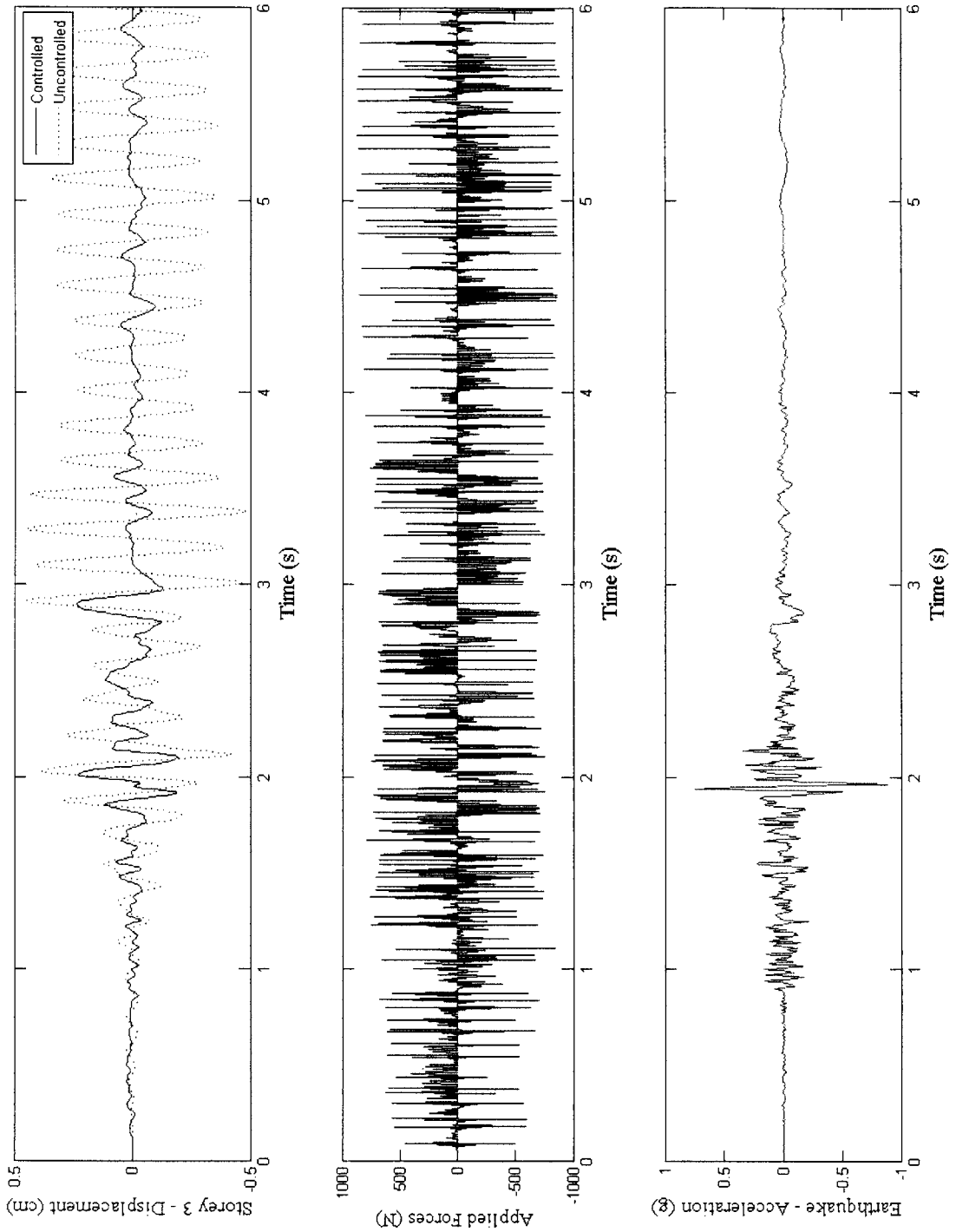

Figure 5.12: Performance for Northridge earthquake

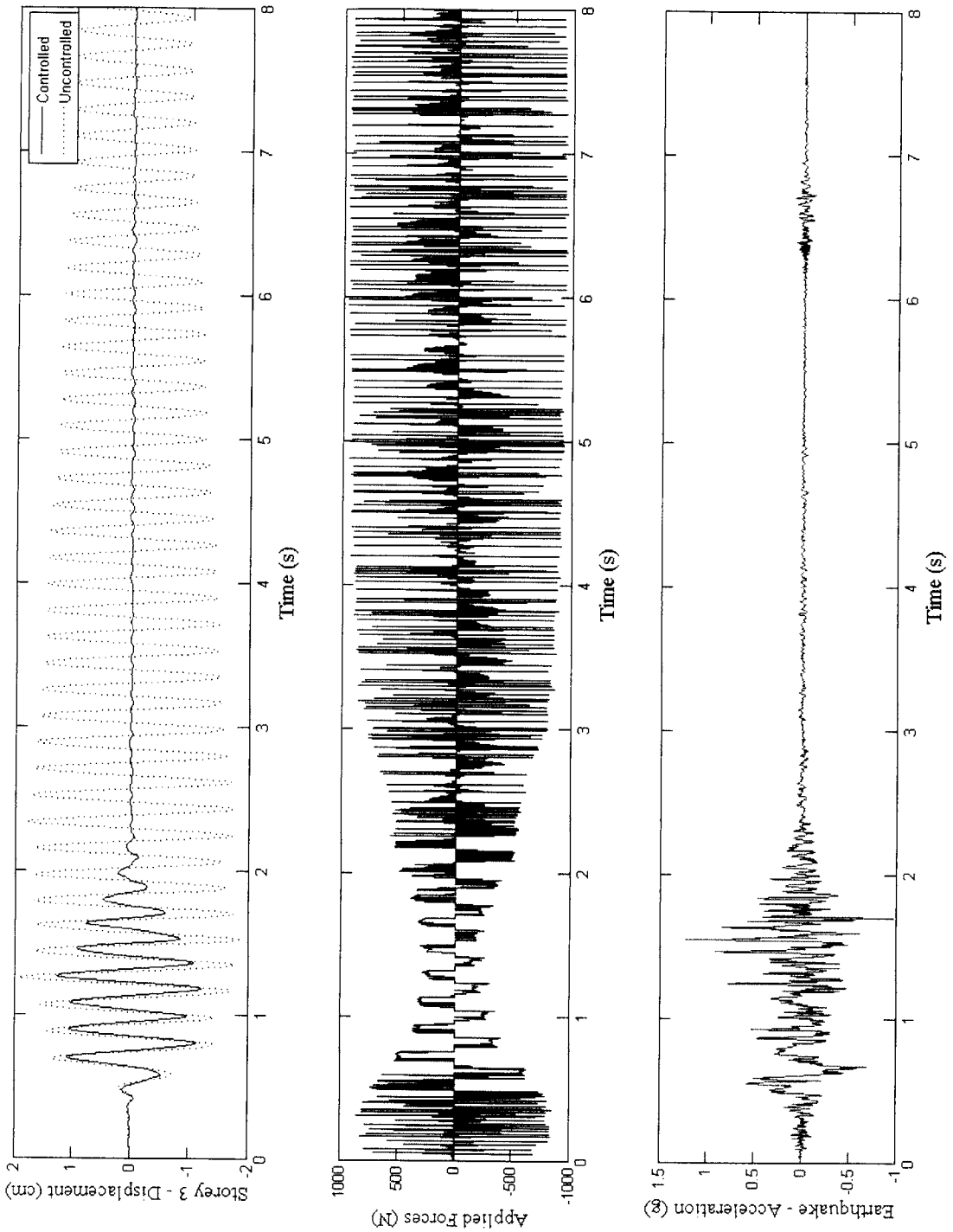

Figure 5.13: Performance for San Fernando earthquake (Pacoima)

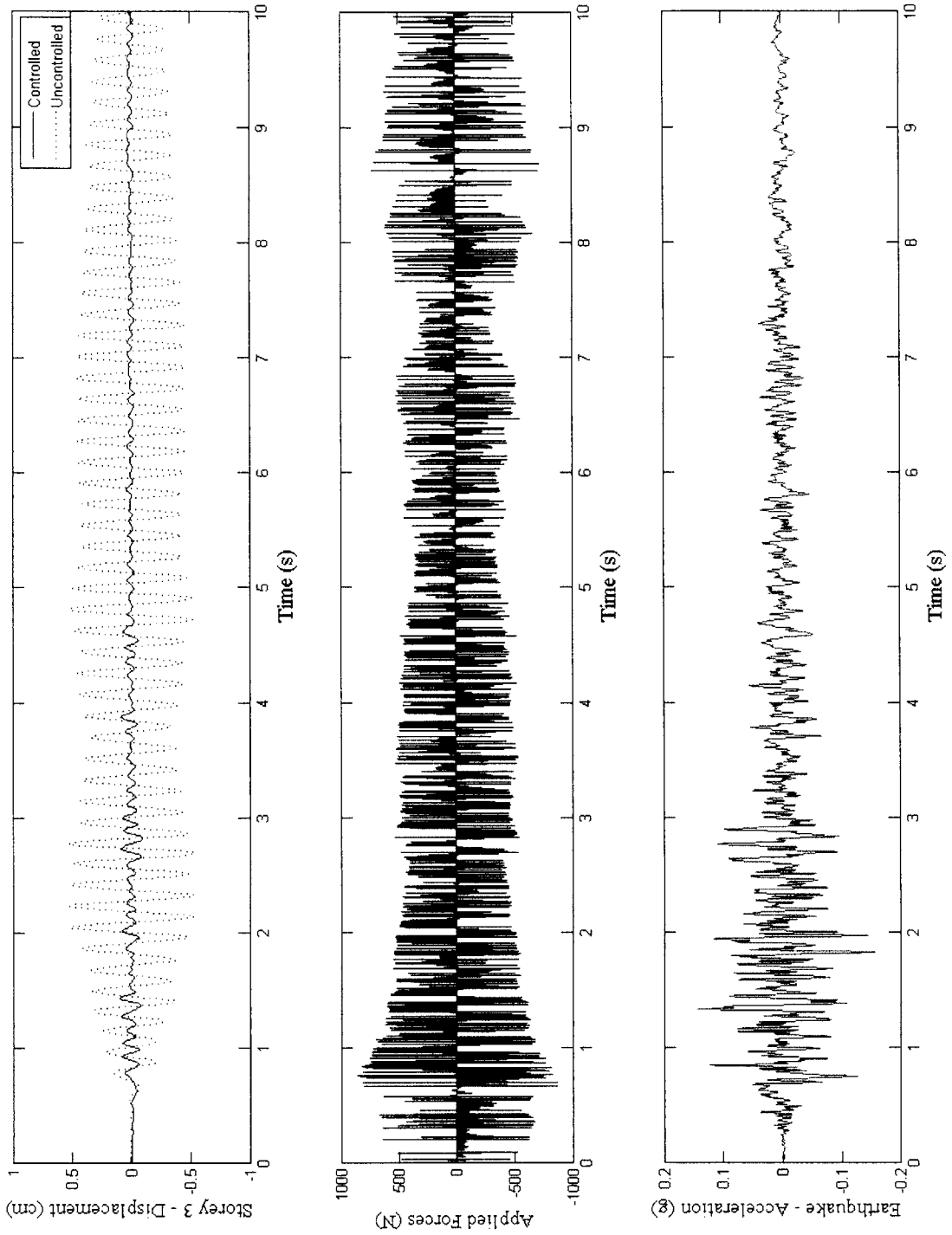

Figure 5.14: Performance for Kern County earthquake (Taft)

5.2.4 *Sensitivity of Parameters*

The previous section showed that the algorithm is stable when tested with different earthquakes. This section will look at sensitivity of the tuning parameters. These are: the learning rate η, the fitting smoothness λ and the upper bound for the maximum force.

Figure 5.15 shows the results for the learning rate η tested with the El Centro earthquake, the extreme scenario. It can be observed that the performance is not sensitive to η.

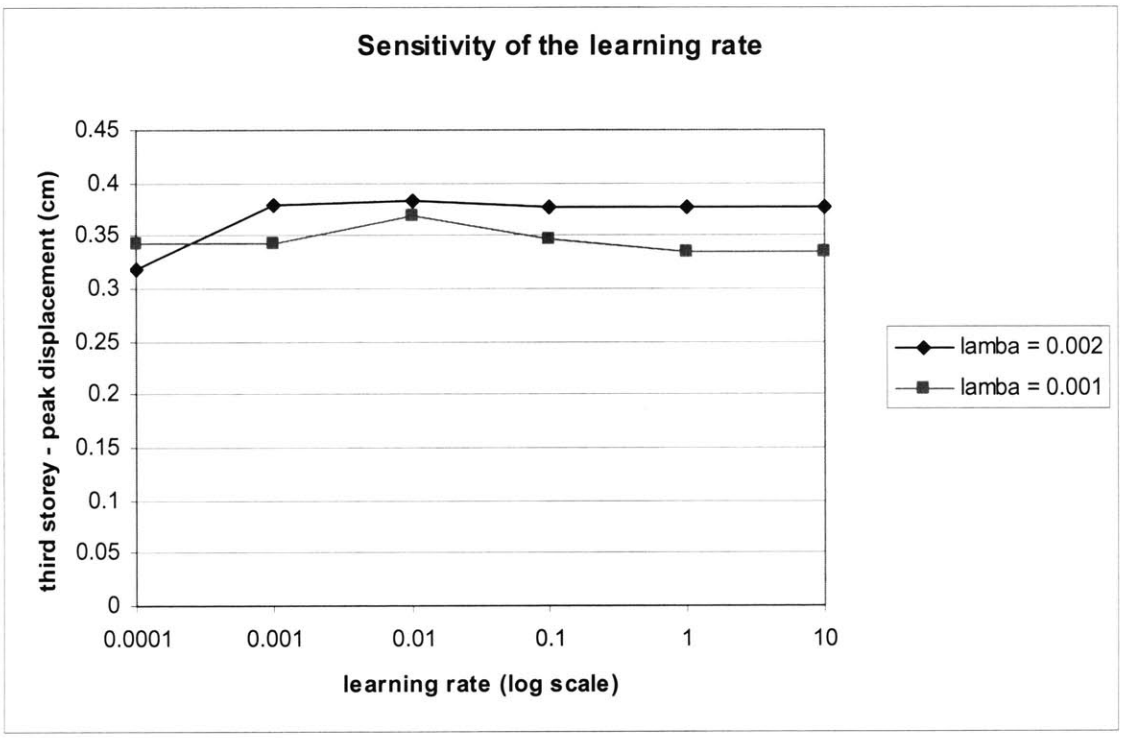

Figure 5.15: Sensitivity of the learning rate

Figure 5.16 shows the results for the smoothness parameter λ. The performance is more sensitive to λ than for the learning rate η. From the smoothness analysis, the selection rule proposed for λ, being that λ should be of the same order of magnitude as the input data (here around 0.002) shows an acceptable stability in performance.

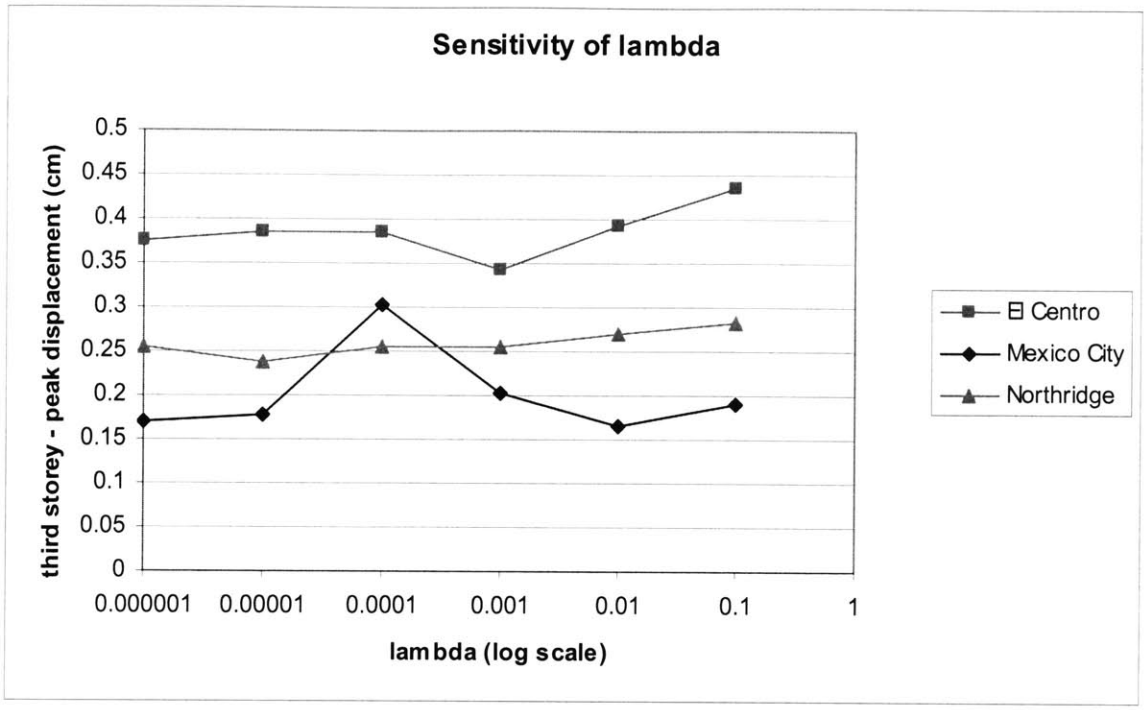

Figure 5.16: Sensitivity of the smoothness parameter

The upper bound parameter has been tested with the original parameters for the learning rate (η = 0.001) and smoothness (λ = 0.002). These user-defined inputs can be grouped as:

$$X_{t+1} = \begin{Bmatrix} \beta_u \\ \beta_d \\ \beta_a \end{Bmatrix} \tag{5-1}$$

where the subscript u is for displacement terms, d for the terms expressing the difference in displacements and a for the acceleration terms. The original parameters are β_u = 0.01; β_d = 0.001; β_a = 0.005.

All the β have been scaled from 10^2 to 10^{-2} and the peak displacements for the third storey compared along with the total applied forces over time for the El Centro, Mexico City and Northridge earthquakes. The results from the analysis show that the performances were strictly invariable with respect to the upper bound, as long as the

upper bound is there. Otherwise, the algorithm will not converge quickly enough to perform well if it is only helped by its learning rules. Figure 5.17 shows the algorithm performances with no upper bound.

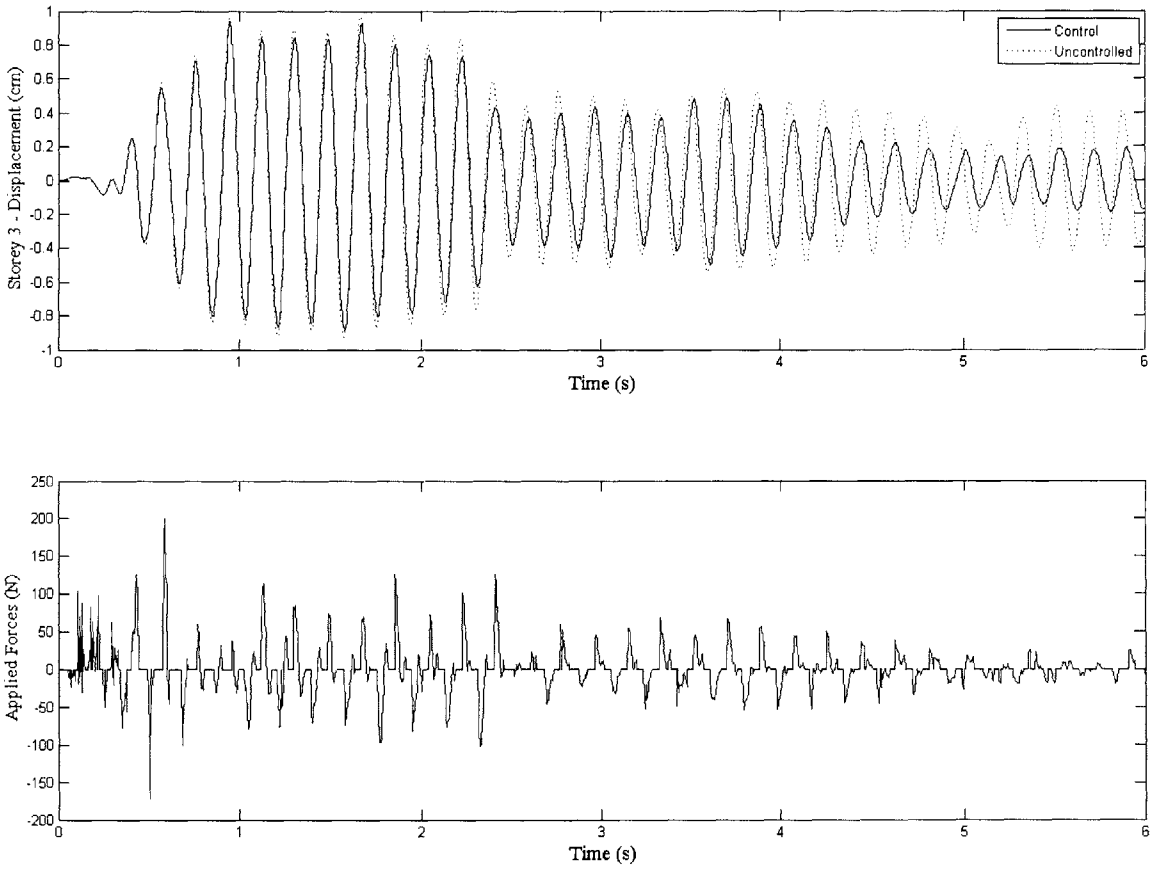

Figure 5.17: Performance without upper bound (El Centro)

The sensitivity analysis shows that the upper bound inputs could have been any numbers, and there might have been a more efficient way to introduce it in the algorithm. However, the upper bound has been kept and assigned fixed values according to selection rules for clarity regarding its purpose.

This subsection shows that the algorithm is stable with respect to its tuning parameters, but one should be careful in selecting the smoothness parameter λ.

5.2.5 Algorithm versus Passive-On 1000 N – All Five Earthquakes

In this subsection, the algorithm is compared with the passive-on 1000 N for all five earthquakes. Figures 5.18 to 5.22 graph the displacement profiles. Except for the case of the San Fernando (Pacoima) earthquake, the algorithm does slightly better at mitigating vibrations. What differentiates Pacoima from the others is its high intensity. Hence, for this case, the input data are not optimally scaled since the building acceleration is expected to be greater than for the other cases. Otherwise, it means that the algorithm gives more importance to the building acceleration: if the building experiences accelerations too far apart from the known ones, the predicted forces will be reduced. This can be observed in figure 5.13 where the controller prescribed less forces when the earthquake stroke.

This special case shows that the inputs have to be scaled according to the intensity of the earthquake. It is not wrong to assume, though, that the magnitude of intensities can be approximated according to the region and type of soil, whereas the user would be able to scale the inputs with prior knowledge. Additionally, another option would be to change the algorithm such that an automatic scaling of the inputs takes place if the magnitudes become too far apart.

Table 5.5 summarizes the performances for the optimal values where it is shown that the algorithm uses far less energy for a similar performance.

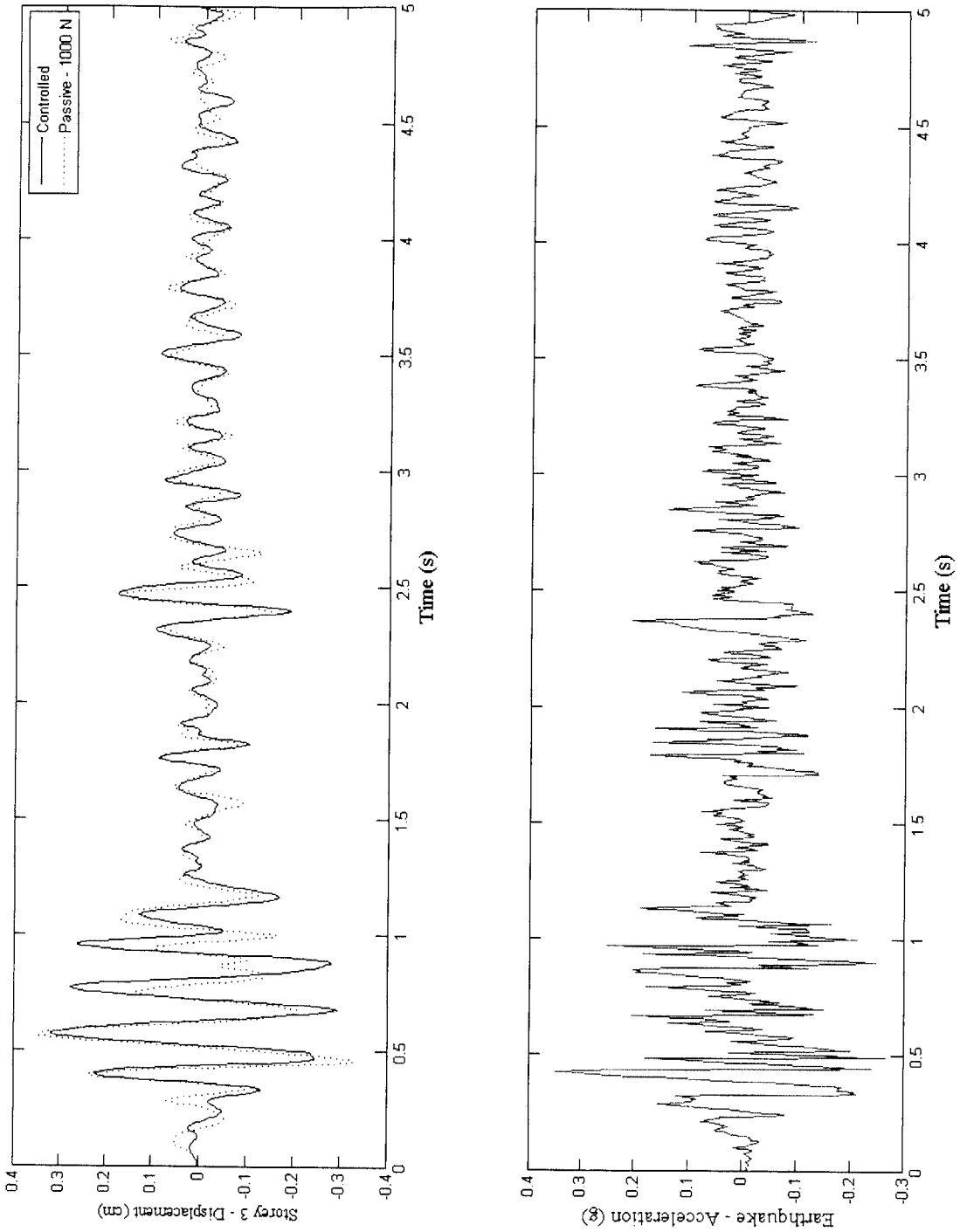

Figure 5.18: Performance for Imperial Valley earthquake (El Centro) versus passive-on 1000 N

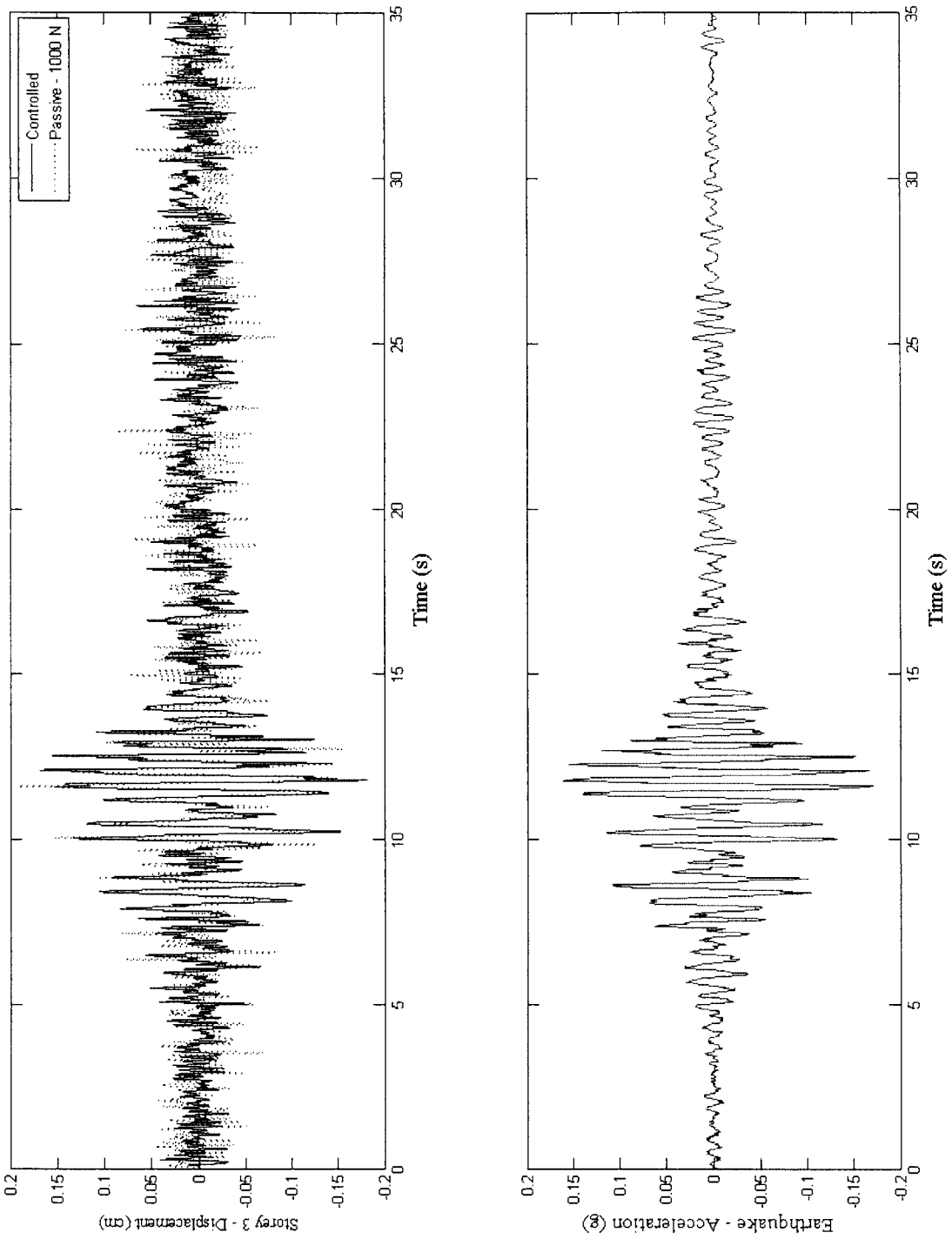

Figure 5.19: Performance for Mexico City earthquake versus passive-on 1000 N

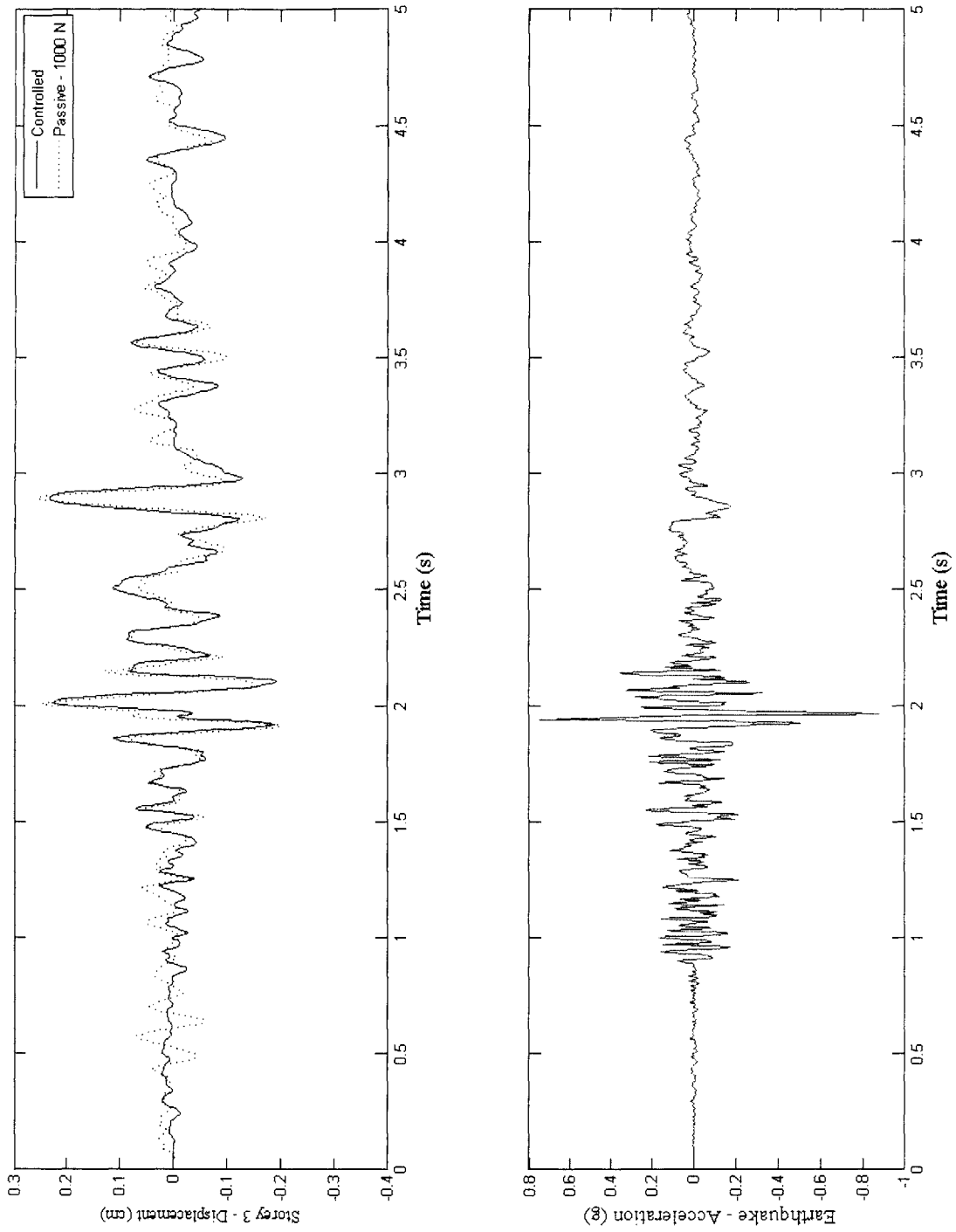

Figure 5.20: Performance for Northridge earthquake versus passive-on 1000 N

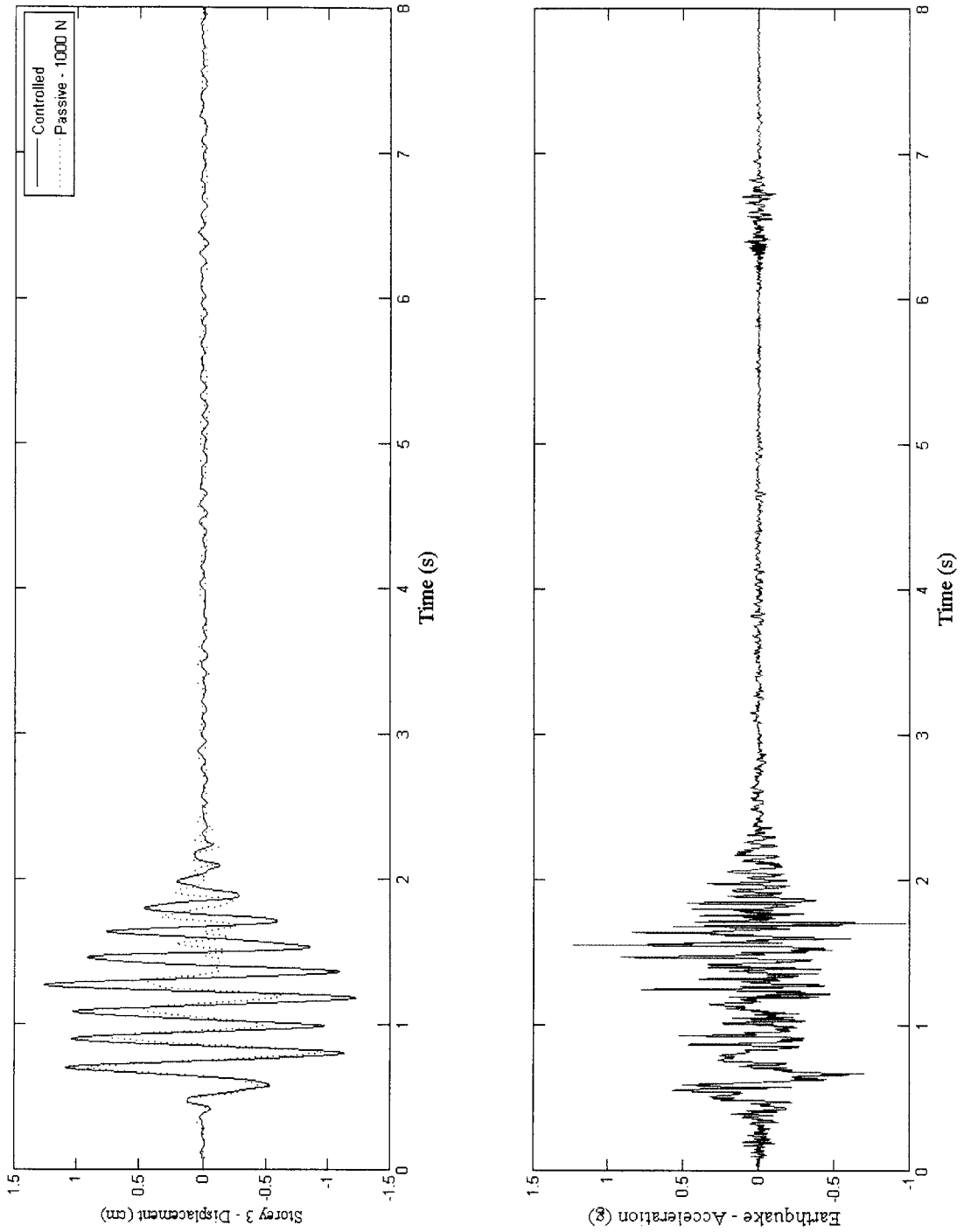

Figure 5.21: Performance for San Fernando earthquake (Pacoima) versus passive-on 1000 N

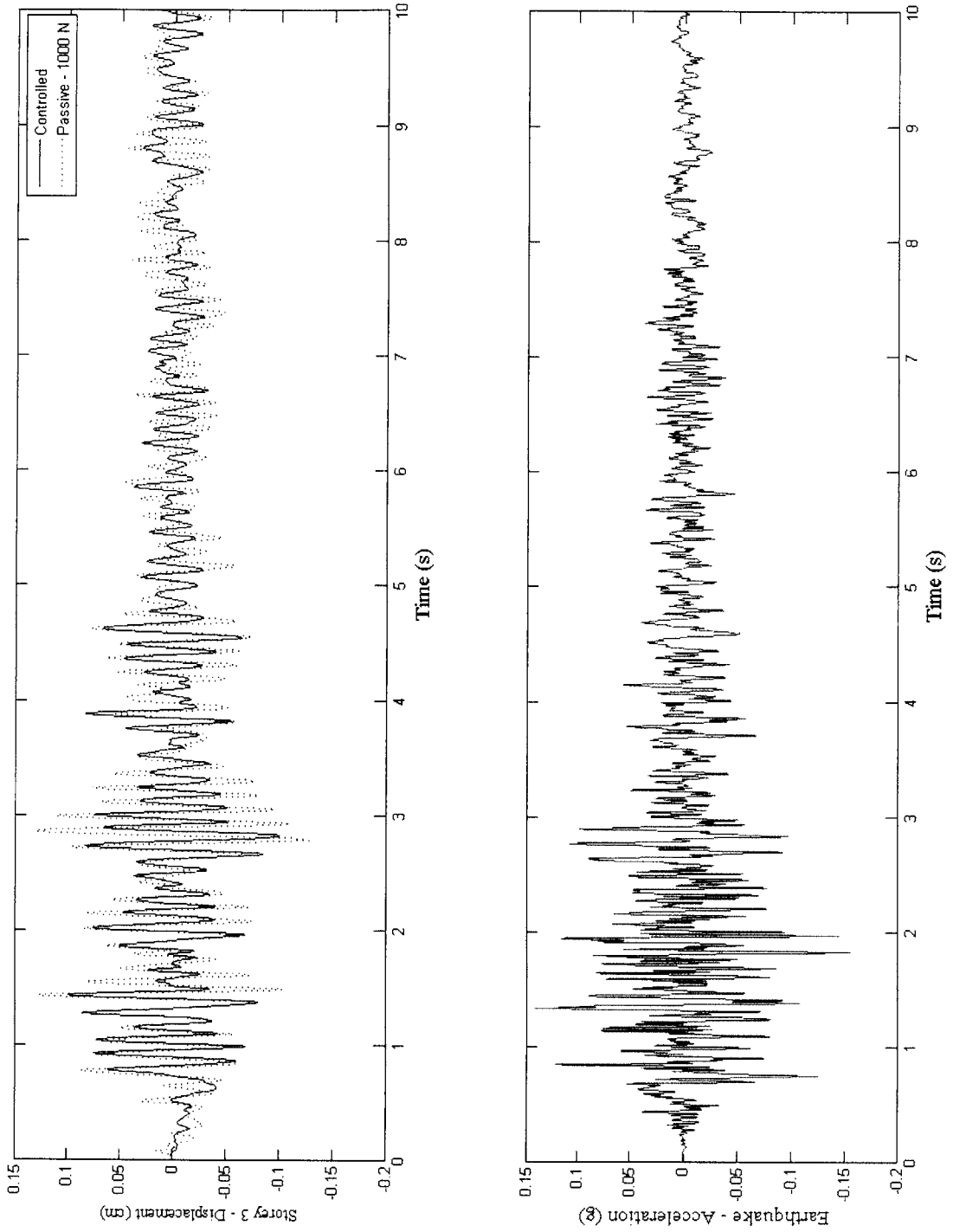

Figure 5.22: Performance for Kern County earthquake (Taft) versus passive-on 1000 N

Earthquake	Imperial Valley	Mexico City	Northridge	San Fernando	Kern County
Peak Displacement Uncontrolled (cm)	0.974	0.347	0.482	1.92	0.542
Peak Displacement Algorithm (cm)	0.318	0.184	0.232	1.25	0.100
Peak Displacement Passive-1000N (cm)	0.346	0.192	0.258	0.953	0.131
Reduced Peak Displacement Algorithm (%)	67.4	46.9	51.9	34.9	81.5
Reduced Peak Displacement Passive-1000N (%)	64.5	44.6	46.5	50.4	75.8
Total Applied Forces Algorithm (kN)	267	1640	228	647	679
Total Applied Forces Passive-1000N (kN)	385	2170	351	897	1140
Running Time (s)	5	35	5	8	10
Applied Force / Second Algorithm (kN)	53.4	46.9	45.6	80.9	67.9
Applied Force / Second Passive-1000N (kN)	77.0	62.0	70.2	112	114

Table 5.5: Comparison of performances for all five earthquakes – algorithm versus passive-on 1000 N

It follows that the San Fernando case has been tested with scales of 10 000 and 100 000 rather than 1 000. Results are illustrated in figures 5.23 and 5.24, and summarized in table 5.6. The performance of the algorithm has significantly increased, but is still underneath the passive-on case results. Also, it can be observed that when the accelerations are excessively scaled, the algorithm will start to be insensitive high new displacement inputs. This shows that the proposed scheme has a limitation towards earthquakes of clustered acceleration types.

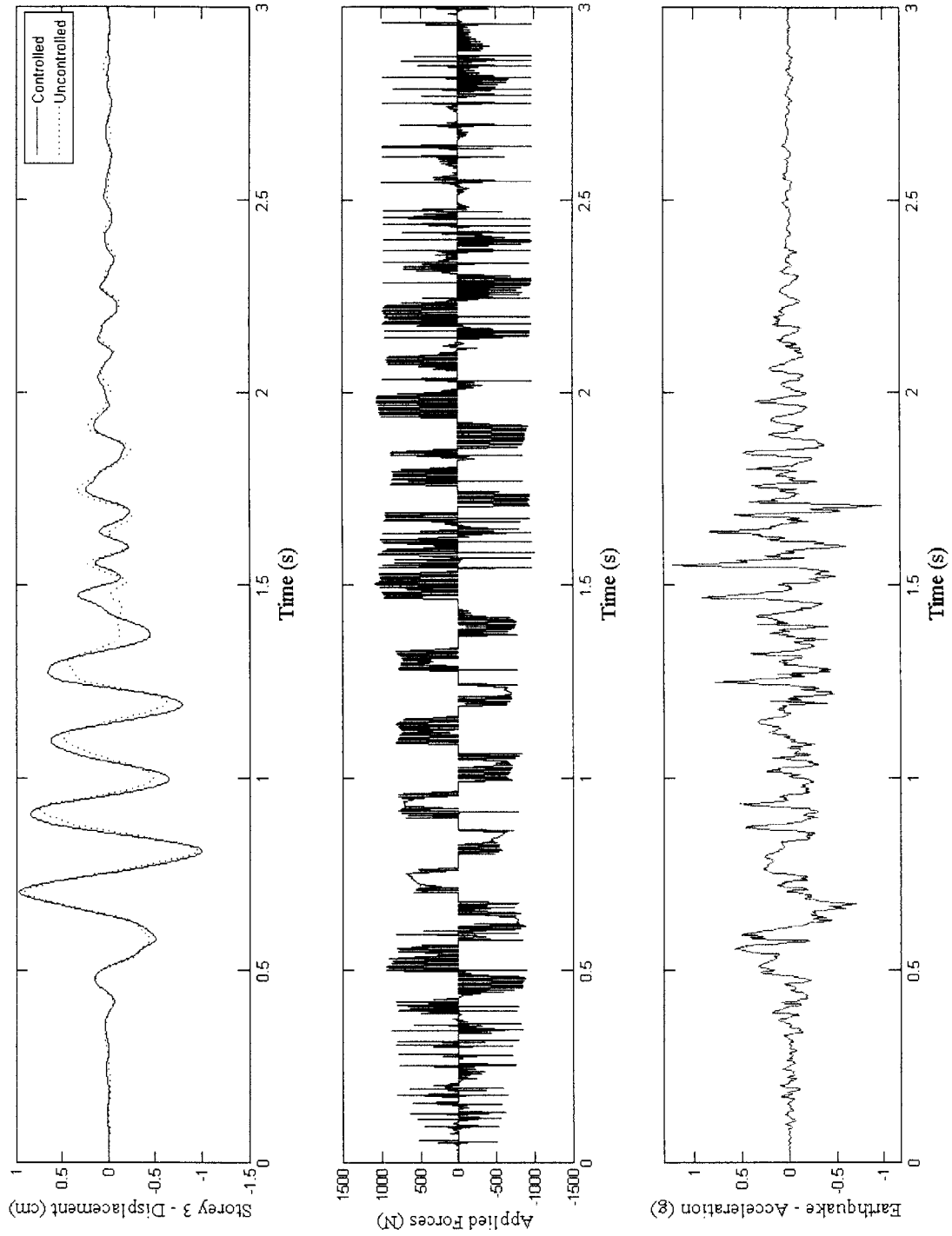

Figure 5.23: Performance for San Fernando earthquake (Pacoima), acceleration input scaled by 10 000

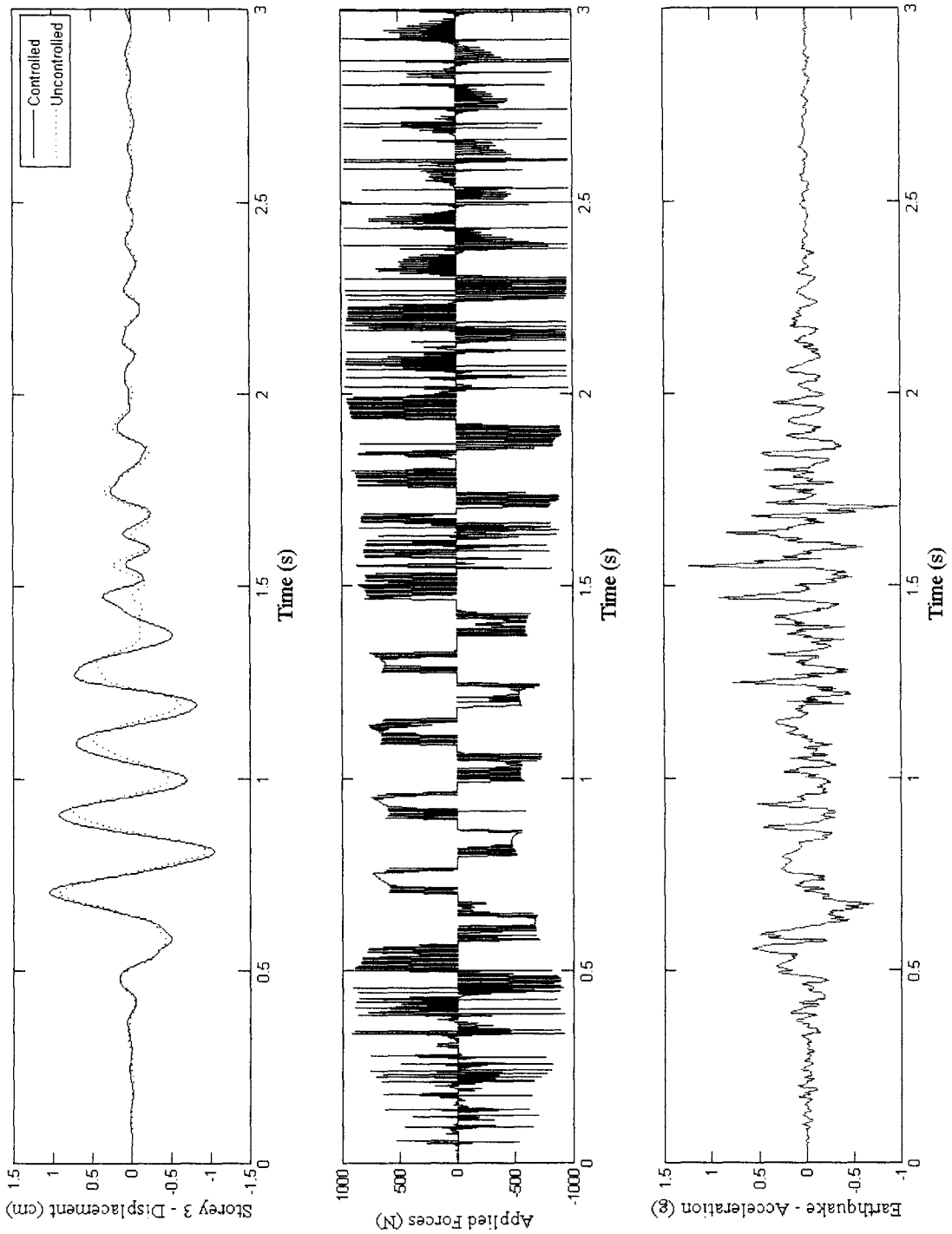

Figure 5.24: Performance for San Fernando earthquake (Pacoima), acceleration input scaled by 100 000

	1 000 scale	10 000 scale	100 000 scale
Peak Displacement Algorithm (cm)	1.25	1.01	0.978
Peak Reduction Algorithm (%)	34.9	47.4	49.0
Total Applied Forces Algorithm (kN)	251	400	432
running time (s)	3	3	3
applied force / second - algorithm (kN)	83.7	133	144

Table 5.6: Summary of results for San Fernando earthquake

6.0 CONCLUSION

The proposed algorithm is a first step towards online learning for structural control in civil engineering. It performs well at mitigating vibrations for low energy costs and is able to do well for different situations without any additional tuning. It does not perform as well as classical control does, but as stated previously, its relative performance with respect to classical controls would be expected to be better for large-scale situations. When compared to the passive case with full current input, the proposed algorithm does similarly with significantly lower energy input. However, some extreme situations such as the San Fernando earthquake might require proper scaling of inputs or a new version of the algorithm allowing automatic scaling.

Invariably, the proposed algorithm has some limitations. First, it still requires a significant quantity of sensors: a strain gauge and an accelerometer per storey. It might be possible to adapt the control scheme to only have some selected DOFs dynamic responses as input. A second limitation is its performance regarding clustered acceleration type earthquakes, as discussed in section 5. Additionally, the algorithm is sometime sensitive to scaling of input elements data whereas one has to pay particular attention during setup. It also has to be said that the simulation has been done with MATLAB and that some assumptions have been introduced in the model, such as the maximum reaction forces of the actuator. An effort has been made to build this simulation rigorously and as close as possible to the reality.

Magnetorheological actuators, by themselves, are capable of mitigating earthquake vibrations upon a general power failure as they can run on small batteries. This online algorithm is a promising step towards a broad implementation of magnetorheological actuators since it does not require full knowledge of the structure or the actuator and can adapt to external situations. In addition, it could adjust regardless of the location of the actuators and their allowable forces, which is essential for MR actuators since their maximum forces are limited.

It would be interesting to extend the algorithm for decentralized control of multiple actuators and assess its performance. Also, a large-scale experiment would allow better comparison of results with respect to classical theory. Moreover, as suggested in the text, the algorithm could be adapted to have automatic scaling of inputs and to control with a limited number of sensors for a MDOF structure. Finally, the algorithm could be evaluated upon distributed loads such as strong winds.

REFERENCES

Annema, A.-J. (1995), *Feed-Forward Neural Networks: Vector Decomposition, Analysis, Modelling and Analog Implementation*, Kluwer Academic Publishers, 1995.

Beaver, J.L. (1999), *Control Algorithms For Adaptive and Semi-Active Systems*, Master of Science Thesis, Department of Civil and Environmental Engineering, MIT.

Cho, S.W., Kim, B.-W., Jung, H.-J. and Lee, I.-W. (2005a), *Implementation of Modal Control for Seismically Excited Structures using Magnetorheological Dampers*: Journal of Engineering Mechanics, February 2005, 177-184

Cho, S.W., Jung, H.-J. and Lee, I.-W. (2005b), *Smart Passive System Based on Magnetorheological Damper*: Smart Materials and Structures, 2005:14, 707-714

Connor, J.J. (2003), *Introduction to Structural Motion Control*, Prentice Hall, 2003.

Dyke, S.J., Spencer, B.F., Quast, P. and Sain, M.K. (1995), *Role of Control-Structure Interaction in Protective System Design*, Journal of Engineering Mechanics, ASCE, February 1995, p. 322-338.

Dyke, S.J., Spencer, B.F., Sain, M.K. and Carlson, J.D. (1996), *Modeling and Control of Magnetorheological Dampers for Seismic Response Reduction*, Smart Materials and Structures, 5:1996, p. 565-575.

Evgeniou, T, Pontil, M. and Poggio, T. (2000), *Regularization Networks and Support Vector Machines*, Advances in Computational Mathematics, 2000

Hagan, M.T., Demuth, H.B. and Beale, M. (1996), *Neural Network Design*, PWS Publishing Company, 1996.

Jansen, L. and Dyke, S. J. (2000), *Semiactive Control Strategies for MR Dampers: Comparative Study*: Journal of Engineering Mechanics, August 2000, 795-803

Jung, S., Ghaboussi, J. and Kwon, S.-D. (2004), *Estimation of Aeroelastic Parameters of Bridge Decks Using Neural Networks*, Journal of Engineering Mechanics, ASCE, November 2004, p.1356-1364.

Langley, Pat (1996), *Elements of Machine Learning*, Morgan Kaufmann Publishers, 1996.

Liu, Y., Gordaninejad, F., Evrensel, C. A., Wang X. and Hitchcock, G. (2005), *Comparative Study on Vibration Control of a Scale Bridge Using Fail-Safe Magneto-Rheological Fluid Dampers*: Journal of Structural Engineering, May 2005, 743-751

Lord Corporation
www.lord.com

Ribakov, Y. and Gluck, J. (2001), *Selective Combine Control Stiffness and Magnetorheological Damping System in Nonlinear Multistorey Structures*: The Structural Design of Tall Buildings, 11, 171-195

Rifkin, R. and Lippert, R. (2007), *Value Regularization and Fenchel Duality*, JMLR (to be published).

Russell, S. and Norvig, P. (2003), *Artificial Intelligence: A Modern Approach*, Second Edition, Prentice Hall, 2003.

Sanner, R.M., Slontine, J.-J. (1992), *Gaussian Networks for Direct Adaptive Control*, IEEE Transactions on Neural Networkds, Vol. 3, no. 6, November 1992, p. 837-863.

Schölkopf, B. and Smola, A. (2001), *Learning with Kernels*, The MIT Press, Cambridge, 2001.

Spencer, B. F. Jr. and Nagarajajah, S. (2003), *State of the Art of Structural Control*: Journal of Structural Engineering, July 2003: 845-856

Winston, P.H. (1992), *Artificial Intelligence*, Third Edition, Addison-Wesley Publishing Company, June 1992.

Wu, W.J., Cai, C.S. and Chen, S.R. (2004), *Experiments on Reduction of Cable Vibration Using MR Dampers*: 17th ASCE Engineering Mechanics Conference, June 13-16, 2004, University of Delaware, Newark, DE.

Yang, G., Spencer, B. F. Jr., Carlson, J. D. and Sain, M. K. (2002), *Large-Scale MR Fluid Dampers: Modeling and Dynamic Performance Considerations*: Engineering Structures 24, 309-323

Yang, G., Spencer, B. F. Jr., Jung, H.-J. and Carlson, J.D.(2004), *Dynamic Modeling of Large-Scale Magnetorheological Damper Systems for Civil Engineering Applications*: Journal of Engineering Mechanics, September 2004, 1107-1114.

APPENDIX A – MATLAB CODE

This appendix presents the MATLAB code used for the simulation. Note that it has not been written by an expert in computer programming and that consequently, it can easily be optimized. The average running time of this code with a 1.50 GHz / 512 Mb RAM laptop is 7.4 milliseconds per time step.

```
clear all; clf;

dt=0.02/5;
steps=6/dt;
int=50;
delay=5;

% Get Earthquake Data

load ElCentro_NS.txt
ag_elcentro=ElCentro_NS(:,2)*9.81;
%load Mexcit2.txt
%ag_mexico=Mexcit2(:,2)*9.81;
%load Nridge2.txt
%ag_northridge=Nridge2(:,2)*9.81;
%load Pacoima1.txt
%ag_pacoima=Pacoima1(:,2)*9.81;
%load Kern1.txt
%ag_kern=Kern1(:,2)*9.81;

Fmax=1000;
for i=1:steps
    p_ag(i)=ag_elcentro(i);
end

% Structure Properties

M=eye(3)*98.3;
C=[175,-50,0;-50,100,-50;0,-50,50];
K=1e5*[12,-6.84,0;-6.84,13.7,-6.84;0,-6.84,6.84];
I=eye(3); Itwo=eye(2); Isix=eye(6); Z=zeros(3);
A=[Z,I;-inv(M)*K,-inv(M)*C];
Bg=[0;0;0;-1;-1;-1];
Ef=[1;0;0];
Bf=[0;0;0;inv(M)*Ef];

% Uncontrolled Response

Xunc=zeros(6,steps);
for i=1:steps
    Xunc(:,i+1)=expm(A*dt)*Xunc(:,i)+inv(A)*(expm(A*dt)-
Isix)*(Bg.*p_ag(i));
```

```
    Xdotunc(:,i)=A*Xunc(:,i)+Bg.*p_ag(i);
end

% Controlled Response

    % Tuning Parameters

nd=1; nv=1; na=1000;
n=.001;
bu=0.01; bd=0.001; ba=0.005;
lam=.002;
sig=.1;

    % Algorithm

Xctrl=zeros(6,steps);
for i=1:steps
    tic
    if i<delay+1
        % Initial Values
        Flearn(i)=0; pred(i)=0; mode(i)=1; Xctrl(1,i)=0;
    else
        % State Inputs
        indata(1,i)=Xctrl(1,i-1)/nd;
        indata(2,i)=Xctrl(1,i-2)/nd;
        indata(3,i)=Xctrl(2,i-1)/nd;
        indata(4,i)=Xctrl(2,i-2)/nd;
        indata(5,i)=Xctrl(3,i-1)/nd;
        indata(6,i)=Xctrl(3,i-2)/nd;
        indata(7,i)=(Xctrl(1,i-1)-Xctrl(1,i-2))/nv;
        indata(8,i)=(Xctrl(2,i-1)-Xctrl(2,i-2))/nv;
        indata(9,i)=(Xctrl(3,i-1)-Xctrl(3,i-2))/nv;
        indata(10,i)=Xdotctrl(4,i-1)/na;
        indata(11,i)=Xdotctrl(4,i-2)/na;
        indata(12,i)=Xdotctrl(4,i-3)/na;

        % Upper Bound
        indata(1:6,i+1)=bu;
        indata(7:9,i+1)=bd;
        indata(10:12,i+1)=ba;

        % Kernel
        if i<int+delay
            look=i-delay;
        else
            look=int;
        end
        for ii=1:look+1
            for jj=1:look+1
                KERN(ii,jj)=exp((-norm(indata(:,ii)-
indata(:,jj)))/sig);
            end
            if ii<look+1
                U(ii)=pred(i-look-1+ii);
            else
```

```
                U(ii)=Fmax;
            end
        end
        I=eye(size(KERN));
        c=(KERN+lam*I)\U';

        % New State Inputs
        newdata(1,i)=Xctrl(1,i)/nd;
        newdata(2,i)=Xctrl(1,i-1)/nd;
        newdata(3,i)=Xctrl(2,i)/nd;
        newdata(4,i)=Xctrl(2,i-1)/nd;
        newdata(5,i)=Xctrl(3,i)/nd;
        newdata(6,i)=Xctrl(3,i-1)/nd;
        newdata(7,i)=(Xctrl(1,i)-Xctrl(3,i-1))/nv;
        newdata(8,i)=(Xctrl(2,i)-Xctrl(2,i-1))/nv;
        newdata(9,i)=(Xctrl(3,i)-Xctrl(1,i-1))/nv;
        newdata(10,i)=Xdotctrl(4,i)/na;
        newdata(11,i)=Xdotctrl(4,i-1)/na;
        newdata(12,i)=Xdotctrl(4,i-2)/na;

        % Force Predictor
        pred(i)=0;
        for ii=1:look+1
            pred(i)=pred(i)+exp((-norm(newdata(:,i)-
indata(:,ii)))/sig)*c(ii);
        end

        %Applying Forces
        if (sign(Xdotctrl(4,i))==sign(Xctrl(1,i)-Xctrl(1,i-1)))
            Flearn(i)=pred(i)*-sign(Xctrl(1,i)-Xctrl(1,i-1));
        else
            Flearn(i)=0;
        end

        if (abs(Flearn(i))>98.3*3*abs(Xdotctrl(4,i)))
            Flearn(i)=-98.3*3*abs(Xdotctrl(4,i))*sign(Xctrl(1,i)-
Xctrl(1,i-1));
            check=0;
        end
    end

    % Simulated Dynamic Responses
    Xctrl(:,i+1)=expm(A*dt)*Xctrl(:,i)+inv(A)*(expm(A*dt)-
Isix)*(Bg.*p_ag(i)+Bf.*Flearn(i));
    Xdotctrl(:,i+1)=A*Xctrl(:,i+1)+Bg.*p_ag(i)+Bf.*Flearn(i);

    % Adapting Forces
    if i>delay
        if max(abs(Xctrl(1:3,i)))>n/2
            n=n*10;
        end

        if (sign(Xdotctrl(4,i))==sign(Xctrl(1,i)-Xctrl(1,i-1)))
            if (abs(Xdotctrl(4,i+1))-abs(Xdotctrl(4,i))<0)
```

```
                pred(i)=pred(i)+(1-
pred(i)/Fmax)*Fmax*abs(Xctrl(3,i+1))/n;
            else
                pred(i)=pred(i)*(1-abs(Xctrl(3,i+1))/n);
            end
        end
    end
    toc
    time(i)=i/250;
end

% Plots

subplot(3,1,1); plot(time,Xctrl(3,1:steps)*100,'b'); hold all;
plot(time,Xunc(3,1:steps)*100,'k:')
ylabel('\fontname{times} Storey 3 - Displacement (cm)', 'FontSize', 12)
xlabel('\fontname{times} Time (s)', 'FontSize', 14)
legend('Controlled','Uncontrolled')
subplot(3,1,2); plot(time,Flearn(1:steps),'b');
ylabel('\fontname{times} Applied Forces (N)', 'FontSize', 12)
xlabel('\fontname{times} Time (s)', 'FontSize', 14)
subplot(3,1,3); plot(time,p_ag(1:steps)/9.81,'r');
ylabel('\fontname{times} Earthquake - Acceleration (g)', 'FontSize',
12)
xlabel('\fontname{times} Time (s)', 'FontSize', 14)
```