sensors

MDPI

Article

Seismic Model Parameter Optimization for Building Structures

Lengyel Károly †, Ovidiu Stan *,† and Liviu Miclea †

Department of Automation, Faculty of Automation and Computer Science, Technical University of Cluj-Napoca, Memorandumului Str. 28, 400014 Cluj-Napoca, Romania; Karoly.Lengyel@student.utcluj.ro (L.K.); Liviu.Miclea@aut.utcluj.ro (L.M.)

* Correspondence: ovidiu.stan@aut.utcluj.ro

† These authors contributed equally to this work.

Received: 18 February 2020; Accepted: 29 March 2020; Published: 1 April 2020

Abstract: Structural dynamic modeling is a key element in the analysis of building behavior for different environmental factors. Having this in mind, the authors propose a simple nonlinear model for studying the behavior of buildings in the case of earthquakes. Structural analysis is a key component of seismic design and evaluation. It began more than 100 years ago when seismic regulations adopted static analyzes with lateral loads of about 10% of the weight of the structure. Due to the dynamics and non-linear response of the structures, advanced analytical procedures were implemented over time. The authors' approach is the following: having a nonlinear dynamic model (in this case, a multi-segment inverted pendulum on a cart with mass-spring-damper rotational joints) and at least two datasets of a building, the parameters of the building's model are estimated using optimization algorithms: Particle Swarm Optimization (PSO) and Differential Evolution (DE). Not having much expertise on structural modeling, the present paper is focused on two aspects: the proposed model's performance and the optimization algorithms performance. Results show that among these algorithms, the DE algorithm outperformed its counterpart in most situations. As for the model, the results show us that it performs well in prediction scenarios.

Keywords: structural dynamic modeling; optimization; DE; PSO; parameter estimation; extended Kalman filter; inverted pendulum

1. Introduction

Structural dynamic modeling of buildings has come a long way in the last 50 years, due to the competition of software development companies and the increased availability of computational resources. These technologies have evolved from simulating only prismatic beams to including geometrical and material nonlinearities [1]. From a control engineering perspective, these models have a particularly great importance when designing control systems for earthquake hazard mitigation. Having a good model exclusively for the above mentioned purpose can significantly improve the behavior of these systems [2].

The civil engineering field is imaginative, and it ranges from water-resources to structural design and analysis. Generally speaking, the problems in this field are unstructured and imprecise, influenced by a designer's intuitions and past experiences. The conventional computing methods based on analytic or empirical relationships take time and are labor intensive when they are presented with real life problems. In addition, Soft Computing techniques (SC) based on the reasoning, intuition, conscience, and knowledge of an individual can be easily empowered to study, model, and analyze such problems [3,4].

Unlike conventional computing technology based on exact solutions, in SC, either independent or mutually complementary work supports engineering activities by utilizing the human mind's cognitive behavior to achieve cost-effective solutions aimed at exploiting the trivial and uncertain nature of the problem in a given tolerance of imprecision to achieve a quick solution to a problem [5]. As a multidisciplinary field, SC employs a variety of complementary tools, including statistical, probability, and optimization tools.

According to Falcone et al., SC should be divided into two main domains [5]. The first one, approximate thinking, collects a set of knowledge-driven methods that sacrifice health or completeness in order to achieve a substantial speed of thinking. The second one, randomized search, is also a family of digitally optimized techniques, such as direct search, free derivative search, or black-box search, which work by moving iteratively to better positions in the search space, which are sampled from a surrounding hyper sphere.

Earthquake engineering can be described as the civil engineering branch devoted to reducing the risks of an earthquake. An earthquake is the moment when the Earth's surface is shaking, which is caused by moving interactions on the boundary of a plate [6]. The sudden release of energy, called seismic waves, kills thousands of people and destroys many buildings. In this narrow context, earthquake engineering examines problems that occur when the earthquake occurs and seeks methods that minimize the damage caused by its activities. The first leads to the prediction of an earthquake, while the second leads to the optimal design of objects' seismic performance. A whole range of earthquake engineering problems have arisen that are suitable to solve by SC [7,8]. The focus of SC in earthquake engineering is on solving two types of problems: the search for the best seismic structural design (system analysis); data analysis for earthquake prediction (modeling and simulation). In order to achieve earthquake safety, seismic design optimization addresses passive and active structures [9].

The goal of this project is to measure the performances of different variants of DE and PSO in optimizing the parameters in a proposed seismic model for building structures that is lightweight enough to be used for different applications that require easy computation and reliability. The authors will analyze the convergence speed and other indicators of the algorithms' performance and will compare the two algorithms in this use case. The achieved model will also be tested in two scenarios: simulation and prediction. The prediction will be performed using an extended Kalman filter.

The proposed model is a multiple segment inverted pendulum on a cart with mass-spring-damper rotational joints, as illustrated in Figure 1.

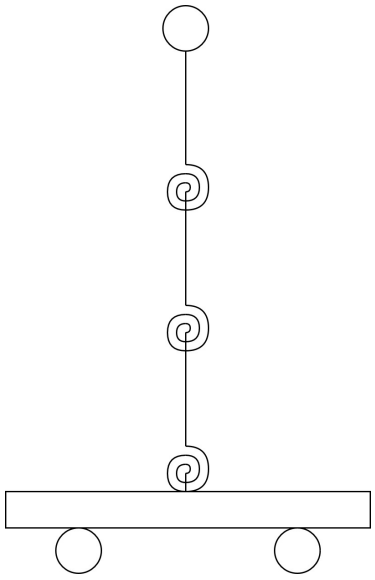


Figure 1. Three segment inverted pendulum on a cart with mass-spring-damper joints.

The bibliographic search for this project can be divided into the following sections, regarding the field in which it was performed:

- System identification and parameter estimation
- Kinetic modeling
- Kalman filter
- Optimization

1.1. Context

Most buildings are deformed significantly when strong earthquakes affect them. One of the factors contributing to quantitative thinking beyond the elastic response of structures is the gap between measured ground speed and the seismic design forces defined in codes [10]. There is however a long way to go prior to more advanced seismic codes for the explicit nonlinear analysis. The use of force reducing factors was initially the most popular approach, and today, this approach is still popular. Although this concept of taking inelastic behavior into account has been useful for many decades in linear analysis, it is only possible to give a realistic assessment of structural behavior in the inelastic range through non-linear analysis. A gradual implementation of nonlinear analyses, which should be explicitly able to simulate the second fundamental feature of a structured answer to strong seismic movement of the Earth, namely the inelastic behavior, characterizes present developments of the analysis processes in seismic codes. Data on the structure need to be known for such nonlinear analyses, which makes them well suited for analyzing existing structures. For newly designed structures, a preliminary design must be carried out before a nonlinear analysis is started [11].

1.2. System Identification and Parameter Estimation

Due to the uncertainty, time-lagging, multi-variable couplings, and the limitations between the input and output, traditional model control methods are becoming increasingly difficult to control complex processes correctly in the rapid development of modern industry. Due to the complex structure, different parameters and time variations for industrial applications, this is a challenge for traditional identification methods, particularly in multivariate systems. Methods for identifying multi-variable systems date back to the 1960s, but the majority of methods for identifying them require noise-free observations. Together with their high calculation costs, this makes them difficult to apply in practice [12]. In view of the above problems, many scientists proposed that a polynomial matrix be substituted for the state space model, to define the multi-variable system [13].

Some researchers then proposed the Hankel matrix-based methods for row subspace identification. The first step in this method is to obtain the system's increased observability matrix (or status sequence) and then calculate the parameter matrix of each sub-space. Multi-variable output error status [13], sub-space state-space identification numerical algorithms [14], and canonical variate analyses [15] are the main representative techniques.

Input signal selection is an important factor in system identification, as stated in [16], where the authors discussed the importance of the input signal selection and explained, briefly, a few types of input signals for system identification. The discussed signals were: the step, pseudo random binary sequence, auto-regressive moving average process, and sum of sinusoids. Based on this information and from prior knowledge, our choices for identifying signals were the step and PRBS signals. In the same book, in Chapter 1, the authors discussed the influence of data feedback on the identification performances. This is of great importance, because our system has strong feedback. They concluded that by having feedback in a system, it can make it unidentifiable. However, by having a reference signal, the previously mentioned problem disappears, affecting the identification performance.

Extensive research has also been done on PSO's performance compared to that of GA. One example is [17], where the authors discussed the performance of the PSO algorithm compared to that of Genetic Algorithms (GA) in system identification. Their case was a nonlinear mode,l and

the experiment was performed online. They concluded that this type of algorithm is an efficient tool in nonlinear system identification, producing similar and better results than GA, having the advantage of low computational cost and faster convergence. Worden et al. also recently arrived at the same conclusion about nonlinear system identification [18]. The identification of non-linear systems involves much more than linear identification. The following aspects contribute to this observation: non-linear models live in a multiplex system of a greater size, while linear models live in easier to characterize simple hyperplanes; in non-linear system identification, structural model errors are frequently inevitable, and this affects the three main choices: experiment design, model selection, and cost-function selection; entering noise before non-linearity requires new numerical tools to solve the problem of optimization [19]. Moreover, extensive research has been done to compare parameter estimation capabilities to PSO variations like PSO, APSO, and Quantum behaved PSO (QPSO) [20,21]. The nonlinear model types on which the experiments are performed are the Hammerstein and Wiener models. Their conclusion was that using swarm intelligence, such as modifying the original algorithm, improved the parameter estimation performance. Other variations of the PSO algorithm have been studied for system identification; for example, the PSO-QI algorithm was discussed in [22], where the authors of the paper analyzed the use of the PSO-QI algorithm for system identification, which was compared to to the classic PSO and DE. They concluded that for system identification, the modified algorithm was the best among the three because of its fast convergence.

Research has also been done when using DE for system identification and parameter estimation in systems. The work in [23] discussed the optimal approximation of linear systems using a Differential Evolution (DE) algorithm. The authors incorporated a search space expansion scheme in order to overcome the difficulty of specifying proper intervals for initializing the DE search. Besides PSO, DE variations have also been studied for these tasks, for example in [24], where the authors discussed a hybrid DE algorithm for nonlinear parameter estimation of kinetic systems. In this article, the authors combined the DE algorithm with the Gauss–Newton method. Basically, the DE was used to provide a good initial point for the Gauss–Newton algorithm, which then found the absolute minimum. Their conclusion was that this approach was an effective one for this kind of task.

1.3. Kinematic and Kinetic Modeling

Kinematics refers to the study of object movement without taking into account the forces acting on it. An n segmented inverted pendulum can be considered as a kinematic chain (parts serially connected by joints). Each element can be defined as a rigid body defining a geometric relationship between two joints [25]. Based on these assumptions and on Natsakis' course [26], an n segmented inverted pendulum kinematic model can be looked at as a one degree of freedom joint series of n elements connected on $n - 1$ links with the length considered to be zero. The axis of a joint is determined by the rotation of link i in relation to link $i - 1$. The distance between two different axes can be measured by determining the common perpendicular on them. If two axes are parallel, they can describe an infinite number of common perpendiculars, but all with the same length.

The forward kinematics model describes the relation between variable orientation or displacement inputs for each joint and the position and orientation of the end effector, represented in a 4×4 homogeneous transformation matrix. There are several approaches for computing the forward kinematics model, but in this paper, we will discuss the Denavit–Hartenberg convention [27,28]. The coordinate system for each link is defined by the following rules: the rotation axis of the joint represents the Z axis; the perpendicular on the plane formed by the current joint Z axis and the following joint Z axis represents the current joint X axis. The convention is based on four parameters, set out in Table 1 after defining the coordinate system.

Table 1. Denavit–Hartenberg parameters.

No.	Symbol	Description
1	r_{i-1}	the distance between axes Z_{i-1} and Z_i measured on the X_{i-1} axis
2	α_{i-1}	the angle between axes Z_{i-1} and Z_i measured around X_{i-1}
3	d_i	the distance between axes X_{i-1} and X_i measured on the Z_i axis
4	Θ_i	the angle between axes X_{i-1} and X_i measured around the Z_i axis

Olav et al. proposed the Lagrangian approach to determine the kinetic model [29], but the advantages [30] and limitations of this type of model [31] can be easily found in the literature.

1.4. Kalman Filter

In 1960, R. E. Kalman introduced his famous discrete data filtering technique [32]. The Kalman filter is basically a set of mathematical equations that provides an efficient way of computing the least squares problem using a recursive method. It is very powerful, because it can estimate the future, the present, and the past states of a system, even if its true nature is unknown. The original algorithm is suitable for linear state space models. For nonlinear state space models, the extended Kalman filter was introduced. This algorithm linearizes the operation around a current estimate with the help of partial derivatives [33].

The Kalman filter, even though it was introduced in 1960, is widely used and lately has provided one of the most common ways to minimize the disadvantages [34] associated with strap-down inertial navigation systems [35]. The filtering method requires an accurate dynamic model [36] and observed integration model, including an inertial sensor error stochastic model and a priori details on content regression coefficients between the two systems [37]. However, there are several inconsistencies, as follows: the difference in the linearization approach; precise stochastic modeling that cannot accurately model sensors; the need for stochastic parameters to be adjusted. Each needs a new a priori sensor system and information [38]. In addition, some filtering methods [39,40] were successfully applied.

In the fine alignment process, the error of the inertial sensors is estimated and compensated using the optimum estimation algorithm in order to improve the accuracy of the initial attitude matrix. The most frequently used estimates are based on a Kalman filter, which can handle only linear systems and requires accurate information about noise statistics [41].

Petritoli et al., concentrated on the well-known data fission with integrity monitoring, low cost sensors, and a low energy consumption computer; however, they did not take into account the aging effects of such low cost sensors in depth [42].

The Kalman filter is relatively less mathematically complicated and easier to deploy compared to the other filters, such as the particle filter. However, the capacity of the Kalman filter to position nonlinear integrated systems accurately is limited [42,43]. However, for example, there are also some advantages of using this approach [44]. Eom et al. established a method for the improvement of physical estimates using multiphysical models and Kalman data fusion filters by processing raw measurements within a sensor [45].

1.5. Optimization

Optimization is the most appropriate solution, with a set of restrictions. It is in the nature of humans to try to find an optimal solution for each problem. Due to advances in computing technology and algorithms, large optimization problems can be very easily solved. However, there still exist a great number of problems whose search is very broad, and the classical optimization techniques cannot trace these problems. Metaheuristics are highly useful and always provide an optimum solution to solve these difficult optimization problems [46].

1.5.1. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is one of the optimization methods used in this project and was first presented in [47] by Kennedy and Eberhart. PSO is a swarm intelligence evolutionary algorithm that simulates bird and fish predatory behavior, and due to PSOs being simple in structure, strong maneuverability, easy implementation and other characteristics, they have attracted much attention from scientists and researchers. proposed a new optimization algorithm called particle swarm optimization, which according to the authors, lied somewhere between genetic algorithms and evolutionary algorithms. They proposed a very simple, but effective algorithm that could optimize a wide variety of functions. The developments, applications, and resources of the PSO algorithm, based on a computer science and engineering perspective, were described in [48]. This work also briefly described the inertia weight parameter and the possible need for a constriction factor. PSO has so far been applied successfully in many areas [49–52], and some improved PSO versions have also been studied [52–56]. Basically, the PSO algorithm has been used to find an optimum search space in complex areas by interacting with people in a particle population [57]. A number of problems such as artificial neural network training [58], fuzzy logic control [59], or pattern classification [60] have been successfully addressed.

In [61], a modification of the PSO algorithm was proposed, called adaptive particle swarm optimization, which will be implemented in this project. Their modified algorithm enables automatic control of certain algorithm parameters such as inertia weights and acceleration constants. Other modifications of the original algorithm have been discussed. For example, the work in [62] discussed a modification of the PSO algorithm called the quantum behaved PSO, which relies on the QPSO, but in addition, uses a recombination operator based on interpolation, which generates a new vector of possible solutions. In this article, the author also briefly described the classical QPSO algorithm, which will be implemented in this project with the proposed modifications

Parameter selection has also been subject to extensive research. In [63], the author discussed different parameter selection methods for the PSO algorithm, including previous proposals from researchers, while the work in [64] also discussed the best PSO parameters for different situations.

Shi and Eberhart [65] were the ones that first introduced the basic evolution equations of the algorithm with the inertial weight as the relatively important PSO control parameter, and some research has since been undertaken on the inertia weight's influence on optimization performance. In accordance with Bayesian theory, Zhang et al. designed an adaptive adjustment strategy for the weight of inertia [66], while at the same time fully applying its historical position. Although the convergence accuracy of this enhanced PSO was greater, the rate of convergence was slow.

1.5.2. Differential Evolution

In [67], the authors proposed a new global optimization method called differential evolution. They concluded that this algorithm was superior to Adaptive Simulated Annealing (ASA), as well as the Annealed Nelder–Mead approach (ANM). It was also superior in terms of the ease of use, since only two parameters had to be chosen from a well-defined interval. DE also has its variations, one being the EDE, which is different in terms of the trial population generation.

DE is a search algorithm based on a population that works with a collection of solutions modified over the generations to find better solutions through selection, generation, and replacement schemes [46,68]. DE is an evolutionary approach to complex problems with optimization. The DE is a simple and popular stochastic algorithm based on the population. When measured against the benchmark problem and actual performance optimization issues, DE outperformed other competitive evolutionary algorithms. DE's main drawback, like other stochastic optimization algorithms, is early convergence and stagnation at suboptimal points. Unlike many other evolutionary computation techniques, basic DE is a very simple algorithm, whose implementation in any standard programming language requires just a couple of lines of code. However, while optimizing a wide range of objective

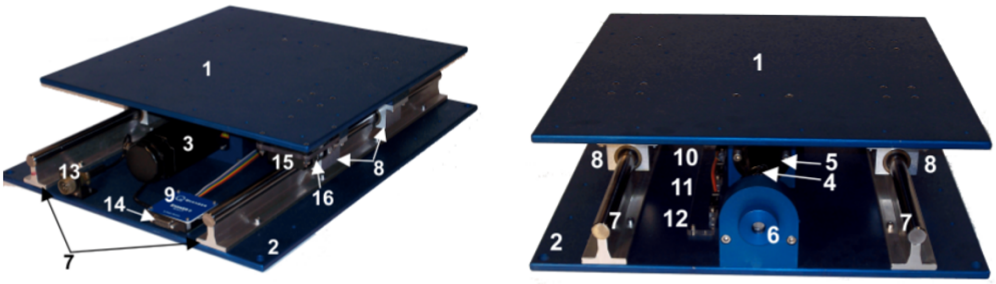
functions, DE shows remarkable performance in terms of ultimate precision, computational speed, and robustness [69].

2. Materials and Methods

2.1. Quanser Shake Table II

The Quanser Shake Table II (STII) is a shake table device for training, which had originally been developed by the University Consortium on Instructional Shake Tables (UCIST). It can be used for educational [70] and research purposes in various topics such as mechanical, aerospace [71] and civil engineering structural dynamics [72], vibration isolation [73], and feedback control [74].

The STII (as shown in Figure 2) [75] has a maximum load of 7.5 kg at an acceleration of 2.5 g (24.525 m/s^2). The staßge is dispatched on two metal shafts with a hard ground with linear rows that allow smooth linear movements with low path deflection. From the center, the stage is able to move ±7.62 cm or ±3 inches (total journey of 15.24 cm). A robust ball screw is connected to a 400 Watt three phase brushless DC actuator. The electric motor has a built-in high-resolution encoder for measuring the stage position at a resolution of 3.10 µm. To measure the step acceleration directly, an analog accelerometer is mounted directly on the stage. Figure 2 lays out the components given in Table 2. Figure 2a provides a cornered perspective of the back, while Figure 2b provides a front view of the shake table.



(**a**) Back corner view (**b**) Front view

Figure 2. Shake Table II components.

Table 2. Shake Table II components.

ID Number	Component	ID Number	Component
1	Stage	9	Sensor circuit board
2	Base plate	10	Right limit sensor
3	DC motor	11	Home position sensor
4	Lead screw	12	Left limit sensor
5	Ball nut	13	Motor leads connector
6	Manual adjustment knob	14	Motor encoder and Hall sensors' connector
7	Linear guide	15	Accelerometer
8	Linear bearing block	16	Accelerometer connectors

2.2. Initial Data

The experiments were performed on a Quanser Shake Table II. The balsa structure was tested on the above mentioned shake table for a given earthquake, and the logged data were the table acceleration reference, the actual acceleration of the table, and the accelerations of the top of the structure. An illustration of the balsa structure, while being tested, can be seen in Figure 3.

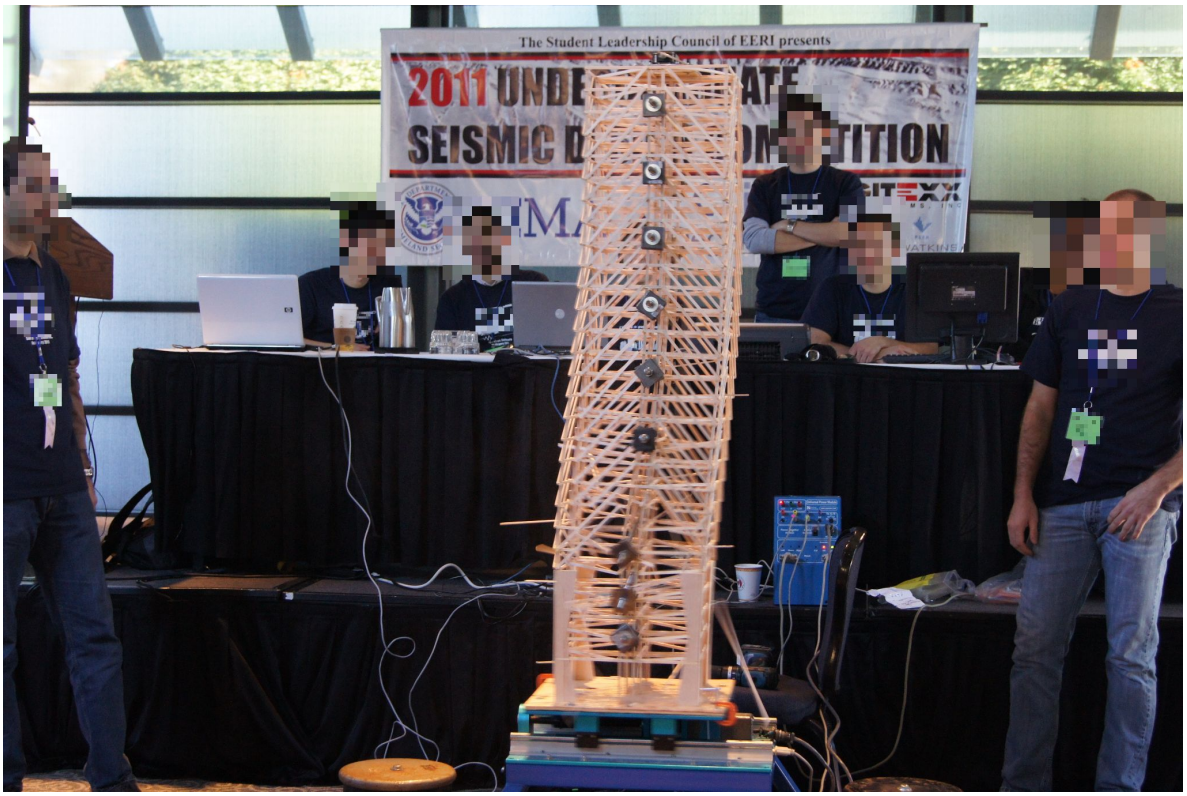


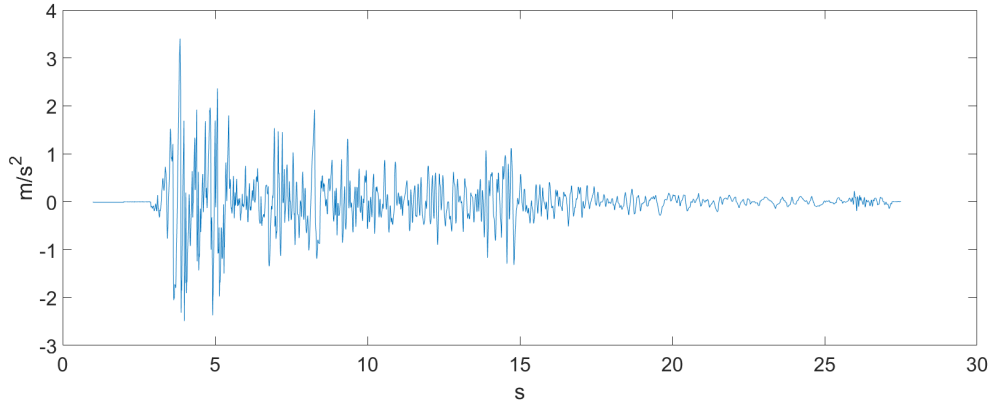
Figure 3. Shake Table II in action.

The sampled data came with a sampling rate of 500 Hz, and both earthquakes, which can be seen in Figures 4 and 5, were the ones proposed in the Seismic Design Competition 2019 organized by the EERI Student Leadership Council https://slc.eeri.org/2019-seismic-design-competition/.

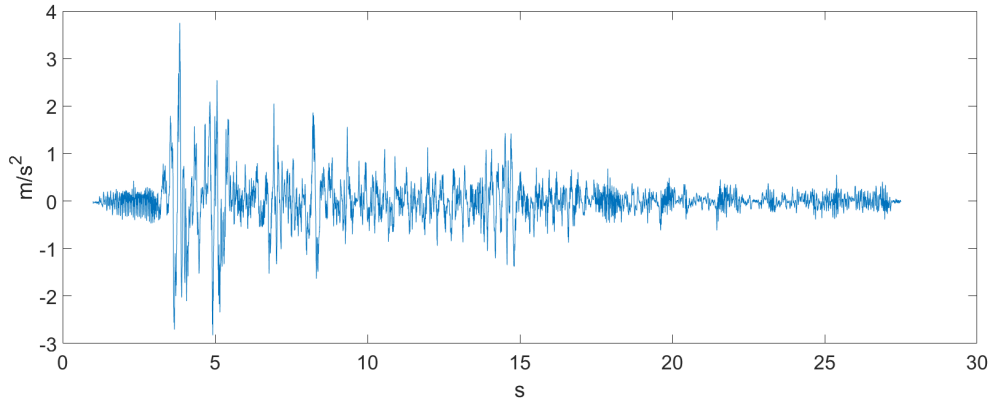
Figure 4 represents Ground Motion 1 of a recorded earthquake. Figure 4a depicts the table acceleration reference, which was the desired displacement of the table, while Figure 4b is the actual table acceleration that was measured with an accelerometer. Figure 5 depicts Ground Motion 2, which is a different, more aggressive earthquake.

The measurements were collected using the Quanser Shake Table II equipment, including both the hardware and software part. The parameters were set to the ones recommended in [75]. The shake table controller also needed the velocity and position setpoints, so for our experiments, we generated data structures from the raw data, which contained:

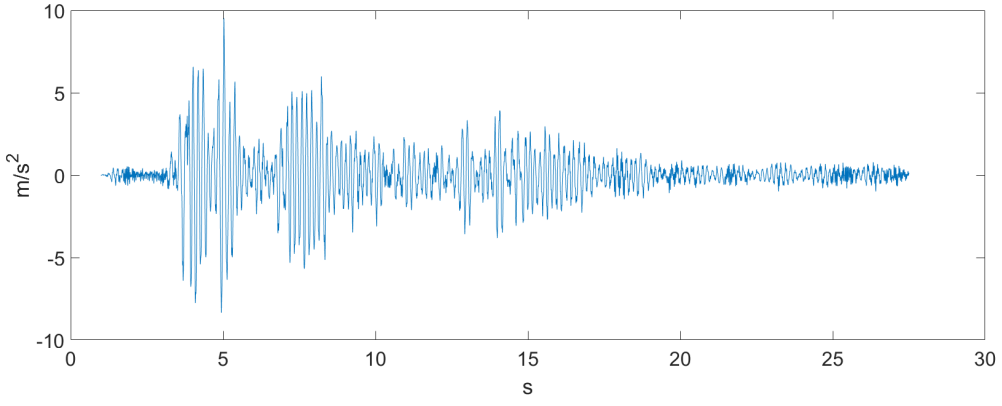
- Sampling time
- Top accelerations
- Actual table's acceleration
- Table's acceleration reference
- Table's velocity reference
- Table's position reference



(**a**) Table's acceleration reference

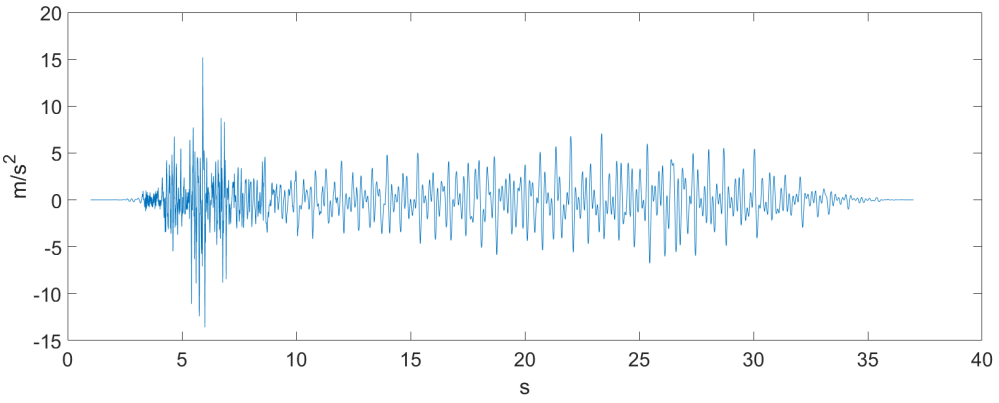


(**b**) Actual table acceleration

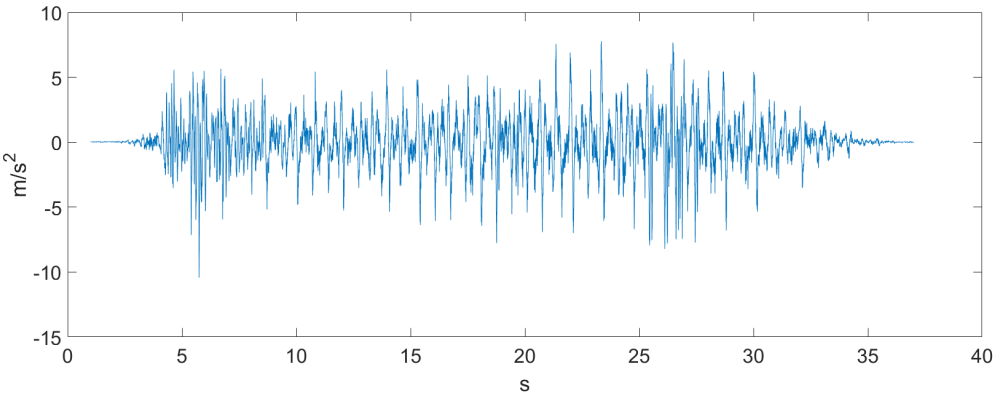


(**c**) Structure's top acceleration

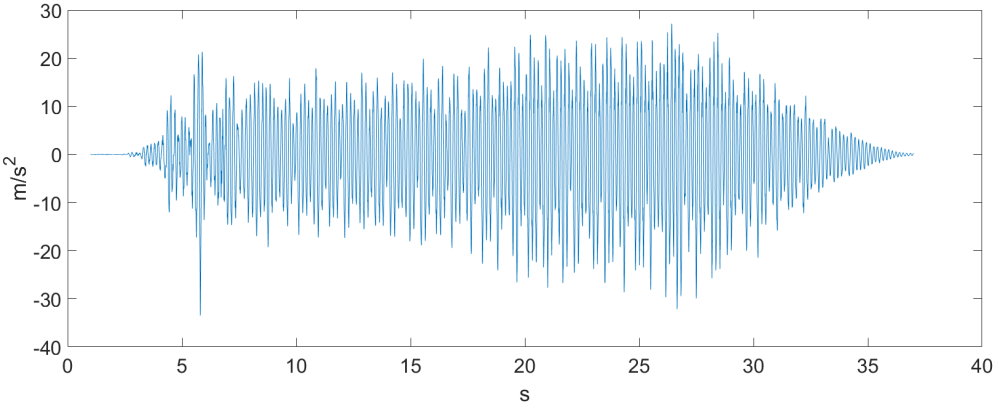
Figure 4. Ground Motion 1.



(**a**) Table's acceleration reference



(**b**) Actual table acceleration



(**c**) Structure's top acceleration

Figure 5. Ground Motion 2.

2.3. Shake Table Controller

The shake table controller design was described in [75]. According to pages 13–15, the controller consisted of a proportional derivative and feed forward controller.

2.3.1. Table Model

The actual table's transfer function can be written in the following format:

$$H(s) = \frac{X(s)}{I(s)} = \frac{1}{K_f s^2} \tag{1}$$

where $X(s)$ is the table displacement, $I(s)$ is the motor current, and:

$$Kf = \frac{M_t P_b}{K_t} \tag{2}$$

is the model gain, where M_t is the total mass being moved by the motor, P_b is the pitch of the ball screw, and K_t is the current-torque coefficient [75].

Since our Lagrangian model, which is discussed later, needed force at the input, the model transfer function becomes:

$$H(s) = \frac{X(s)}{F(s)} = \frac{1}{M_t s^2} \tag{3}$$

2.3.2. Table Controller

As described in [75], the proportional derivative plus feed forward controller had the following form:

$$I(s) = K_p e(s) + K_d e(s)s + K_f e(s)s^2 \tag{4}$$

where K_p, K_d, and K_f are the PD+FF controller gains. Considering that the feed forward element was zero and substituting Equation (1), in order to find the closed-loop transfer function, the equation becomes:

$$K_f s^2 X(s) = K_p(X_d(s) - X(s)) + K_d s(b_{sd} X_d(s) - X(s)) \tag{5}$$

where b_{sd} is the velocity weight coefficient and X_d is the table position reference, which yields the closed-loop transfer function:

$$H(s) = \frac{X(s)}{X_d(s)} = \frac{K_p + K_d b_{sd} s}{K_f(s^2 + \frac{K_d}{K_f}s + \frac{K_p}{K_f})} \tag{6}$$

which is equivalent to a standard second order system if $b_{sd} = 0$:

$$H(s) = \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2} \tag{7}$$

The shake table used in our experiments was calibrated to have a natural frequency of $\omega = 15 \cdot 2 \cdot \pi$ and a damping factor of $\zeta = 0.75$. To match this, we needed the following control gains: $K_p = K_f\omega^2$ and $K_d = 2\zeta\omega K_f$. Since $K_f = 0.5492$, we obtained $K_d = 77.6344$ and $K_p = 4.8779$.

To match the closed-loop system described above for our system, the controller is written in the following form:

$$F(s) = K_p e(s) + K_d e(s)s + K_f e(s)s^2 \tag{8}$$

where substituting Equation (3) and considering the feed forward element to be zero, it becomes:

$$M_t s^2 X(s) = K_p(X_d(s) - X(s)) + K_d s(b_{sd} X_d(s) - X(s)) \tag{9}$$

yielding the closed-loop transfer function:

$$H(s) = \frac{X(s)}{X_d(s)} = \frac{K_p + K_d b_{sd} s}{M_t(s^2 + \frac{K_d}{M_t}s + \frac{K_p}{M_t})} \tag{10}$$

Considering the velocity weight coefficient to be zero, the transfer function matches the standard second order system described in Equation (7). In order to match the same natural frequency and damping factor as stated above, the following control gains were necessary: $K_p = M_t\omega^2$ and $K_d = 2\zeta\omega M_t$. Since $K_f = M_t = 7.74$ (the mass of the table), we obtained $K_d = 1094.21$ and $K_p = 68{,}751.66$.

2.3.3. Filters

In the shake table laboratory guide [75], it was also stated the that direct derivatives from the encoder were not taken in order to avoid noisy signals; instead, the table displacement was filtered with the following second order filters to obtain the stage's velocity and acceleration:

$$H_{f1}(s) = \frac{\dot{X}_f(s)}{X(s)} = \frac{\omega_d^2 s}{s^2 + 2\zeta_d\omega_d s + \omega_d^2} \tag{11}$$

$$H_{f2}(s) = \frac{\ddot{X}_f(s)}{X(s)} = \frac{\omega_f^2 s^2}{s^2 + 2\zeta_f\omega_f s + \omega_f^2} \tag{12}$$

where $\omega_d = 2 \cdot \pi \cdot 50$, $\zeta_d = 0.9$, $\omega_f = 2 \cdot \pi \cdot 25$, $\zeta_f = 0.9$, $\dot{X}_f(s)$ is the filtered velocity and $\ddot{X}_f(s)$ is the filtered acceleration.

2.3.4. Discretization

The controller Equation (8) can also be written in the following form:

$$F(s) = K_p(X_d(s) - X(s)) + K_d(\dot{X}_d - X(s)H_{f1}(s)) + K_f(\ddot{X}_d(s) - X(s)H_{f2}(s)) \tag{13}$$

that is:

$$F(s) = K_p(X_d(s) - X(s)) + K_d(\dot{X}_d(s) - \dot{X}_f(s)) + K_f(\ddot{X}_d(s) - \ddot{X}_f(s)) \tag{14}$$

and by applying the z-transformation:

$$F(z) = K_p(X_d(z) - X(z)) + K_d(\dot{X}_d(z) - \dot{X}_f(z)) + K_f(\ddot{X}_d(z) - \ddot{X}_f(z)) \tag{15}$$

Furthermore, by applying the z-transform to Equations (11) and (12) with the zero order hold method and a sampling interval of 0.002 s, we obtain:

$$H_{f1}(z) = \frac{110.7z - 110.7}{z^2 - 1.094z + 0.3227} \tag{16}$$

$$H_{f2}(z) = \frac{2.467 \cdot 10^4 z^2 - 4.834 \cdot 10^4 z + 2.366 \cdot 10^4}{z^2 - 1.889z + 0.8931} \tag{17}$$

which are equivalent to:

$$\dot{X}_f(z) = 1.094\dot{X}_f(z)z^{-1} - 0.3227\dot{X}_f(z)z^{-2} + 110.7X(z)z^{-1} - 110.7X(z)z^{-2} \tag{18}$$

$$\begin{gathered}\ddot{X}_f(z) = 1.889\ddot{X}_f(z)z^{-1} - 0.8931\ddot{X}_f(z)z^{-2} + 2.467 \cdot 10^4 X(z) \\ -4.834 \cdot 10^4 X(z)z^{-1} + 2.366 \cdot 10^4 X(z)z^{-2}\end{gathered} \tag{19}$$

These yield the final controller's equation:

$$F(k) = K_p(X_d(k) - X(k)) + K_d(\dot{X}_d(k) - \dot{X}_f(k)) + K_f(\ddot{X}_d(k) - \ddot{X}_f(k)),$$

$$\dot{X}_f(k) = 1.094\dot{X}_f(k-1) - 0.3227\dot{X}_f(k-2) + 110.7X(k-1) - 110.7X(k-2) \tag{20}$$

$$\begin{gathered}\ddot{X}_f(z) = 1.889\ddot{X}_f(k-1) - 0.8931\ddot{X}_f(k-2) + 2.467 \cdot 10^4 X(k) - \\ 4.834 \cdot 10^4 X(k-1) + 2.366 \cdot 10^4 X(k-2)\end{gathered}$$

A block diagram of the shake table controller can be seen in Figure 6.

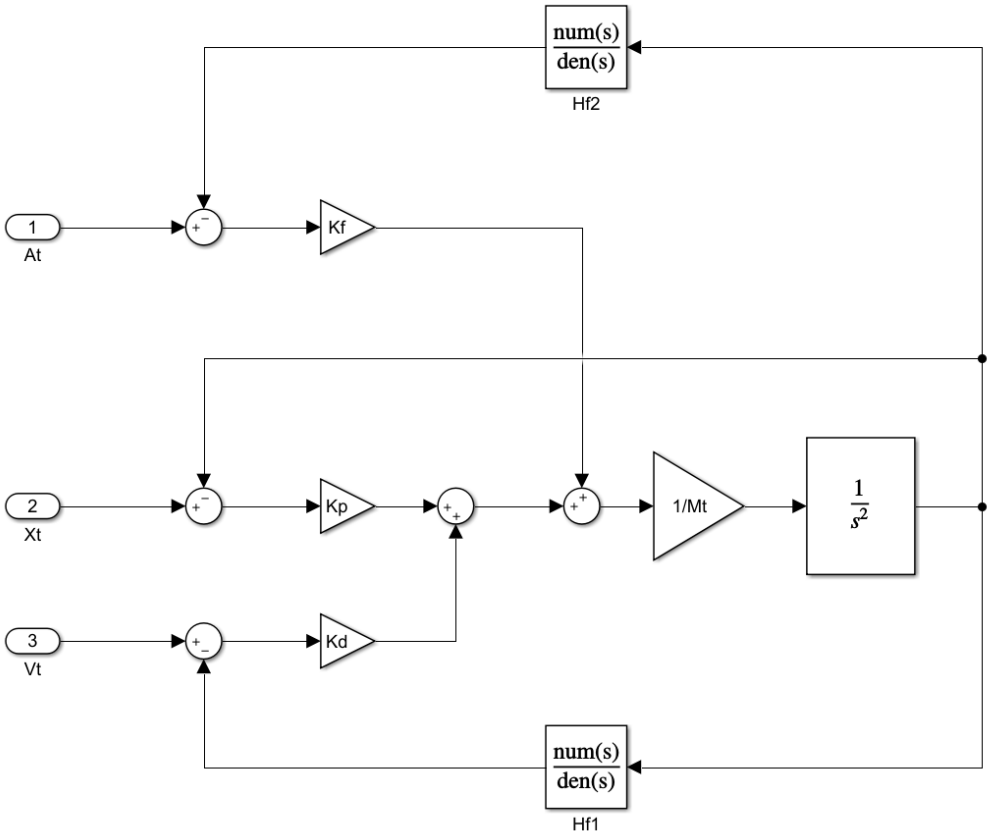


Figure 6. Controller's block diagram.

2.4. Forward Kinematics

2.4.1. Denavit–Hartenberg Parameters

For a multi-segment inverted pendulum on a cart, as seen in Figure 7, the Denavit-Hartenberg (DH) parameters can be seen in Table 3, where q_i $i = 0, 1, ..., n$ are the joints and l_i $i = 0, 1, ..., n$ are the segment lengths.

Table 3. Denavit-Hartenberg (DH) parameters.

q_i	d	r	α	θ
q_0	-	0	$\frac{-\pi}{2}$	$\frac{-\pi}{2}$
q_1	0	0	$\frac{\pi}{2}$	-
q_2	0	l_1	0	-
q_n	0	l_n	0	-

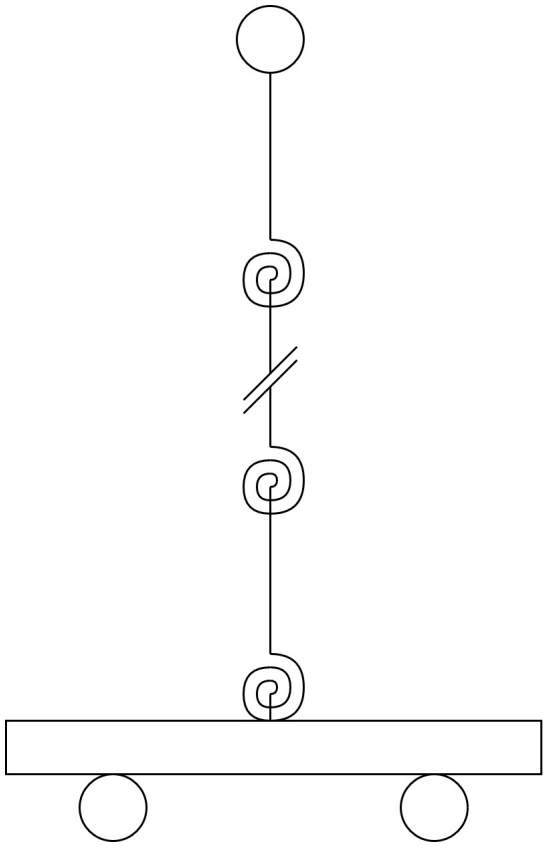


Figure 7. N segment inverted pendulum on a cart with mass-spring-damper joints.

2.5. Forward Kinematic Model

According to [26], having the DH parameters, the direct geometric model [26] is:

$$R_i^{i+1} = Rot(x, \alpha_i) \cdot Trans(x, r_i) \cdot Rot(z, \theta_i) \cdot Trans(z, d_i) \tag{21}$$

that is:

$$R_i^{i+1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & r_i \\ \sin\theta_i \cos\alpha_i & \cos\theta_i \cos\alpha_i & -\sin\alpha_i & -d_i \sin\alpha_i \\ \sin\theta_i \sin\alpha_i & \cos\theta_i \sin\alpha_i & \cos\alpha_i & d_i \cos\alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{22}$$

In our case, it simplifies to:

$$R_0^1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_1^2 = \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin q_1 & \cos q_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_i^{i+1} = \begin{bmatrix} \cos q_i & -\sin q_i & 0 & l_{i-1} \\ \sin q_i & \cos q_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{23}$$

for $i = 2, 3, ..., n$.

Inverse Kinematics Model and the Jacobian

The inverse kinematics model and the Jacobian will not be calculated analytically, because this is too complex and not the purpose of this project; however, we will mention them in the upcoming sections. A brief description [26] of the inverse kinematics model is Equation (24), whereas Equation (25) is that for the Jacobian.

$$q = R_i^0(P_x, P_y, P_z)$$

$$q = [q_1, q_2, ..., q_i]^T \tag{24}$$

$$i = 1, 2, ..., n$$

$$\xi = J\dot{q}$$

$$\xi = [\dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]^T \tag{25}$$

where ξ denotes the linear and angular velocities of the end-effector, in our case the top of our model, and q denotes the joint positions.

2.6. Dynamic Modeling

The dynamic model of the kinematic chain described above was obtained using a Lagrangian-based approach, because it is suitable for complex kinetic chains. This type of dynamic model is based on substituting the Lagrangian of the system, Equation (26), into Equation (27):

$$L = K - P \tag{26}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau \tag{27}$$

Equation (27) can also be written in a more condensed form [26]:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau$$

$$D(q) = \sum_{i=1}^{n} [m_i J_{vi}^T J_{vi} + J_{\omega i}^T R_i I_i R_i^T J_{\omega i}]$$

$$C_{kj}(q) = \sum_{i=1}^{n} \frac{1}{2} \left(\frac{\partial d_{kj}}{\partial q_j} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right) \tag{28}$$

$$G(q) = \frac{\partial P}{\partial q}$$

$$P = \sum_{i=1}^{n} g h_i m_i$$

where I_i is the inertia tensor matrix of a joint and h_i is the height of the joint related mass. The D matrix contains the terms related to the inertia of the system; C contains the terms related to the centrifugal and Coriolis terms; and the G matrix contains the terms related to the potential energies of the system; in our case, gravity. However, our system contained elastic and viscous damping forces due to the mass-spring-damper joint, which is why we introduced the K and B matrices, so that Equation (28) becomes:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + Kq + B\dot{q} = \tau$$

$$K = \begin{bmatrix} 0 & 0 & ... & 0 \\ 0 & k_1 & ... & 0 \\ . & . & ... & . \\ . & . & ... & . \\ . & . & ... & . \\ 0 & 0 & ... & k_n \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & ... & 0 \\ 0 & b_1 & ... & 0 \\ . & . & ... & . \\ . & . & ... & . \\ . & . & ... & . \\ 0 & 0 & ... & b_n \end{bmatrix} \tag{29}$$

where k_i are the elastic coefficients and b_i are the viscous damping coefficients, $i = 1, ..., n$. This can be written in a more usable format:

$$\ddot{q} = D^{-1}(q)(\tau - C(q,\dot{q})\dot{q} - G(q) - Kq - B\dot{q}) \tag{30}$$

In our project matrices, D, C, and G not only depended on q and $\dot{q}$, but also on l, the segment lengths, and m, the segment's weights, because these are also parameters that have to be estimated later.

Continuous Model

Equation (30) can also be written in a state-space format, which will be used later on for the simulation:

$$\dot{x} = A(q, \dot{q}, m, l)x + B(q, m, l)u - \begin{bmatrix} 0_{1\times n} & G(q, m, l) \end{bmatrix}^T$$

$$x = Cx + Du$$

$$x = \begin{bmatrix} q_0 & q_1 & . & . & q_n & \dot{q_0} & \dot{q_1} & . & . & \dot{q_n} \end{bmatrix}^T$$

$$A(q, \dot{q}, m, l) = \begin{bmatrix} 0_{n\times n} & I_n \\ -D^{-1}(q, m, l)K & -D^{-1}(q, m, l)(C(q, \dot{q}, m, l) + B) \end{bmatrix} \tag{31}$$

$$B(q, m, l) = \begin{bmatrix} 0_{n\times 1} \\ D^{-1}(q, m, l) \end{bmatrix}$$

$$C = I_{n\times n}$$

$$D = 0_{n\times 1}$$

2.6.1. Discrete Model

By considering the approximation of the derivative:

$$\dot{x} \approx \frac{x_{k+1} - x_k}{T_s} \tag{32}$$

we can write:

$$x_{k+1} = x_k + T_s\dot{x} \tag{33}$$

that is:

$$x_{k+1} = x_k + T_s(A(q, \dot{q})x + B(q)u - \begin{bmatrix} 0_{1\times n} & G(q) \end{bmatrix}^T) \tag{34}$$

where T_s is the sampling time.

2.7. Objective Function

Having the model of the controller and the dynamic model of the multi-segment inverted pendulum on a cart, we could simulate the behavior of our model for any given input set. However, our model was only an analogy to a real structural behavior, so we could not approximate non-zero initial conditions, this being an important criterion. A brief description of the objective function algorithm can be seen in Algorithm 1.

Algorithm 1: Objective function.

Input : $k, b, m, l, data, K$
Output: fit

initialize $states, x_t, v_t, a_t, x_r, e, c$;
for $u_i \in data$ **do**
  $\tau \leftarrow \begin{bmatrix} c^{i-1} & 0_n \end{bmatrix}^T$
  $states^i \leftarrow Discrete\_Model(\tau, states^{i-1}, k, b, m, l)$
  $\begin{bmatrix} q^i & \dot{q}^i \end{bmatrix}^T = states^i$
  $x_t^i = q_0^i$;
  $v_t^i = 1.094v_t^{i-1} - 0.3227v_t^{i-2} + 110.7x_t^{i-1} - 110.7x_t^{i-2}$
  $a_t^i = 1.493a_t^{i-1} - 0.5681a_t^{i-2} + 2.467 \cdot 10^4 x_t^i - 4.834 \cdot 10^4 x_t^{i-1} + 2.366 \cdot 10^4 x_t^{i-2}$
  $e^i = \begin{bmatrix} x_u^i & v_u^i & a_u^i \end{bmatrix}^T - \begin{bmatrix} x_t^i & v_t^i & a_t^i \end{bmatrix}^T$
  $c^i = K \cdot e$
  $x_r^i = Transformation\_Matrix(q^i)$
end
$a_r = \frac{d^2 x_r}{dt^2}$
$a_t' = \frac{d^2 x_t}{dt^2}$
$fit = \frac{\|y_1 - a_r\|}{\|y_1 - \overline{y_1}\|} + \frac{\|y_2 - a_t'\|}{\|y_2 - \overline{y_2}\|}$

The fit of the objective function was the sum of the normalized mean squared error of the table and roof accelerations between the model and data.

2.8. Prediction

One goal of our project was to test our model for prediction and state estimation. Our choice was the extended Kalman filter, because of its performance and ease of implementation.

2.8.1. Extended Kalman Filter

The extended Kalman filter is presented in Algorithm 2.

Algorithm 2: Extended Kalman filter.

Input : $data, model\_parameters$
Output: $xpred$

for $u_k \in data$ **do**
  $K = P_{pred} C^T (CPC^T + R)^{-1}$
  $\widehat{x_k} = xpred_{k-1} + K(y(k-1) - Cxpred_{k-1})$
  $P = (I - KC)P_{pred}$
  $xpred_k = Discrete\_Model(\widehat{x_{k-1}}, u_{k-1})$
  $F_k = \frac{\partial Discrete\_Model}{\partial x}|_{\widehat{x_{k-1}}, u_{k-1}}$
  $P_{pred} = F_k P F_k^T + Q$
end

2.9. Optimization Stopping Criterion

The stopping criterion used in these optimization algorithms for this project were very simple. Since we have been talking about the model fitness of some data and the objective value of the function was the normalized mean squared error, we could formulate the stopping criterion based on this value.

For example, if we wanted a fit greater than or equal to 90%, the NMSE should be less than or equal to 0.1. This stopping criterion was simple and straightforward; however, our algorithm could be caught in an infinite loop if the population converged to a local minimum. That is why another criterion was inserted: the maximum number of generations or iterations. In this project, the stopping criterion consisted of a maximum number of 500 or 1000 generations and a fit of 90%.

2.10. Optimization Constraint Handling

The constraints presented in this project were linear constraints of the following form: $A \cdot x \leq B$, where A is $nc \times dim$, nc being the number of constraints and dim the dimension of the problem or the number of parameters to be optimized.

The constraint handling technique was also simple and straightforward and consisted of Algorithm 3.

Algorithm 3: Constraint handling.

```
if A · x ≤ B then
    x is in the feasible domain
else
    x is not in the feasible domain
end
```

In our case, x was in the feasible domain, which meant that the function would be evaluated in x, and in the case of PSO, it had the chance of being the global attractor, while in DE, it meant that it had the chance of being present in the next generation of candidates. X not being in the feasible domain meant that the function would not be evaluated at that point, so in the case of PSO, it would never have the chance of being the global attractor, while in DE, this meant that it would not survive to the next generation.

2.11. Particle Swarm Optimization

The traditional particle swarm optimization algorithm was one of the optimization algorithms used in this project. A brief description of the algorithm is presented in Algorithm 4.

The Ackley function, seen in Figure 8, is a widely used benchmark function for optimization algorithms, because it has many local minimum points. Figure 9 presents the evolution of candidates of the PSO algorithm on the Ackley function. The algorithm parameters were set to $\omega = 0.749$, $c_1 = 1.494$, and $c_2 = 1.494$ with a population size of 20. The candidates were generated in the interval of $[-32, 32]$ for each dimension, and no constraints were used. As illustrated in Figure 8, the direction gradient and forward direction were different when the dimension of the Ackley function increased [12]. The global algorithm convergence speed could be detected by this function.

The evolution of the candidates, presented on Figure 9, is illustrated in three subfigures. Figure 9a represents the initial candidate positions, which should be randomly distributed. Knowing that the minimum of Ackley's function is in $[0, 0]$, it is clear in Figure 9b that by the 15th iteration, the candidates were approaching this point. Figure 9c represents the candidates position in the 50th iteration, and it was clear that only a few candidates did not manage to find the minimum.

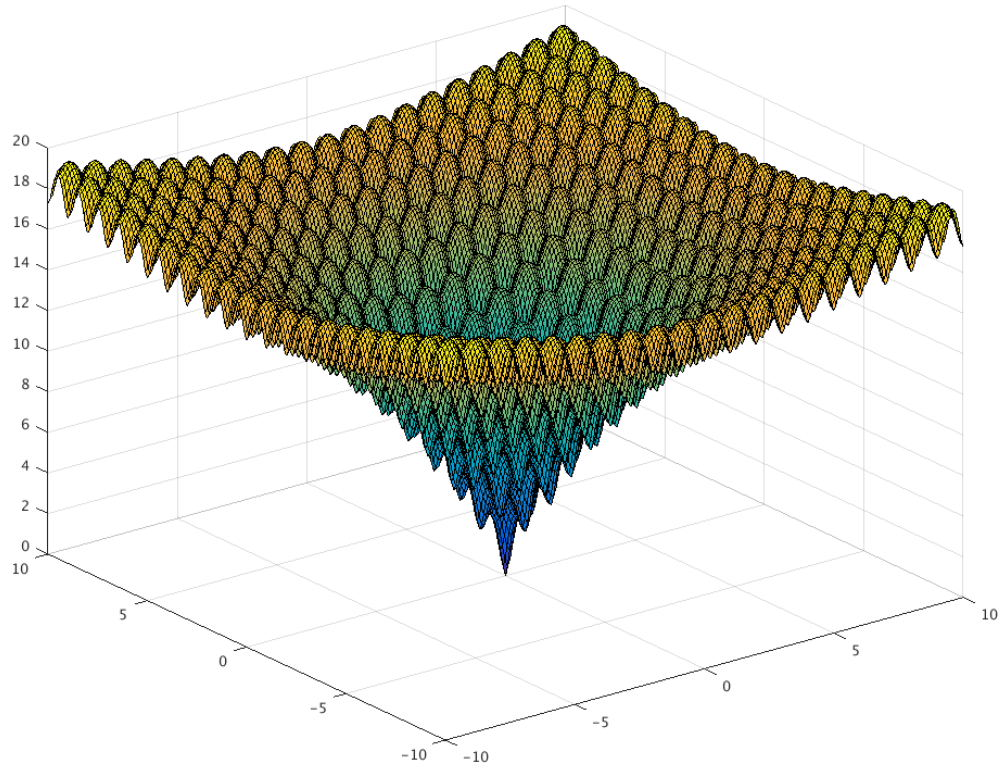


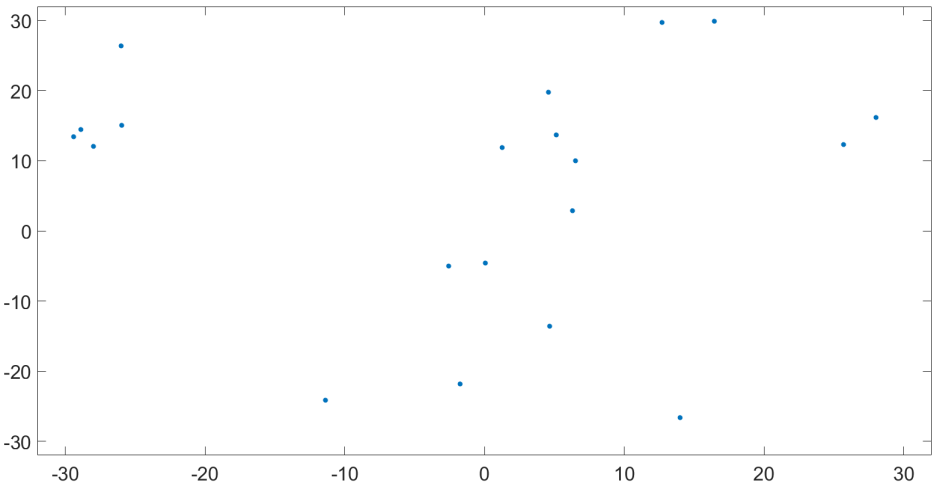
Figure 8. Three-dimensional graph of the Ackley function.

The variables are the following: f is the objective function; lim is a vector of the initial particle limits; A, B are the constraint matrices; dim is the number of dimensions; n is the number of particles; x_i is a particle; p_i is x_i's best known position; g is the global best position; c_1 is the cognitive component; $c2$ is the social component; and ω is the inertia weight.

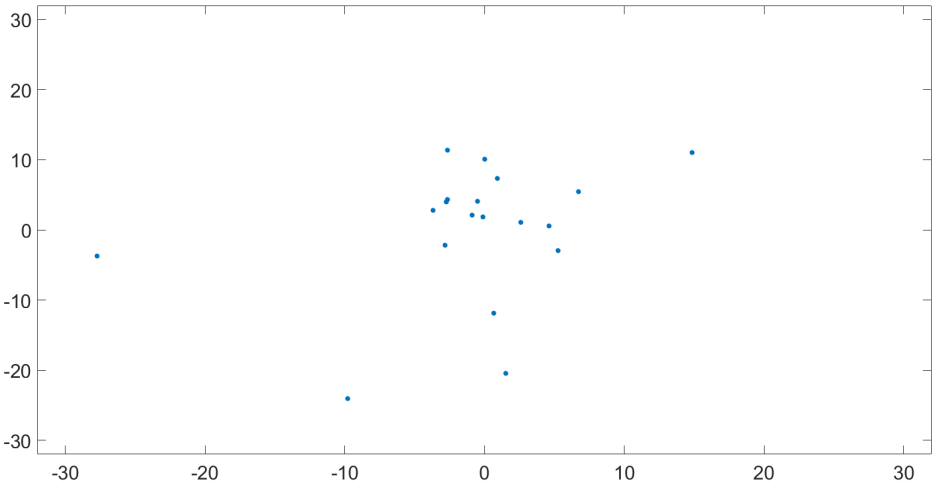
The inertia weight, social component, and cognitive component were selected from [63], in which different proposals were discussed. This paper discussed four scenarios, which can be seen on Table 4. The population size, according to [76], was not so sensitive to the problems; however a population between 20 and 50 is usually used, except for applications with special needs in this matter.

Table 4. Proposed PSO parameters.

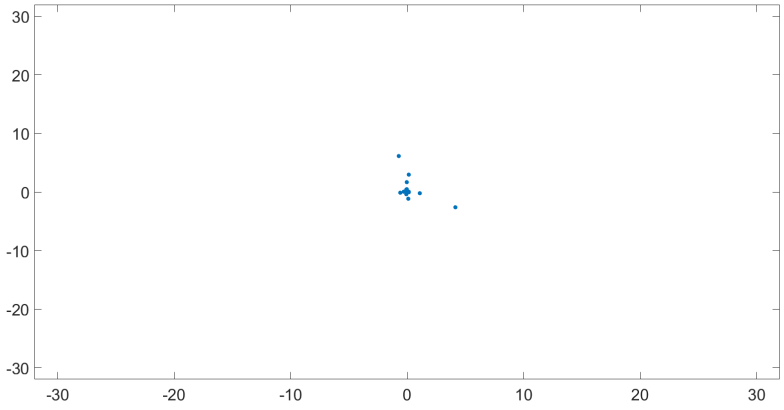
	ω	c_1	c_2
Clerc and Kennedy	0.729	1.494	1.494
Trelea	0.6	1.7	1.7
Carlisle and Dozier	0.729	2.041	0.948
Jiang, Luo, and Yang	0.715	1.7	1.7



(**a**) Iteration No. 1



(**b**) Iteration No. 15



(**c**) Iteration No. 50

Figure 9. PSO swarm behavior on Ackley's function.

Algorithm 4: Particle swarm optimization.

Input : $f, lim, A, B, dim, n, c_1, c_2, \omega, it_{max}$
Output: g

initialize f_g
for $x_i \in 1, 2, ..., n$ **do**
  while $A \cdot x_i <= B$ **do**
    $x_i = lim_{lo} + lim_{up} \cdot rand()$
  end
  $p_i = x_i$
  if $fp_i < fg$ **then**
    $fg = fp_i$
    $g = p_i$
  end
  $v_i = -|lim_{up} - lim_{lo}| + 2 \cdot |lim_{up} - lim_{lo}| \cdot rand()$
end
while $i < it_{max}$ **do**
  for $i \in 1, 2, ..., n$ **do**
    for $d \in 1, 2, ..., dim$ **do**
      $r_g = rand()$
      $r_p = rand()$
      $v_{i,d} = \omega v_{i,d} + c_1 r_p (p_{i,d} - x_{i,d}) + c_2 r_g (g_d - x_{i,d})$
    end
    $x_i = x_i + v_i$
  end
  for $i \in 1, 2, ..., n$ **do**
    if $A \cdot x_i <= B$ **then**
      $fx = f(x_i)$
      if $fx < fp_i$ **then**
        $fp_i = fx$
        $p_i = x_i$
        if $fp_i < fg$ **then**
          $fg = fp_i$
          $g = p_i$
        end
      end
    end
  end
end

2.12. Differential Evolution

As stated in the previous section, in [67], a new global optimization method was proposed called differential evolution. A brief description of the algorithm is presented in Algorithm 5, where f is the objective function, lim is a vector containing the initial candidate region, A and B are the constraint matrices, dim is the dimension of the objective function, n is the population size, F is the differential weight, Cr is the crossover probability, and it_{max} is the maximum number of iterations. In the algorithm, x_i denotes the ith member of the population and s_i denotes the ith member of the trial population, whereas g is the global best. In this algorithm, the global best did not influence the

candidates behavior; it was present only for storing and returning the best possible solution.

Algorithm 5: Differential evolution.

Input : $f, lim, A, B, dim, n, F, Cr, it_{max}$
Output: g

initialize f_g
for $x_i \in 1, 2, ..., n$ **do**
  while $A \cdot x_i <= B$ **do**
    $x_i = lim_{lo} + lim_{up} \cdot rand()$
  end
  $fx_i = f(x_i)$
  if $fx_i < fg$ **then**
    $fg = fx_i$
    $g = x_i$
  end
end
while $i < it_{max}$ **do**
  for $i \in 1, 2, ..., n$ **do**
    $a = rand \in 1, 2, ..., n$
    $b = rand \in 1, 2, ..., n$
    $c = rand \in 1, 2, ..., n$
    $R = rand \in 1, 2, ..., n$
    for $d \in 1, 2, ..., dim$ **do**
      $r = rand \in [0, 1]$
      if $R = d || r < Cr$ **then**
        $s_{i,d} = x_{a,d} + F \cdot (x_{b,d} - x_{c,d})$
      else
        $s_{i,d} = x_{i,d}$
      end
    end
  end
  for $i \in 1, 2, ..., n$ **do**
    if $A \cdot s_i <= B$ **then**
      $fs_i = f(s_i)$
      if $fs_i < fx_i$ **then**
        $fx_i = fs_i$
        $x_i = s_i$
        if $fx_i < fg$ **then**
          $fg = fx_i$
          $g = x_i$
        end
      end
    end
  end
end

The parameter selection for this algorithm could be performed in many different ways, the best one being a process of meta-optimization. This meant optimizing the algorithm using another optimization algorithm. However, this was not as simple as it seemed, because these algorithms were stochastic;

therefore, they would always find the minimum of the function in a different number of iterations. This was why in this project, the selected parameters would be the ones proposed in previous research. As stated in [77], one scenario was choosing a population size between $5 \cdot dim$ and $20 \cdot dim$ with a differential weight F of 0.5. Another scenario, according to [78], was to select a population size between $3 \cdot dim$ and $8 \cdot dim$ with a differential weight of 0.6 and the crossover probability bounded between [0.3,0.9]. Furthermore, in [79], it was advised that $F \in [0.4, 0.95]$, and as for the crossover probability, it should lie in the range $[0, 0.2]$ if the variables were separable and within $[0.9, 1]$ when the function variables were dependent. In this project, the above mentioned three scenarios were tested.

3. Results

This section discusses the performances of the optimization algorithms, as well as the proposed model using the best parameters obtained after optimization.

The fit is discussed in percentages, using the normalized mean squared error formula.

3.1. Optimization Algorithms

3.1.1. Particle Swarm Optimization

The tested parameters were the ones proposed in Table 4, for a population of 50, respectively 20. Figure 10 illustrates the value of the objective function over the iterations. The mean number of iterations, standard deviation and success rate can be seen on Tables 5 and 6.

Table 5. PSO performance for $n = 50$.

	Mean Iterations	Standard Deviation	Success Rate
Clerc and Kennedy	116.32	42.24	100%
Trelea	90.13	43.0137	100%
Carlisle and Dozier	110.49	43.394	100%
Jiang, Luo, and Yang	150.2323	63.0657	99%

Table 6. PSO performance for $n = 20$.

	Mean Iterations	Standard Deviation	Success Rate
Clerc and Kennedy	184.5	87.33	100%
Trelea	163.9596	88.5733	99%
Carlisle and Dozier	206.1837	98.2121	98%
Jiang, Luo, and Yang	230.8936	91.3517	94%

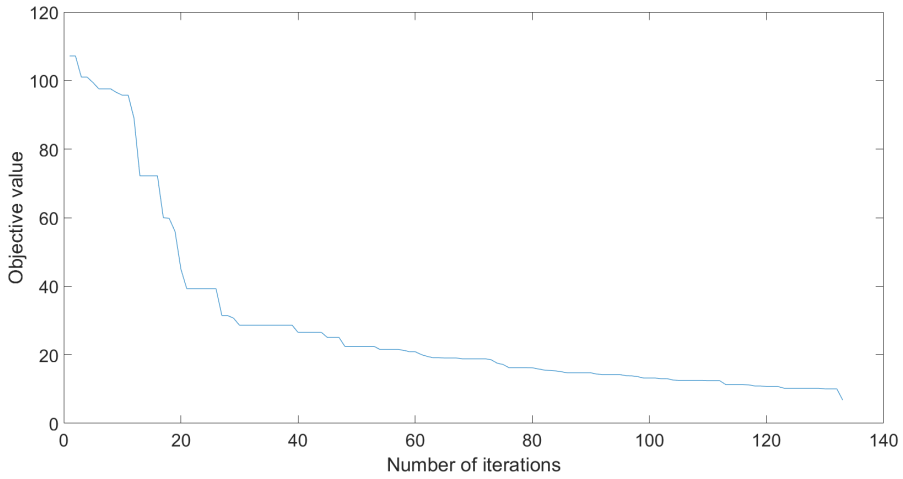


Figure 10. Objective value evolution for PSO.

3.1.2. Differential Evolution

The differential evolution algorithm was tested for a population of 64 ($8 \cdot dim$) 100 times for the parameters seen in Table 7. The evolution of the objective value can be seen in Figure 11.

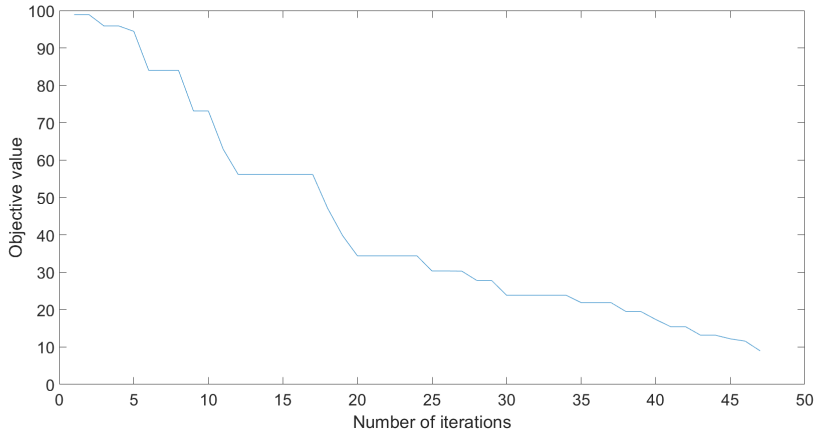


Figure 11. Objective value evolution for DE.

Table 7. DE performance for n = 64.

F	CR	Mean Iterations	Standard Deviation	Success Rate
0.4	0.9	51.1194	7.6228	100%
0.6	0.9	95.2239	12.8261	100%
0.8	0.9	224.4478	27.1624	100%
0.9	0.9	353.1940	42.0434	100%

3.2. Proposed Model Fitness for Prediction

As seen in Figures 12 and 13, the Kalman filter tried to estimate the top displacement of the structure correctly; however, this was not implemented as it should be. As stated, our discrete model returned only angular positions and velocities, but in the dataset, we only had linear accelerations on one axis, which when filtered and integrated, gave us linear positions. The error was calculated between the dataset and the result obtained from the forward kinematic model. However, it would only work correctly if the error was calculated between the dataset and the actual unmodified states of the model.

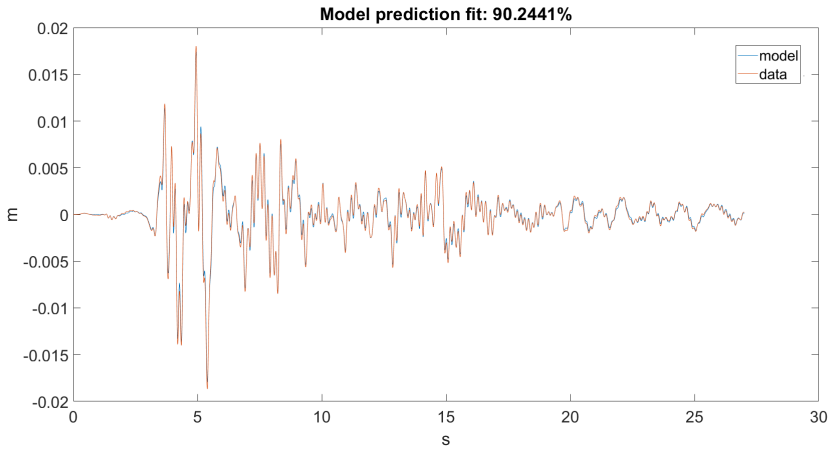


Figure 12. Prediction for Ground Motion 1.

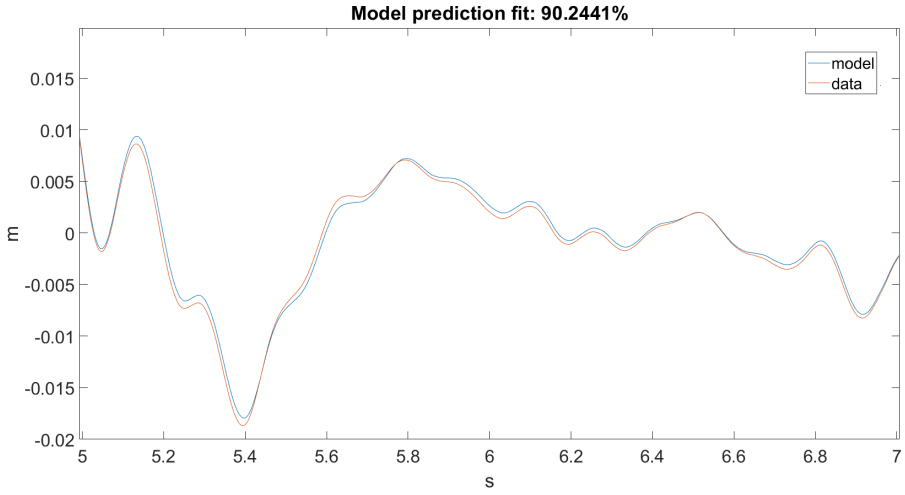


Figure 13. Prediction for Ground Motion 1, zoomed in.

3.3. Proposed Model Fitness for Simulation

As seen in Figure 14, our model returned an acceptable value for Ground Motion 1. Figure 15 zooms in on the previously mentioned figure for a better visualization. Unfortunately, this could not be said about Ground Motion 2, the reason being that the input was much larger in this case. The performances can be seen on Figures 16 and 17.

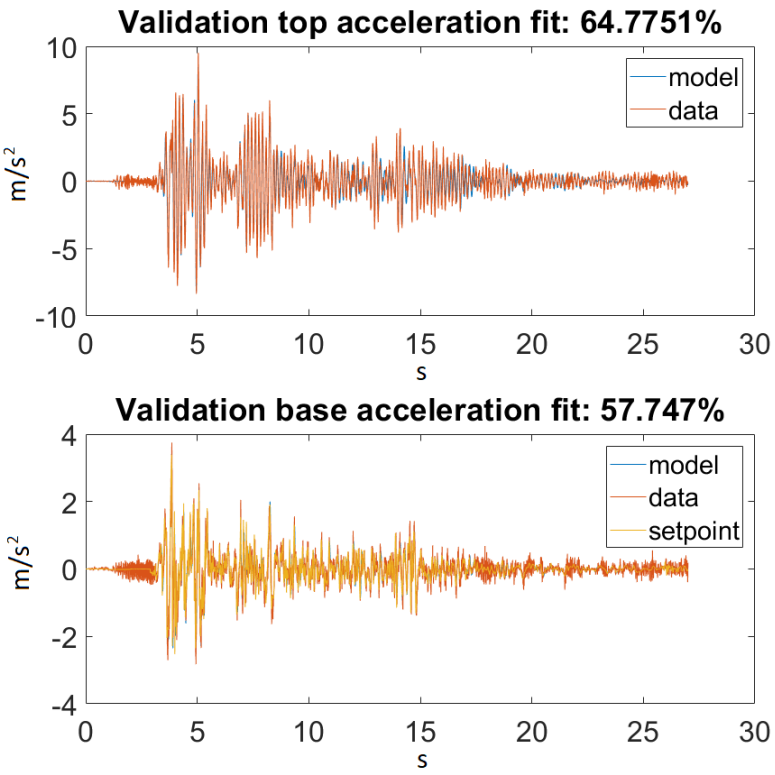


Figure 14. Best fit for Ground Motion 1.

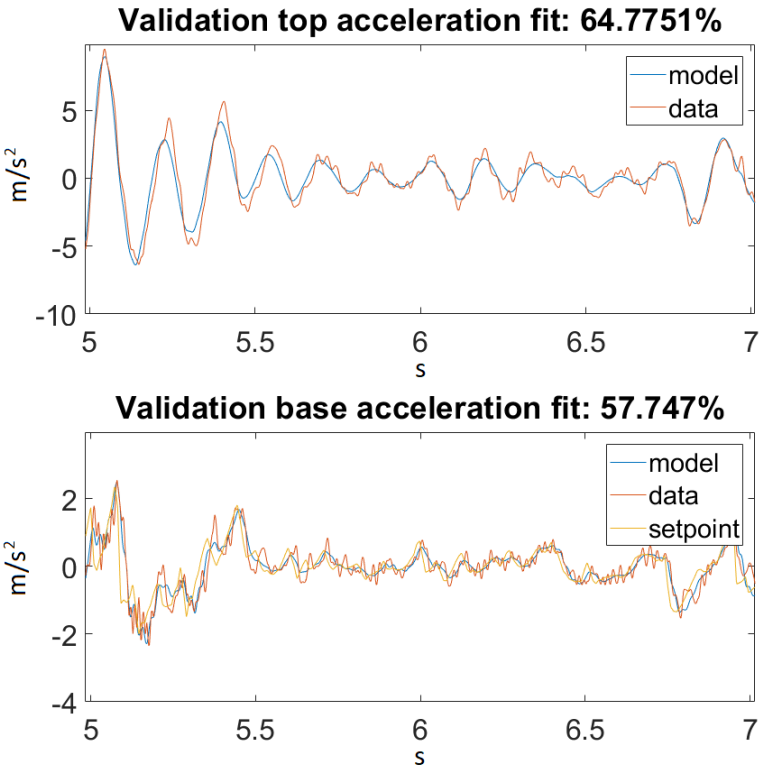


Figure 15. Best fit for Ground Motion 1, zoomed in.

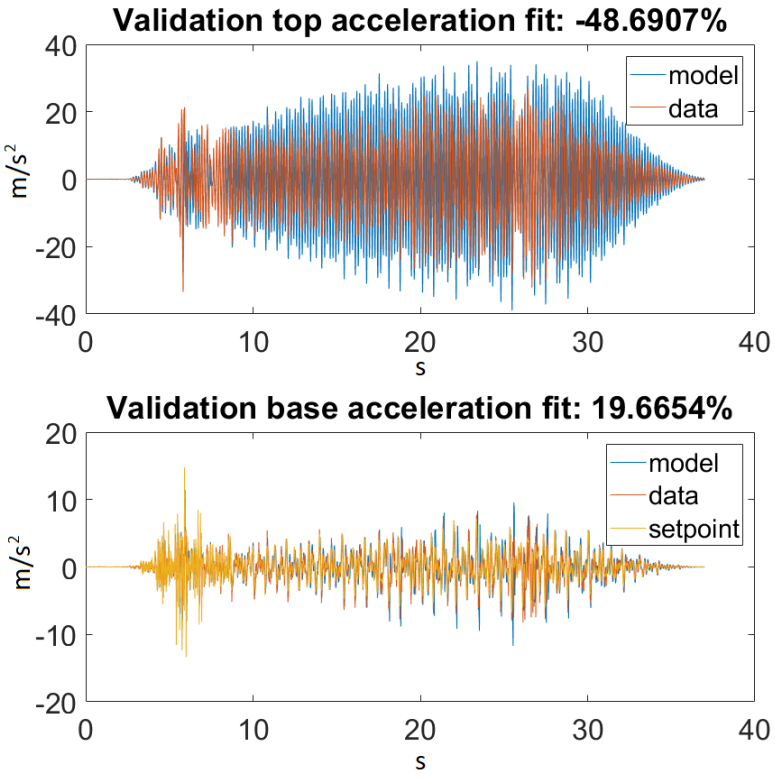


Figure 16. Best fit for Ground Motion 2.

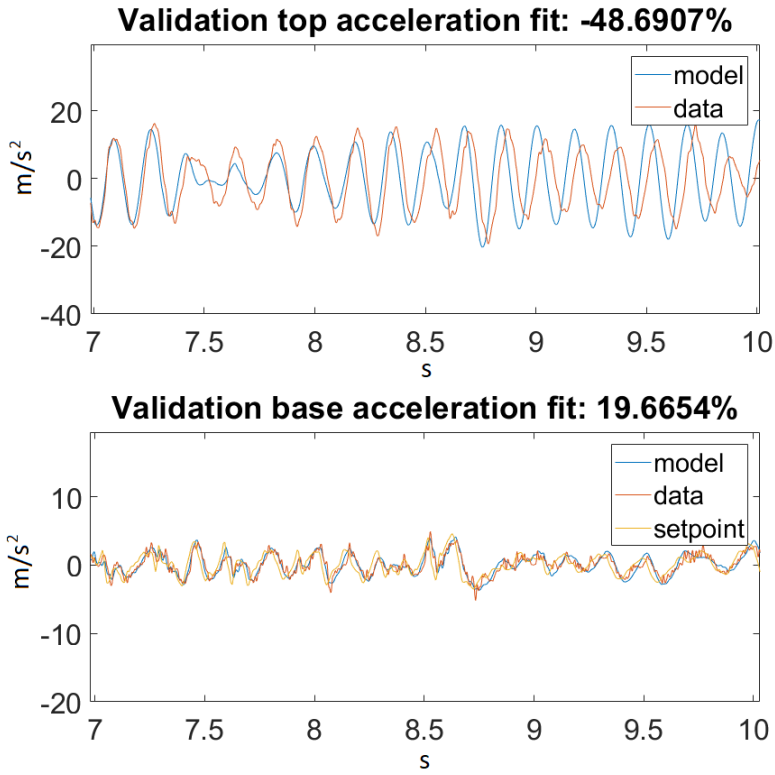


Figure 17. Best fit for Ground Motion 2, zoomed in.

4. Discussion

4.1. Optimization Algorithms

Having discussed the results in the previous section, it was clear that the DE algorithm outperformed its PSO counterpart. The small standard deviation from the mean value is to be noted. Having the figures and tables above, further discussion of these results is not necessary.

4.2. Proposed Model

The most important conclusion that can be drawn is that this kind of model, where material nonlinearity was not included, will never have an extraordinary performance for every input range. For smaller inputs, it could nicely approximate the dynamics of a balsa structure, but this also happened because the material behaved linearly for this magnitude of stress.

On the other hand, taking a look at Figures 14 and 16, one can also observe that for Ground Motion 1, it closely followed the structure dynamics, while for Ground Motion 2, it returned an especially poorly fit value. However, looking closely at Ground Motion 2, one can see that it revealed that the model followed the building dynamics, but it was out of phase. A good model would perform approximately the same for every input range. This confirmed that something was missing from our model, which could be the material elastic or damping nonlinearity.

As for the prediction, the extended Kalman filter nicely predicted the displacement of the top of the system; however, this was not entirely correct. Since in our data, we only had accelerations in the shake axis, we could not directly approximate the states. The first reason was that, as stated before, the data contained some low frequency noise, and the accelerations when integrated yielded extremely unreal displacements. This is why the data were filtered with a second order Butterworth high pass filter at 0.8 Hz, as recommended by the manufacturer. The second reason was that, since we only had acceleration on the y-axis and did not have this on the z-axis, we could not compute the

equivalent angular positions from those data to calculate the error, since our model's states were angular positions and velocities. In the algorithm, the error was calculated between the integrated data and the position calculated from the transformation matrix. This was not entirely accurate, because the extended Kalman filter worked properly only if the error was calculated from the states of the system.

Author Contributions: Conceptualization, L.K. and O.S.; software, L.K.; investigation, L.K.; writing, original draft preparation, L.K. and O.S.; validation, O.S.; funding acquisition, O.S.; formal analysis, L.M.; supervision, L.M.; project administration, L.M. All authors read and agreed to the published version of the manuscript.

Funding: The research presented in this paper was supported by the Robots and Society Cognitive Systems for Personal Robots and Autonomous Vehicles (ROBIN) (PN-III-P1-1.2-PCCDI-2017- 0734) project.

Acknowledgments: Special thanks to the 2019 Seismic Design Competition (SDC) team of the Technical University of Cluj-Napoca for giving access to the shake table data, especially to Bács Béla, as well as to Lucian Busoniu, Zsofia Lendek, and Tassos Natsakis from the Technical University of Cluj-Napoca for their excellent course material in domains related to this project.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PSO	Particle Swarm Optimization
APSO	Adaptive Particle Swarm Optimization
QPSO	Quantum Particle Swarm Optimization
Q-PSO	Quantum behaved Particle Swarm Optimization
PSO-QI	Quantum Infused Particle Swarm Optimization
DE	Differential Evolution
EDE	Elitist Differential Evolution
P	Proportional
PD	Proportional derivative
FF	Feed forward
GA	Genetic algorithm
NMSE	Normalized mean squared error

References

1. Foutch, D.A.; Yun, S.Y. Modeling of steel moment frames for seismic loads. *J. Constr. Steel Res.* **2002**, *58*, 529–564. doi:10.1016/S0143-974X(01)00078-5. [CrossRef]
2. Dyke, S.J.; Spencer, B.F.; Sain, M.K.; Carlson, J.D. Modeling and control of magnetorheological dampers for seismic response reduction. *Smart Mater. Struct.* **1996**, *5*, 565–575. doi:10.1088/0964-1726/5/5/006. [CrossRef]
3. Pratihar, D.K. *Soft Computing: Fundamentals and Applications*; Alpha Science International Ltd.: Oxford, UK, 2014; ISBN-13: 978-1783322053.
4. Sharma, D.; Chandra, P. A comparative analysis of soft computing techniques in software fault prediction model development. *Int. J. Inf. Technol.* **2019**, *11*, 37–46. doi:10.1007/s41870-018-0211-3. [CrossRef]
5. Falcone, R.; Lima, C.; Martinelli, E. Soft computing techniques in structural and earthquake engineering: A literature review. *Eng. Struct.* **2020**, *207*, 110–269. doi:10.1016/j.engstruct.2020.110269. [CrossRef]
6. Shabariram, C.P.; Kannammal, K.E. Earthquake prediction using map reduce framework. In Proceedings of the International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 5–7 January 2017; pp. 1–6. doi:10.1109/ICCCI.2017.8117745. [CrossRef]
7. Sun, B.; Hu, H.; Chen, X. State of the Earthquake Field Disaster Investigation Information Service System. In Proceedings of the 13th Symposium on Piezoelectrcity, Acoustic Waves and Device Applications (SPAWDA), Harbin, China, 11–14 January 2019; pp. 1–5. doi:10.1109/SPAWDA.2019.8681791. [CrossRef]
8. Huang, J.; Wang, X.; Yong, S.; Fenh, Y. A Feature Engineering Framework for Short-term Earthquake Prediction Based on AETA Data. In Proceedings of the IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), Chongqing, China, 24–26 May 2019; pp. 563–566. doi:10.1109/ITAIC.2019.8785773. [CrossRef]

9. Pohsie, G.H.; Chisari, C.; Rinaldin, G.; Amadio, C.; Fragiacomo, M. Optimal design of tuned mass dampers for a multi-storey cross laminated timber building against seismic loads. *Earthq. Eng. Struct. Dyn.* **2016**, *45*, 1977–1995. doi:10.1002/eqe.2736. [CrossRef]
10. Kattamanchi, S.; Tiwari, R.K.; Ramesh, D.S. Non-stationary ETAS to model earthquake occurrences affected by episodic aseismic transients. *Earth Planets Space* **2017**, *69*, 157. doi:10.1186/s40623-017-0741-0. [CrossRef]
11. Alcalde, J.; Bond, C.E.; Johnson, G.; Butler, R.W.H.; Cooper, M.A.; Ellis, J.F. The importance of structural model availability on seismic interpretation. *J. Struct. Geol.* **2017**, *97*, 161–171. doi:10.1016/j.jsg.2017.03.003. [CrossRef]
12. Jiang, M.; Jin, Q. Multivariable System Identification Method Based on Continuous Action Reinforcement Learning Automata. *Processes* **2019**, *7*, 546. doi:10.3390/pr7080546. [CrossRef]
13. Nakayama, M.; Oku, H.; Ushida, S. Closed-loop identification for a continuous-time model of a multivariable dual-rate system with input fast sampling. *IFAC Pap.* **2018**, *51*, 415–420. doi:10.1016/j.ifacol.2018.03.071. [CrossRef]
14. Gumussoy, S.;Ozdemir, A.A.; McKelvey, T.; Ljung, L.; Gibanica, M.; Singh, R. Improving linear state-space models with additional n iterations. *IFAC Pap.* **2018**, *51*, 341–346. doi:10.1016/j.ifacol.2018.09.158. [CrossRef]
15. Pilario, K.E.S.;Cao, Y.; Shafiee, M. Mixed kernel canonical variate dissimilarity analysis for incipient fault monitoring in nonlinear dynamic processes. *Comput. Chem. Eng.* **2019**, *123*, 143–154. doi:10.1016/j.compchemeng.2018.12.027. [CrossRef]
16. Soderstorm, T.; Stoica, P. *System Identification*; Cambridge University Press: Cambridge, UK, 1989. doi:10.1017/S026646660000880X. [CrossRef]
17. Panda, G.; Mohanty, D.; Majhi, B.; Sahoo, G. Identification of nonlinear systems using particle swarm optimization technique. In Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 3253–3257. doi:10.1109/CEC.2007.4424889. [CrossRef]
18. Worden, K.; Barthorpe, R.J.; Cross, E.J.; Dervilis, N.; Holmes, G.R.; Manson, G.; Rogers, R.J. On evolutionary system identification with applications to nonlinear benchmarks, Mechanical Systems and Signal Processing. *Mech. Syst. Signal Process.* **2018**, *112*, 194–232. doi:10.1016/j.ymssp.2018.04.001. [CrossRef]
19. Schoukens, J.; Ljung, L. Nonlinear System Identification A User-Oriented Roadmap. *IEEE Control. Syst. Mag.* **2019**, *39*, 28–99.
20. Liu, J.; Xu, W.; Sun, J. Nonlinear System Identification of Hammerstien and Wiener Model Using Swarm Intelligence. In Proceedings of the 2006 IEEE International Conference on Information Acquisition, Weihai, China, 20–23 August 2006; pp. 1219–1223. doi:10.1109/ICIA.2006.305921. [CrossRef]
21. Hou, Z.-X. Wiener model identification based on adaptive particle swarm optimization. In Proceedings of the 2008 International Conference on Machine Learning and Cybernetics, Kunming, China, 12–15 July 2008; pp. 1041–1045. doi:10.1109/ICMLC.2008.4620558. [CrossRef]
22. Luitel, B.; Venayagamoorthy, G.K. Particle swarm optimization with quantum infusion for system identification. *Eng. Appl. Artif. Intell.* **2010**, *23*, 635 – 649. doi:10.1016/j.engappai.2010.01.022. [CrossRef]
23. Cheng, S.-L.; Hwang, C. Optimal approximation of linear systems by a differential evolution algorithm. *IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum.* **2001**, *31*, 698–707. [CrossRef]
24. Zhao, C.; An, A.; Xu, Q. A hybrid differential evolution algorithm for nonlinear parameter estimation of kinetic systems. In Proceedings of the 2012 24th Chinese Control and Decision Conference (CCDC), Taiyuan, China, 23–25 May 2012; pp. 1696–1700. doi:10.1109/CCDC.2012.6244271. [CrossRef]
25. Lazea, G.h.; Lupu, E.; Dobra, P. *Control Systems and Integrated Manufacturing*; Editura Mediamira: Cluj, Romania, 1998; ISBN 973-9358-22-5.
26. Natsakis, T. Robotic Systems Control. Available online: https://natsakis.com/course/robotic-systems-control/ (accessed on 22 December 2019).
27. Gaidhani, A.; Moon, K.S.; Ozturk, Y.; Lee, S.Q.; Youm, W. Extraction and Analysis of Respiratory Motion Using Wearable Inertial Sensor System during Trunk Motion. *Sensors* **2017**, *17*, 2932. doi:10.3390/s17122932. [CrossRef]
28. Chiang, M.-H.; Lin, H.T. Development of a 3D Parallel Mechanism Robot Arm with Three Vertical-Axial Pneumatic Actuators Combined with a Stereo Vision System. *Sensors* **2011**, *11*, 11476–11494. [CrossRef]
29. Egeland, O.; Gravdah, J.T. *Modeling and Simulation for Automatic Control*; Norwegian University of Science and Technology: Trondheim, Norway, 2013; ISBN 82-92356-01-0.

30. Cheng, C.; Huang, H. Learn the Lagrangian: A Vector-Valued RKHS Approach to Identifying Lagrangian Systems. *IEEE Trans. Cybern.* **2016**, *46*, 3247–3258. [CrossRef]
31. Aggarwal, D.; Kumar, V.; Girdhar, A. Lagrangian relaxation for the vehicle routing problem with time windows. In Proceedings of the International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), Kannur, India, 6–7 July 2017; pp. 1601–1606.
32. Kalman, R.E. A New Approach to Linear Filtering and Prediction Problems. *Res. Inst. Adv. Stud. Baltim. Md* **1960**, *82*, 35–45. [CrossRef]
33. Welch, G.; Bishop, G. An Introduction to the Kalman Filter. 2006. Available online: https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf (accessed on 12 July 2019).
34. Vanicek, P.; Omerbashich, M. Does a navigation algorithm have to use a Kalman filter? *Can. Aeronaut. Space J.* **1999**, *45*, 292–296.
35. Grewal, M.S.; Andrews, A.P. *Kalman Filtering: Theory and Practice Using MATLAB*, 3rd ed.; Wiley: New York, NY, USA, 2001.
36. Agarwal, V.; Arya, H.; Bhaktavatsala, S. Design and development of a real-time DSP and FPGA-based integrated GPS-INS system for compact and low power applications. *IEEE Trans. Aerosp. Electron. Syst.* **2009**, *45*, 443–454. [CrossRef]
37. Noureldin, A.; Karamat, T. B.; Eberts, M.D.; El-Shafie, A. Performance enhancement of MEMS-based INS/GPS integration for low-cost navigation applications. *IEEE Trans. Veh. Technol.* **2009**, *58*, 1077– 1096. [CrossRef]
38. Noureldin, A.; Irvine-halliday, D.; Mintchev, M.P. Accuracy limitations of FOG-based continuous measurement-while-drilling surveying instruments for horizontal wells. *IEEE Trans. Instrum. Meas.* **2001**, *51*, 1177–1191. [CrossRef]
39. Xu, X.; Xu, X.; Zhang, T.; Li, Y.; Tong, J. A Kalman Filter for SINS Self-Alignment Based on Vector Observation. *Sensors* **2017**, *17*, 264. doi:10.3390/s17020264. [CrossRef] [PubMed]
40. Wang, Y.; Sun, F.; Zhang, Y.; Liu, H.; Min, H. Central difference particle filter applied to transfer alignment for SINS on missiles. *IEEE Trans. Aerosp. Electron. Syst.* **2012**, *48*, 375–387. [CrossRef]
41. Juan, L.; Feng, B.; Xu, W. Particle swarm optimization with particles having quantum behavior. In Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004, Portland, OR, USA, 19–23 June 2004; pp. 325–331. doi:10.1109/CEC.2004.1330875. [CrossRef]
42. Petritoli, E.; Giagnacovo, T.; Leccese, F. Lightweight GNSS/IRS Integrated Navigation System for UAV Vehicles. In Proceedings of the 2014 IEEE Metrology for Aerospace (MetroAeroSpace), Benevento, Italy, 29–30 May 2014; pp. 56–61. doi:10.1109/MetroAeroSpace.2014.6865894. [CrossRef]
43. Hinüber, V.; Edgar, L.; Reimer, C.; Schneider, T.; Stock, M. INS/GNSS Integration for Aerobatic Flight Applications and Aircraft Motion Surveying. *Sensors* **2017**, *17*, 941. [CrossRef]
44. Valade, A.; Acco, P.; Grabolosa, P.; Fourniols, J.-Y. A Study about Kalman Filters Applied to Embedded Sensors. *Sensors* **2017**, *17*, 2810. [CrossRef]
45. Eom, K.H.; Lee, S.J.; Kyung, Y.S.; Lee, C.W.; Kim, M.C.; Jung, K.K Improved Kalman Filter Method for Measurement Noise Reduction in Multi Sensor RFID Systems. *Sensors* **2017**, *11*, 10266–10282. [CrossRef]
46. Swagatam, D.; Mullick, S.S.; Suganthan, P.N. Recent advances in differential evolution–an updated survey. *Swarm Evol. Comput.* **2016**, *27*, 1–30. doi:10.1016/j.swevo.2016.01.004. [CrossRef]
47. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; pp. 1942–1948. doi:10.1109/ICNN.1995.488968. [CrossRef]
48. Eberhart, R.C.; Shi, Y. Particle swarm optimization: Developments, applications and resources. In Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546), Seoul, South Korea, 27–30 May 2001; pp. 81–86. doi:10.1109/CEC.2001.934374. [CrossRef]
49. Lin, Q.; Liu, S.; Zhu, Q.; Tang, C.; Song, R.; Chen, J.; Coello, A.A.C.; Wong, K.C.; Zhang, J. Particle swarm optimization with a balanceable fitness estimation for many-objective optimization problems. *IEEE Trans. Evol. Comput.* **2018**, *22*, 32–46. [CrossRef]
50. Deng, X.; Sun, X.; Liu, R.; Liu, S. Consensus control of second-order multiagent systems with particle swarm optimization algorithm. *J. Control. Sci. Eng.* **2018**, *2018*, doi:10.1155/2018/3709421. [CrossRef]

51. Zhang, S.; Xu, J.; Lee, L.H.; Chew, E.P.; Wong, W.P.; Chen, C. Optimal computing budget allocation for particle swarm optimization in stochastic optimization. *IEEE Trans. Evol. Comput.* **2017**, *21*, 206–219. [CrossRef] [PubMed]
52. Liu, Z.; Li, X.; Zhang, H.; Wu, L.; Liu, K. An enhanced approach for parameter estimation: using immune dynamic learning swarm optimization based on multicore architecture. *IEEE Syst. Man Cybern. Mag.* **2016**, *2*, 26–33. [CrossRef]
53. Leboucher, C.; Shin, H.S.; Chelouah, R.; Le Menec, S.; Siarry, P.; Formoso, M.; Tsourdos, A.; Kotenkoff, A. An enhanced particle swarm optimisation method integrated with evolutionary game theory. *IEEE Trans. Games* **2018**, *10*, 221–230. [CrossRef]
54. Bonyadi, M.R.; Michalewicz, Z. Stability analysis of the particle swarm optimization without stagnation assumption. *IEEE Trans. Evol. Comput.* **2016**, *20*, 814–819. [CrossRef]
55. Zhang, H.; Liang, Y.; Zhang, W.; Xu, N.; Guo, Z.; Wu, G. Improved PSO-based method for leak detection and localization in liquid pipelines. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3143–3154. [CrossRef]
56. Ovidiu, S.; Miclea, L.; Sarb, A. Elderly Fall Forecast Based on Adapted Particle Swarm Optimization Algorithm. *Int. J. Model. Optim.* **2018**, *7*, 251–255.
57. Clerc, M.; Kennedy, J. The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Trans. Evol. Comput.* **2002**, *6*, 58–73. [CrossRef]
58. Garro, B.A.; Vazquez, R.A. Designing artificial neural networks using particle swarm optimization algorithms. *Comput. Intell Neurosci.* **2015**, *2015*, 369298, doi:10.1155/2015/369298. [CrossRef]
59. Borni, A. Abdelkrim, T.; Zaghba, L.; Bouchakour, A.; Lakhdari, A.; Zarour, L. Fuzzy logic, PSO based fuzzy logic algorithm and current controls comparative for grid-connected hybrid system. In Proceedings of the AIP Conference Proceedings, Paris, France, 23 February 2017; p. 020006, doi:10.1063/1.4976225. [CrossRef]
60. Tran-Ngoc, H.; Khatir, S.; De Roeck, G.; Bui-Tien, T.; Nguyen-Ngoc, L.; Wahab, M.A. Model Updating for Nam O Bridge Using Particle Swarm Optimization Algorithm and Genetic Algorithm. *Sensors* **2018**, *18*, 4131. doi:10.3390/s18124131. [CrossRef]
61. Zhan, Z.; Zhang, J.; Li, Y.; Chung, H.S. Adaptive Particle Swarm Optimization. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2009**, *39*, 1362–1381. doi:10.1109/TSMCB.2009.2015956. [CrossRef] [PubMed]
62. Pant, M.; Thangaraj, R.; Abraham, A. A new quantum behaved particle swarm optimization. In Proceedings of the GECCO08: Genetic and Evolutionary Computation Conference, Atlanta, GA, USA, 12–16 July 2008; pp. 87–94. doi:10.1145/1389095.1389108. [CrossRef]
63. Stroessner, C. Particle Swarm Optimization & Parameter Selection. Available online: http://wwwmayr.informatik.tu-muenchen.de/konferenzen/Ferienakademie14/slides_papers/\paper_Christoph_Stroessner.pdf (accessed on 14 June 2019).
64. Pedersen, M.E.H. Good Parameters for Particle Swarm Optimization. no. HL1001, 2010. Available online: http://shorturl.at/ANY19 (accessed on 15 January 2020).
65. Shi, Y.H.; Eberhart, R.C. A modified particle swarm optimizer. In Proceedings of the 1998 IEEE International Conferenceon Evolutionary Computation, Anchorage, AK, USA, 4–9 May 1998; pp. 69–73.
66. Zhang, L.; Tang, Y.; hua, C.; Guan, X. A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques. *Appl. Soft Comput.* **2015**, *5*, 138–149. doi:10.1016/j.asoc.2014.11.018. [CrossRef]
67. Storn, R.; Price, K. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. doi:10.1023/A:1008202821328. [CrossRef]
68. Sharma, V.P.; Choudhary, H.R.; Kumar, S.; Choudhary, V. A modified DE: Population or generation based levy flight differential evolution (PGLFDE). In Proceedings of the 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE) Noida, India, 25–27 February 2015; pp. 704–710.
69. Jain, S.; Kumar, S.; Sharma, V.K.; Sharma, H. Improved differential evolution algorithm. In Proceedings of the 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS), Dubai, UAE, 18–20 December 2017; pp. 627–632.
70. Tuhta, S.; Günday, F.; Abrar, O. Experimental study on effect of seismic damper to reduce the dynamic response of bench-scale steel structure model. *Int. J. Adv. Res. Innov. Ideas Educ.* **2020**, *5*, 421–435. doi:10.1016/j.soildyn.2019.03.031. [CrossRef]

71. Tuhta, S.; Günday, F.; Alihassan, A. System Identification of Model Steel Chimney with Fuzzy Logic. *Int. J. Res. Innov. Appl. Sci.* **2020**, *5*, 11–15.
72. Harmanci, Y.E.; Gulan, U.; Holzner, M.; Chatzi, E. A Novel Approach for 3D-Structural Identification through Video Recording: Magnified Tracking. *Sensors* **2019**, *19*, 1229. doi:10.3390/s19051229. [CrossRef]
73. Fu, B.; Jiang, H.; Wu, T. Comparative studies of vibration control effects between structures with particle dampers and tuned liquid dampers using substructure shake table testing methods. *Soil Dyn. Earthq. Eng.* **2019**, *121*, 421–435. doi:10.1016/j.soildyn.2019.03.031. [CrossRef]
74. Li, S.; Sun, L.; Kong, F. Vibration Control Performance Analysis and Shake-Table Test of a Pounding Tuned Rotary Mass Damper under the Earthquake. *Shock Vib.* **2019**, *2019*, doi:10.1155/2019/4038657. [CrossRef]
75. Inc., Q. Shake Table II—Laboratory Guide. User Manual, Markham, Ontario, 2014. Available online: https://www.quanser.com/ (accessed on 12 June 2019).
76. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. doi:10.1007/s00500-016-2474-6. [CrossRef]
77. Storn, R.; Price, K. Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces. *J. Glob. Optim.* **1995**, *23*, doi:10.1.1.1.9696. [CrossRef]
78. Gamperle, R.; Muller, S.D.; Koumoutsakos, P. A Parameter Study for Differential Evolution. *Adv. Intell. Syst. Fuzzy Syst. Evol. Comput.* **2002**, *10*, 293–298.
79. Ronkonnen, J. Continuous Multimodal Global Optimization with Differential Evolution-Based Methods. Ph.D. Thesis, Lappeenranta University of Technology, Lappeenranta, Finland, 2009.