ADAPTIVE STRUCTURAL CONTROL USING DYNAMIC HYPERSPACE

S. LAFLAMME
Department of Civil, Construction, and Environmental Engineering,
Iowa State University, Ames, U.S.A.

ABSTRACT

The design of closed-loop structural control systems necessitates a certain level of robustness to cope with system uncertainties. Neurocontrollers, a type of adaptive control system, have been proposed to cope with those uncertainties. However, the performance of neural networks can be substantially influenced by the choice of the input space, or the hyperspace in which the representation lies. For instance, input selection may influence computation time, adaptation speed, effects of the curse of dimensionality, understanding of the representation, and model complexity. Input space selection is often overlooked in literature, and inputs are traditionally determined offline for an optimized performance of the neurocontroller. Such offline input selection is often unrealistic to conduct in the case of civil structures. In this paper, a novel method for automating the input selection process for neural networks is presented. The method is purposefully designed for online input selection during adaptive identification and control of nonlinear systems. Input selection is conducted online and sequentially, while the excitation is occurring. The algorithm designed for the adaptive input space assumes local quasi-stationarity of the time series, and embeds local maps sequentially in a delay vector using the embedding theorem. The input space of the representation is subsequently updated. The performance of the proposed dynamic input selection method is demonstrated through simulating semi-active control of an existing structure located in Boston, MA, U.S.A. Simulation results show the substantial performance of the proposed algorithm over traditional fixed-inputs strategies.

Keywords: Adaptive control, adaptive hyperspace, adaptive input, online sequential network, self-organizing input, sequential neural network, structural control.

1 INTRODUCTION

Constructing representations for system identification and control of large-scale system is a difficult task, because dynamic parameters are often uncertain. A solution is to use intelligent control, among which neural networks have gained significant popularity due to their universal approximation capability [1, 2]. The performance of black-box representations used in neurocontrol is highly dependant on the selection of the input space [3, 4]. For instance, the choice of neural net inputs may influence computation time, adaptation speed, effects of the curse of dimensionality, understanding of the representation, and model complexity [3, 5, 6].

Several techniques have been proposed to conduct input selection, including filter methods [7], wrapper methods [8], and embedding methods [9], with application to automated input selection for neural networks [4, 10–13]. The majority of applications in literature are offline batch methods where the representation can be trained and evaluated before its utilization, or can be trained sequentially until performance satisfaction is attained. These methods are not applicable to controllers designed for systems that evolve in unknown environments (i.e. civil structures and wind turbines), for which input–output data can hardly be made available, and with an immediate performance requirement on the controller upon the occurrence of external input. Such control problem is specific, and a suitable approach is a sequential adaptive controller, which is herein defined as a black-box controller that constructs the control rule sequentially, while the excitation is occurring. In this case, the input space has to be identified or adapted online to satisfy a given level of control performance.

ISSN: 2046-0546 (paper format), ISSN: 2046-0554 (online), http://www.witpress.com/journals
DOI: 10.2495/CMEM-V3-N1-49-64

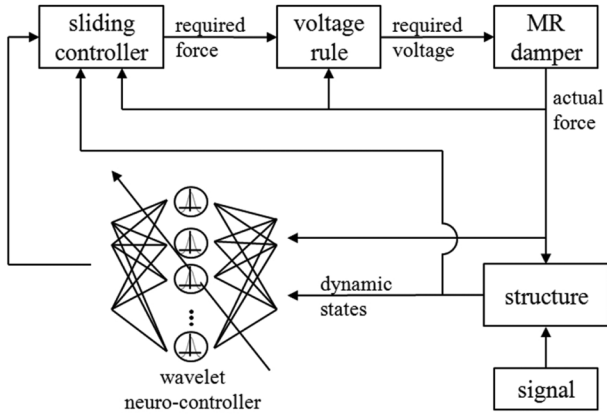


Figure 1: Block diagram of the closed-loop control system.

Input selection based on the embedding methods have emerged from the celebrated Takens embedding theorem [14]. The theorem states that the phase-space of an autonomous system can be reconstructed topologically using a vector formed with a number of delayed measurements from a single state. In other words, there exists a set of inputs, from limited observations, that can represent the system dynamics. The embedding theorem has been extended to a general class of nonautonomous systems with deterministic forcing [15], state-dependant forcing [16], and stochastic forcing [17], and counts numerous applications to analysis and identification of time series [18–20].

This paper presents a novel adaptive neural network for identification and control of unknown systems for which pre-training is not available. It is a short version of the work published earlier by the author and his colleagues [21]. The adaptive neural network is a single-layer wavelet neural network (WNN), with a self-organizing mapping architecture for its hidden layer [22]. The novelty of the neural network is the capacity of the input layer to self-adapt, including the addition and pruning of nodes. The proposed algorithm that selects the input space is termed self-organizing inputs (SOI) algorithm, and is based on Takens embedding theorem. The SOI algorithm determines, at each time steps, the input variables that capture the essential dynamics of the system, and uses the information to smoothly adapt the input space. The proposed SOI–WNN has the substantial benefits of (1) automating the input selection process for time series that is not known a priori; (2) adapting the representation to nonstationarities; and (3) using limited observations.

The paper is organized as follows. Section 2 describes the WNN that is utilized with the SOI algorithm. Section 3 presents the novel SOI algorithm. Section 4 integrates the SOI to the WNN to create the SOI–WNN, and verifies its performance using a simple simulation on tracking of a stationary time series. Section 5 simulates the SOI–WNN on a large-scale structure subjected to wind and equipped with semi-active dampers. Section 6 concludes the paper.

2 WAVELET NEURAL NETWORK

We have presented in Laflamme *et al.* [23] an adaptive WNN augmented by a sliding controller for semi-active control mechanisms. This WNN will be utilized with the novel SOI algorithm. The neural network is summarized in this section, specialized for a nonlinear and nonautonomous controlled systems of the type:

$$\begin{aligned} x(k+1) &= f_x\left(x(k), u(k), k\right) \\ u(k+1) &= f_u\left(y(k), u(k), k\right) \\ y(k) &= f_y\left(x(k), u(k), k\right) \end{aligned} \tag{1}$$

where x denotes the state, u the input, y the observation, f the nonlinear functions, and k the discrete time steps.

Figure 1 shows a representation of the controller. The structure is excited by an external signal and the control device. Its dynamic states, as well as the force inputs, are fed in the adaptive WNN. The WNN outputs are in turn fed in the sliding controller that adjusts the force based on the force reachability of the semi-active devices. A voltage is then sent to the semi-active device based on the required force using a saturation rule, which consists of applying a maximum voltage if the required force is in the same direction and sign as the previously applied force, or no voltage otherwise.

The wavelet neuro-controller is a single-layer feedforward WNN composed of *mexican hat* wavelets ϕ:

$$\phi(\boldsymbol{\zeta}) = \left(1 - \frac{\|\boldsymbol{\zeta} - \boldsymbol{\mu}\|^2}{\boldsymbol{\sigma}^2}\right) e^{-\frac{\|\boldsymbol{\zeta} - \boldsymbol{\mu}\|^2}{\boldsymbol{\sigma}^2}} \tag{2}$$

where $\boldsymbol{\zeta}$ is the input vector, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the centers and bandwidths of the functions respectively, and $\|\cdot\|$ is the 2-norm. The automation of $\boldsymbol{\zeta}$ is discussed in the next section.

2.1 Adaptation rules

The weights, bandwidth, and center of the wavelets are adapted using the back-propagation algorithm for enhanced computation speed. The adaptation rules, in discrete time, are written [23]:

$$\hat{\boldsymbol{\xi}}(k+1) = \hat{\boldsymbol{\xi}}(k) - \Delta t\left(\boldsymbol{\Gamma}_{\xi}\left(\frac{\delta u}{\delta \xi}\right)\text{sign}\left(\boldsymbol{B}^T\boldsymbol{P}^T\right)s\right) \tag{3}$$

where Δt is the discrete time step, $\boldsymbol{\xi} = [\boldsymbol{\gamma}\ \boldsymbol{\mu}\ \boldsymbol{\sigma}]^T$ is the parameter vector, $u = \boldsymbol{\gamma}^T\boldsymbol{\phi}$ is the neural output (here a control force), γ are the weights, $\boldsymbol{B}$ is the state-space input matrix, $\boldsymbol{P}$ represents the user-defined sliding surface weights, $s = \boldsymbol{PX}$ is the sliding surface computed using the state $\boldsymbol{X}$ as the error metric (regulatory problem), $\boldsymbol{\Gamma}$ represents the adaptation weights of the appropriate subscripts, and the hat denotes an estimation.

2.2 Sliding controller

The sliding controller is used for modifying the adaptation mechanism of the WNN when the control force is not reachable. This is the case in semi-active control, because control devices cannot add energy to a system. The force error $\tilde{u} = u_{act} - u_n$ is assumed to be large and bounded, where u_{act} is the actuator force and u_n the force required by the neural network. We define three adaptation regions. First, the region C, which consists of the bounded set of the neural network force output. Second, the region C_d, which is confined within the device reachability. Third, the region C_t, which is the transition region between C_d and C, such that $C \supset C_t \supset C_d$:

$$C_d = \left\{ |\tilde{u}| \leq \alpha_d u_b \,|\, \alpha_d \in [0,1],\, u_b,\, \tilde{u} \in R \right\}$$
$$C_t = \left\{ |\tilde{u}| \leq \alpha_t u_b \,|\, \alpha_t \in [0,1],\, u_b,\, \tilde{u} \in R \right\}$$
$$C = \left\{ |\tilde{u}| \leq u_b \,|\, u_b,\, \tilde{u} \in R \right\}$$

where u_b is a bound on the admissible error $\tilde{u} = u - u_{act}$, u_{act} is the actual force output of the semi-active device, and α_d, α_t are user-defined constants with $\alpha_d \leq \alpha_t$. Subsequently, the control law is modified using an adaptation ratio $(1 - m_b)$ where m_b takes the values:

$$\begin{aligned} m_b &= 0 && \text{if } \tilde{u} \in C_d \\ m_b &= \frac{2e^{-b(|\tilde{u}| - \alpha_d u_b)}}{1 + e^{-b(|\tilde{u}| - \alpha_d u_b)}} && \text{if } \tilde{u} \in C_t - C_d \\ m_b &= 1 && \text{if } \tilde{u} \in C - C_t \end{aligned} \tag{4}$$

where b is a positive constant. Stability of the control law is shown in Laflamme *et al*. [23].

2.3 Self-organizing rules

The hidden layer is organized sequentially following Kohonen's self-organizing mapping theory [22]. A new node is added if:

$$\begin{aligned} \| \boldsymbol{\zeta} - \boldsymbol{\mu} \|^2 &\geq d_{\min} \\ \text{and} \qquad s &\geq s_{\text{all}} \end{aligned} \tag{5}$$

where $d_{\min}$ and s_{all} are the thresholds for the minimum nodal distance to the closest node and minimum allowable error, respectively. Once a node is added, the parameters of the new node i are set to:

$$\begin{aligned} \gamma_i &= s \\ \boldsymbol{\mu}_i &= \boldsymbol{\zeta} \\ \boldsymbol{\sigma}_i &= \lambda \| \boldsymbol{\zeta} - \boldsymbol{\mu} \|^2 \end{aligned} \tag{6}$$

where λ is the network resolution. Note that some bounds exist on the new parameters given that we are using wavelet activation functions. They are not listed here for conciseness; details are given in Laflamme *et al*. [23].

Additionally, nodes can be pruned from the network. A node is pruned if its relative weight with respect to the largest nodal weight falls under a threshold $\gamma_{\min}$ for a given number of consecutive time steps γ_{num}.

3 SELF-ORGANIZING ALGORITHM

The purpose of the SOI algorithm is to sequentially organize the input space of the WNN. The algorithm parameterizes the unknown dynamic system using the time series response of a single observation. Subsequently, the dimension of the input space is adapted smoothly, along with the time lag between observations, using the assumption that the new inputs represent the essential dynamics of the unknown system. The objective is to obtain a more efficient representation for the dynamic system by selecting, at each time step, the inputs that contain a sufficient representation of the current system state.

Takens embedding theorem is applicable to reconstruct the state-space of an autonomous dynamic system. The theorem states that the phase-space of the dynamic system in a topological space M can be reconstructed from a vector v, termed delay vector, constructed with a dimension d of the observations $y(k) = f_y(x(k))$ delayed by a factor τ:

$$\begin{aligned} v(k) &= \left[y(k) \;\; y(k-\tau) \;\; y(k-2\tau) \ldots y(k-(d-1)\tau) \right] \\ &= \Phi(x(k)) \end{aligned} \tag{7}$$

where $\Phi : M \to \mathbb{R}^d$, $\tau = a\Delta t$, with Δt being the sampling rate and a is a positive integer.

Thus, the algorithm takes the delay vector v that would parameterize the reconstructed phase-space, and sequentially adapts the WNN input space $\zeta(\tau_\zeta, d_\zeta)$ smoothly, by using sequential binary changes on and τ_ζ and d_ζ. When d_ζ is modified ($d_\zeta (k+1) \neq d_\zeta (k)$), the modified wavelet bandwidths $\boldsymbol{\sigma}_i$ are adapted smoothly using:

$$\boldsymbol{\sigma}_{\text{mod}} - (\boldsymbol{\sigma}_{\text{mod}} - \boldsymbol{\sigma}_i) m_c(k) \tag{8}$$

where $\boldsymbol{\sigma}_{\text{mod}}$ is a vector of large constants, $\boldsymbol{\sigma}_i$ is the target bandwidths from (6) when a dimension is added or the actual bandwidths to be removed when a dimension is decreased, and $m_c(k)$ takes the values:

$$\begin{aligned} m_c(k) &= \frac{1 - e^{-c(k-k_{\text{mod}})}}{1 + e^{-c(k-k_{\text{mod}})}} \quad \text{if } d_\zeta \text{ is increased} \\ m_c(k) &= \frac{2e^{-c(k-k_{\text{mod}})}}{1 + e^{-c(k-k_{\text{mod}})}} \quad \text{if } d_\zeta \text{ is decreased} \end{aligned} \tag{9}$$

where c is a positive constant, and k_{mod} is the time step when d_ζ has been modified. A dimension is removed from the neural net once the bandwidths fall beyond a threshold. Remark that changes in the input space are restrained to unity, which ensures robustness of the representation, because $y(k - \tau) \simeq y(k - \tau \pm 1)$ and new dimensions are added smoothly based on eqn (9).

Although the theory discussed above applies to autonomous systems, the embedding theorem has been extended to a general class of nonautonomous systems with deterministic forcing [15], state-dependant forcing [16], and stochastic forcing [17]. For nonautonomous stationary systems, it can be shown that the delay vector needs to also include the system inputs u [17]:

$$\begin{aligned} v(k) = \big[& y(k) \;\; y(k-\tau) \;\; y(k-2\tau) \ldots y(k-(d-1)\tau) \\ & u(k) \;\; u(k-\tau) \;\; u(k-2\tau) \ldots u(k-(d-1)\tau) \big] \\ = \Phi & (x(k), u(k)) \end{aligned} \tag{10}$$

where $\Phi : M \to \mathbb{R}^{2d}$. The delay vector may also be constructed by overembedding the state observation y in the case where the input is not observable [20].

Here, the dynamic system of interest (1) is nonstationary. To cope with the problem of nonstationarity, the state dynamics f_x is taken as a series of maps of dimension n, where each map is assumed to be quasi-stationary. This assumption of local quasi-stationarity will be verified later. Thus, a sliding window of size n is used, which returns the observations y on the local dynamics at step k:

$$y(k) = [y(k) \;\; y(k-1) \;\; y(k-2) \ldots y(k-(n-1))] \tag{11}$$

It follows that the delay vector is allowed to be nonstationary.

The embedding theorem holds given that v is constructed using appropriate values for τ and d. The SOI algorithm uses conventional techniques for the determination of those parameters. The time delay τ is computed at each time step using the mutual information (MI) method based on Shannon's information theory [24], and the embedding dimension is selected using the false nearest neighbor (FNN) method using the algorithm presented in Kennel *et al.* [25].

3.1 Mutual information test

The MI test measures the average information gained from a new measurement, or how well can the outputs $\hat{\boldsymbol{y}}$ be measured given the measurements $\boldsymbol{y}$. Fraser and Swinney [24] presented the theory for mutual information based on Shannon's information theory, which in terms of probabilities has the form:

$$\mathrm{MI}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{i=1}^{n} \mathrm{p}\hat{y}_i \log_2 \mathrm{p}\hat{y}_i - \sum_{j=1}^{n} \mathrm{p}\hat{y}_j \log_2 \mathrm{p}y_j + \sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{p}\hat{y}_i y_j \log_2 \mathrm{p}\hat{y}_i y_j \tag{12}$$

where $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ are two sets of n observations. The first local minima of the MI test gives the optimal time delay, whereas the subsequent minima correspond to a system that has exceedingly unfolded. The computation of eqn (12) is conducted in the SOI algorithm by classifying the last n observations in a pre-defined number of bins $\mathrm{MI}_{\mathrm{bin}}$.

3.2 False nearest neighbor test

The FNN test [25] consists of computing the nearest neighbors r from a point $y\,(m)$ in a given dimension d. Thereafter, the dimension is increased, and the distances $R_d(m, r)$ are computed. If the change is above a certain threshold R_{tol}, then a false neighbor is discovered. The dimension is increased until the percentage of false neighbors converges to zero. Mathematically, a false neighbor is discovered if:

$$\left| \frac{R_{d+1}^2(m,r) - R_d^2(m,r)}{R_d^2(m,r)} \right| > R_{\mathrm{tol}} \tag{13}$$

Kennel *et al*. [25] also added a second condition to ensure that nearest neighbors are also close to each other. This condition is written as:

$$\frac{R_{d+1}(m)}{R_A} > R_{\mathrm{A,tol}} \tag{14}$$

with:

$$R_A^2 = \frac{1}{n}\sum_{m=1}^{n} (z(m) - \bar{z})^2$$

where z is the space location of the new point added with the new dimension, $\bar{z}$ is the arithmetic average of z, and $R_{\mathrm{A,tol}}$ is a threshold. An embedding dimension d is found when the number of false neighbors fall below a threshold R_{num}. Note that, in a dynamic system with

forcing, some neighbors that are tested as false neighbors can actually be true neighbors where those crossings occur [26].

3.3 SOI algorithm

The proposed SOI algorithm sequentially:

1. applies eqn (12) on the last n observations in the search space $[\tau(k-1)-1, \tau(k-1)+1]$ to find $\tau(k)$.
2. applies eqns (13) and (14) using $\tau(k)$ on the last n observations in the search space $[d(k-1)-1, d(k-1)+1]$ to find $d(k)$.
3. adapts the input vector $\zeta(k)$ smoothly using unity changes.

Figure 2 summarizes the SOI algorithm integrated to the WNN, which is the proposed SOI–WNN. In the figure, the SOI algorithm selects values of τ and d using the MI and FNN methods on the last n state and input observations found in the sliding window. A delay vector v is constructed and becomes the objective input space. The actual input space ζ of the WNN is adapted smoothly based on v, and a new forcing u is computed.

The next subsection simulates the SOI–WNN with a simple dynamic system to verify the assumption of local stationarity, its sensitivity to the size of the sliding window n, along with its capability to sequentially identify a set of fixed inputs.

4 VERIFICATION

The proposed SOI–WNN is simulated for tracking the sinusoidal reference signal $y^*(t) = 0.02 \sin t$ from the following nonlinear equation:

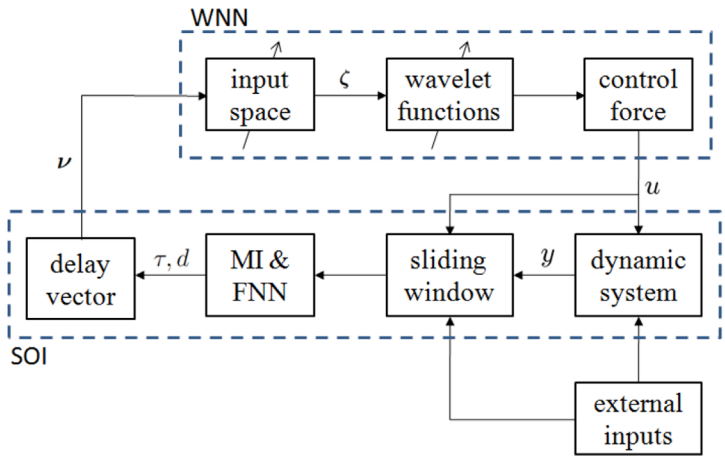


Figure 2: Block diagram of the proposed SOI–WNN.

$$y(x,\dot{x},u)=\left[1-\left(\frac{(x-0.1)^2}{0.1^2}+\frac{(\dot{x}-0.05)^2}{0.05^2}\right)\right]\times e^{\left(-\frac{(x-0.1)^2}{0.1^2}+\frac{(\dot{x}-0.05)^2}{0.05^2}\right)}+u \tag{15}$$

where the excitation input is $x = 0.2 \sin 5t$, and u is the control input. We have selected this arbitrary example, a particle traveling on a wavelet, because of the stationarity of the excitation and reference signal, which isolates the source of nonstationarity in the adaptive representation. In addition, the stationarity of both signals allows us to pre-process their time series using Takens embedding theorem in order to determine the fixed inputs that would appropriately represent their dynamics. Lastly, by using a periodic excitation, it is possible to let the neurocontroller converge to a given control rule. The example in the next section includes a nonperiodic nonstationary excitation.

The WNN non-adaptive parameters are kept constant throughout the simulations, and no training is conducted a priori. Table 1 lists the values selected for those parameters. The selected parameters for the MI and FNN tests are arbitrary. Note that R_{tol} is large to account for crossings in the phase-space due to the forcing u. The minimum nodal distance and error are set equal and at 5% of $\max_t y^*$. The network resolution and pruning parameters are arbitrary and based on previous simulations in Laflamme *et al.* [23]. The adaptation weights are set to unity based on the sampling rate of 100 Hz, a sampling rate typical in structural control, except for the wavelet weights, which are augmented to 10 for enhanced adaptation speed.

The objective is to have the neuro-controller learn the control function as quickly as possible. A delay is induced in the actuator using the following dynamics: $\dot{u}_{\text{act}} = -\eta(u_{\text{act}}-u_{\text{n}})$, where η is a voltage delay taken as $\eta = 20\ \text{s}^{-1}$ to be consistent with the actuator dynamics taken in Laflamme *et al.* [23]. A sliding window size of $n = 100$ time steps is selected. The choice of

Table 1: List of non-adaptive parameters.

NN object	Parameter	Value assigned
Inputs	MI_{bin} (MI test)	20
	R_{tol} (FNN test)	15
	$R_{\text{A,tol}}$ (FNN test)	2
	R_{num}	10%
	Window size n	100
Hidden layer	$d_{\min}$	5% $\max_t$ y*
	s_{all}	5% $\max_t$ y*
	λ	10
	$\gamma_{\min}$	1%
	γ_{num}	50
Adaptation	Weights Γ_{μ}	1
	Weights Γ_{σ}	1
	Weights Γ_{γ}	10

n is discussed later in this section. The SOI algorithm is compared against three cases of fixed inputs:

- An input vector built using $\tau = 31$ and $d = 2$, which are parameters obtained from pre-processing the time series of the excitation signal, without forcing ($u = 0$).
- An input vector built using $\tau = 8$ and d = 2, which are parameters obtained from pre-processing the time series of the reference signal.
- An input vector built using $\tau = 31$ and d = 8, which are the optimal fixed input parameters obtained within the search space $\tau = [1,40]$ and $d = [1,10]$ while simulating the system with forcing.

Note that values obtained for τ are coincidentally the same for the first and third cases, and that a dimension of 2 was expected for the first and second cases due to the low complexity of both signals. The large embedding dimension for the third scenario can be explained by a more complex phase-space that counts several crossings once the eqn (15) includes the forcing u.

Figure 3 shows the time series response of the SOI algorithm versus the optimal fixed parameters. The SOI algorithm (blue straight line) results in a quicker convergence and better tracking results than the optimal fixed-input WNN (black dot-dash line). This is due to the dynamics of the control rule changing with time, for which adapting the input space results in a more efficient representation, as hypothesized. Table 2 shows the root means square (RMS) error for the four input strategies over a tracking time of 20 s, along with the average network size. Results from the overall time series show that the SOI gives good performance relative to the fixed input cases, and preserved a lean network size, with a substantial difference compared with the optimal fixed-input strategy $\tau = 31$, $d = 8$. The RMS error taken after 5 s indicates that its convergence is significantly better.

In addition, the performance of the SOI–WNN has been studied under noise. Various levels of Gaussian noise have been induced in the observations. Figure 4 shows the RMS error for

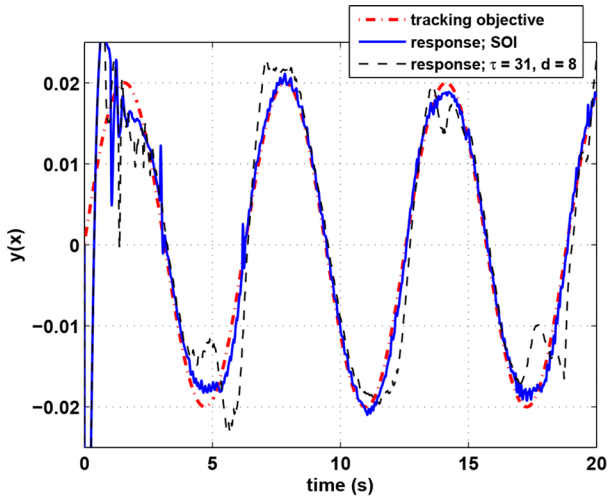


Figure 3: Time series responses. The SOI–WNN (blue straight line) converges more rapidly than the optimal fixed-input WNN (black dot-dash line).

Table 2: RMS error of the controller for various input strategies ($\times 10^{-5}$)

	Self-organizing inputs	$\tau = 31$ $d = 2$	$\tau = 8$ $d = 2$	$\tau = 31$ $d = 8$
Over 20 s	18.9	21.3	23.4	21.1
After 5 s	3.07	13.2	16.4	12.1
Average network size	23.8	25.3	18.6	55.9

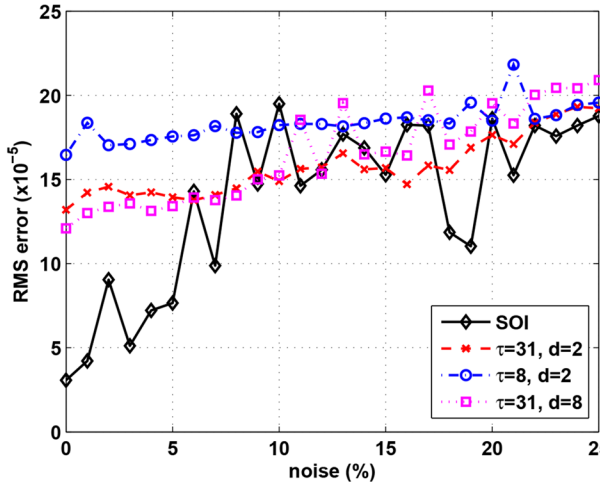


Figure 4: RMS error with respect to noise, after 5 s ($\times 10^{-5}$).

noise ranging from 0% to 25%. The SOI–WNN is capable of significantly outperforming any optimal fixed-input strategies for noise under 5%. However, above that level, the relative performance of the algorithm quickly reduces, with a tendency to perform similarly to the fixed-input strategies.

The SOI algorithm is built under the assumption of quasi-stationarity of local maps within the sliding window. We first verify the performance of the algorithm as a function of the sliding window size n. Figure 5 shows the RMS error after 5 s for various values of n, along with the average computation speed per time step. The performance of the algorithm remains approximately constant for values greater than 45 time steps, with a slight degradation for larger window sizes. Computation speed is affected negatively for low window sizes, because the controller fails at effectively converging. Once the window size is greater than 45, the computation time augments linearly with increasing n. Note that the computation speed remains under the sampling rate of 100 Hz for $45 \leq n \leq 125$. Simulations were conducted in MATLAB with an Intel i7-2600 3.4 GHz CPU.

We now verify the main assumption of quasi-stationarity of local maps. A time se ries stationarity index is constructed by determining the change in the control rule within a map. If the change is minimal, then we can write eqn (1) in a stationary way with $u(k+1) \approx f_u(y(k),\ u(k))$. The observations at step k are taken, and the control force using the control rule at step $k–n$ computed. The stationarity index is built comparing $u(k–n)$ and $u(k)$, and counts the number of local maps that remained under a given percentage change threshold.

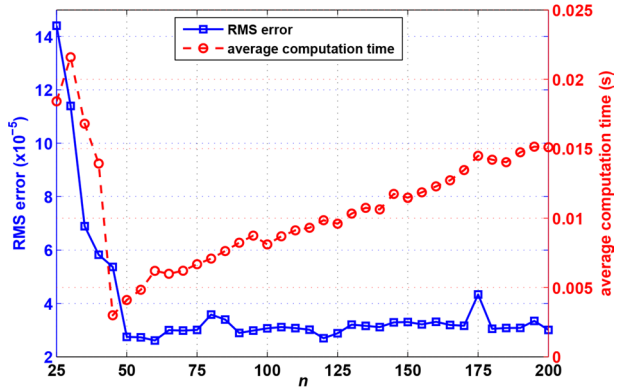


Figure 5: RMS error of the SOI algorithm after 5 s of simulation, along with the average computation speed per time step for various sliding window sizes.

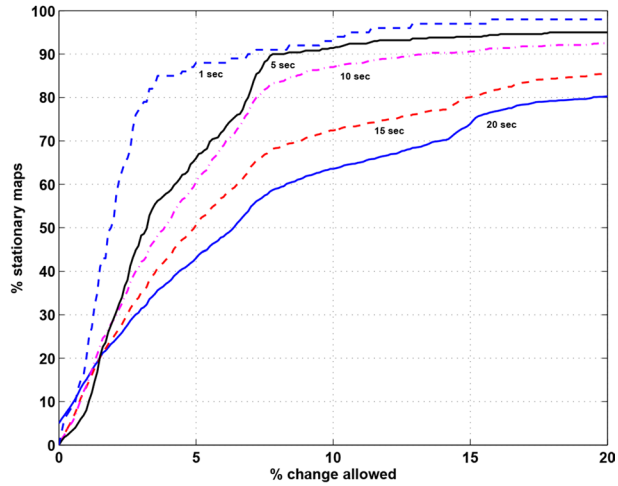


Figure 6: Stationarity index of local maps for the last 1, 5, 10, 15, and 20 s of the simulation.

Figure 6 graphs the stationarity index over various time ranges. The figure shows the number of stationary maps as a function of the percentage of change allowed between $u(k–n)$ and $u(k)$. Results show that 43% of the maps have a change less than 5% over the entire simulation (last 20 s), which increases to 88% for the last 1 s. If a change of 10% is allowed, 64% of maps show to be quasi-stationary over the entire simulation, and 91% over the last 5 s. Results show that the level of quasi-stationarity increases significantly with the convergence of the black-box model. It is estimated that levels of stationary maps above 85% under 10% allowable change satisfy quasi-stationarity, which is met for the last 10 s of the simulation.

Lastly, the SOI algorithm is switched off once the error metric stays below a threshold for a pre-defined number of steps, in order to identify fixed (static) inputs for the representation once the system has converged. For the task, the capacity of the network to prune nodes has been relaxed, as we expect needing a denser network to construct an accurate representation

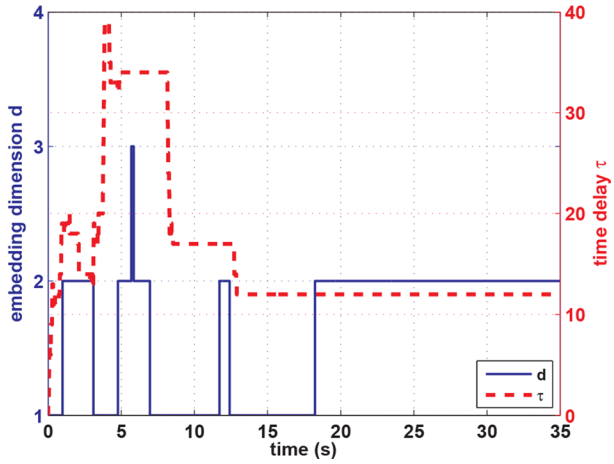


Figure 7: Identification of fixed τ and d for a global representation

of the global dynamics. Figure 7 shows the evolution of the input parameters over time, depicting the self-organizing nature of the input space. The inputs become static after 20 s, identifying the parameters $\tau = 12$ and $d = 2$. This compares well with the pre-processed values of the controlled time series aforementioned to be $\tau = 8$ and $d = 2$, as the phase-space of the sinusoidal target only marginally unfolds between both time delays. The value for d is significantly lower than for the optimal fixed inputs strategy ($d = 8$), because the SOI algorithm computes the optimal d based on the last n observations only.

5 LARGE-SCALE SIMULATION

The WNN controller presented in this paper has been simulated with fixed inputs in Laflamme *et al*. [23]. The simulations were conducted on a 39-storey office tower located in downtown Boston, MA, USA. Here, the same structure is taken, and the SOI–WNN performance as an improvement to the WNN is studied. The simulated system is fully described in Laflamme *et al*. [23], and summarized here for completeness.

The simulated building currently comprises 60 passive damping devices installed on 15 floors, which are replaced in the simulation by semi-active damping devices of similar capacity. The structure is described by McNamara and Taylor in their work [27]. The semi-active devices are large-scale variable friction dampers capable of resisting forces in the order of 1350 kN and are described fully in Laflamme *et al*. [28]. For the scope of this paper, it suffices to consider the devices as nonlinear controllable dampers. For the external excitation, data from a wind tunnel testing are taken and scaled to match the performance described by McNamara and Taylor in their work [27]. The excitation is nonperiodic, nonstationary, and considered as unmeasurable. The sliding controller feature is used, with C taken as 8.33% u_b (50/300 kips), C_t as 50% u_b, and u_b equal to twice the device maximum force of 1350 kN since the force can take equal values of opposite signs.

For the simulation, each of the devices is controlled using local measurements. For each of the 15 decentralized controller, the only available inputs are the acceleration, interstorey displacement, and interstorey velocity of the floor, in addition to the damping force of the local device. The SOI uses the time series of the acceleration to determine the delay vector v

and to subsequently adapt the input space ζ using data from the acceleration and force states. The interstorey displacements and velocities are used for the sliding controller to compute the sliding surface s.

Three main control objectives are utilized for evaluating the performance of the controller: (1) the maximum acceleration of the 37th floor (J1), which corresponds to the highest occupied floor and corresponds with the main control objective in McNamara and Taylor's description [27]; (2) the maximum floor acceleration (J2); and (3) the maximum interstorey displacement (J3). The performance of the SOI–WNN is benchmarked against a linear quadratic regulator (LQR) controller using full-state feedback and full parametric knowledge. In addition, the WNN is simulated with three fixed input strategies optimized over the search space $\tau = [1, 40]$ and $d = [1, 10]$:

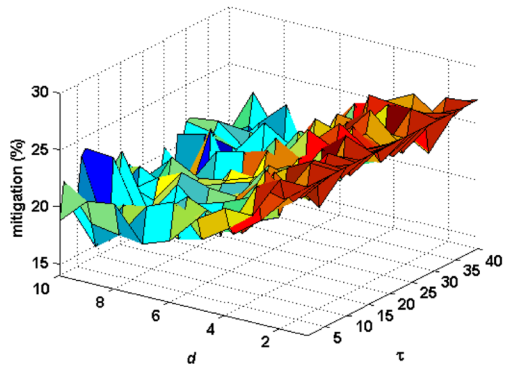
- $\tau = 16$, d = 2, the optimized performance for J1.
- $\tau = 4$, d = 7, the optimized performance for J2.
- $\tau = 40$, d = 2, the optimized performance for J3.

Table 3 shows the mitigation performance provided by all control strategies relative to the existing passive control strategy. The SOI–WNN performs substantially better than all fixed-input strategies. It also achieves better than the LQR controller designed using full parametric knowledge. An explanation for such high performance of the algorithm is the capacity of the input space to adapt to the nonstationary excitation and system nonlinearities. Additionally, despite that the system is nonlinear due to the semi-active control devices, the LQR controller was designed assuming linearity in the control force, from which the control device would try to achieve the required force. Such design is common in structural control, as the dynamic of civil structures is inherently stable, and semi-active devices cannot destabilize the system.

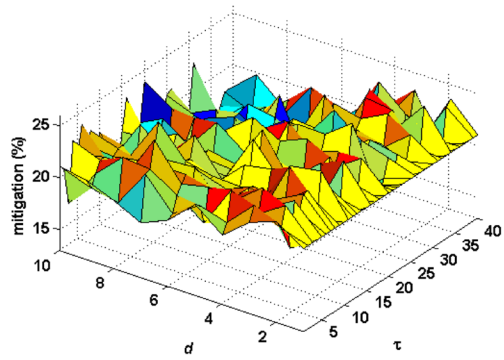
We also look at the sensitivity of the fixed-input strategies. Figure 8 shows three-dimensional plots of the mitigation performance for the three performance indices in function of τ and d. A first observation is that the mitigation performance for both J1 and J3 is close to optimal for a dimension $d = 1$, and does not seem to be influenced by a change in τ, meaning that the controller performance can barely be improved by selecting inputs other than a single feedback. Note that the WNN tries to achieve sequential control, without pre-training. Thus, a fixed-input strategy shows to be inefficient in this case at achieving the task of quick learning, because the system is nonlinear and nonstationary, thus requiring several training examples. Index J2 shows to have a degradation of performance for high dimension and time delay, and also shows a high variation in mitigation performance around the optimal solution of $\tau = 4$, $d = 7$, demonstrating the importance of input selection.

Table 3: Mitigation (%) with respect to the existing passive control strategy

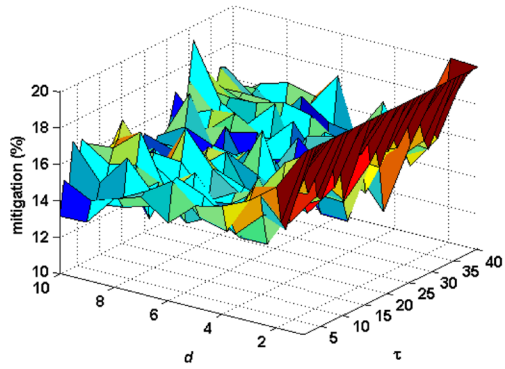
	Linear quadratic regulator	Self-organizing inputs	$\tau = 16$ $d = 2$	$\tau = 4$ $d = 7$	$\tau = 40$ $d = 2$
J1	29.0	32.0	28.0	20.1	26.3
J2	23.7	30.2	24.4	25.5	23.6
J3	24.1	24.2	15.2	15.0	20.1



(a)



(b)



(c)

Figure 8: Sensitivity of the fixed input parameters to mitigation performance for: (a) J1; (b) J2; and (c) J3

6 CONCLUSION

In this paper, a novel type of neural network has been presented. The neural net consists of a single layer feedforward WNN, and the main contribution is the online organization of the input space. A verification using sequential tracking of a nonlinear function showed that the proposed neural network was capable of outperforming any fixed-input strategies. In addition, simulations conducted on an existing structure show the significant mitigation performance of the proposed SOI–WNN to mitigate vibration in a nonstationary and nonlinear system. It performed better than any of the fixed-input strategies, and also better than an LQR controller designed using full knowledge of the system and operated using full-state feedback. The SOI–WNN is a powerful neural network capable of sequential learning, which is of interest for unknown systems that cannot be trained a priori. It combines the benefits of (1) automating the input selection process for time series that are not known a priori; (2) adapting the representation to nonstationarities; and (3) using limited observations.

REFERENCES

[1] Poggio, T. & Girosi, F., Networks for approximation and learning. *Proceedings of the IEEE*, **78(9)**, pp. 1481–1497, 1990. doi: http://dx.doi.org/10.1109/5.58326

[2] Zhang, Q., Benveniste, A. & Hogskola, L., Wavelet networks. *IEEE Transactions on Neural Networks*, **3(6)**, pp. 889–898, 1992. doi: http://dx.doi.org/10.1109/72.165591

[3] Bowden, G., Dandy, G. & Maier, H., Input determination for neural network models in water resources applications. Part 1–background and methodology. *Journal of Hydrology*, **301(1–4)**, pp. 75–92, 2005. doi: http://dx.doi.org/10.1016/j.jhydrol.2004.06.021

[4] da Silva, A., Alexandre, P., Ferreira, V. & Velasquez, R., Input space to neural network based load forecasters. *International Journal of Forecasting*, **24(4)**, pp. 616–629, 2008. doi: http://dx.doi.org/10.1016/j.ijforecast.2008.07.006

[5] Sindelar, R. & Babuska, R., Input selection for nonlinear regression models. *Fuzzy Systems, IEEE Transactions on*, **12(5)**, pp. 688–696, 2004. doi: http://dx.doi.org/10.1109/tfuzz.2004.834810

[6] Hong, X., Mitchell, R., Chen, S., Harris, C., Li, K. & Irwin, G., Model selection approaches for non-linear system identification: a review. *International Journal of Systems Science*, **39(10)**, pp. 925–946, 2008. doi: http://dx.doi.org/10.1080/00207720802083018

[7] Blum, A. & Langley, P., Selection of relevant features and examples in machine learning. *Artificial Intelligence*, **97(1–2)**, pp. 245–271, 1997. doi: http://dx.doi.org/10.1016/s0004-3702(97)00063-5

[8] Kohavi, R. & John, G., Wrappers for feature subset selection. *Artificial Intelligence*, **97(1–2)**, pp. 273–324, 1997. doi: http://dx.doi.org/10.1016/s0004-3702(97)00043-x

[9] Guyon, I. & Elisseeff, A., An introduction to variable and feature selection. *The Journal of Machine Learning Research*, **3**, pp. 1157–1182, 2003.

[10] Yu, D., Gomm, J. & Williams, D., Neural model input selection for a MIMO chemical process. *Engineering Applications of Artificial Intelligence*, **13(1)**, pp. 15–23, 2000. doi: http://dx.doi.org/10.1016/s0952-1976(99)00046-9

[11] Li, K. & Peng, J., Neural input selection: a fast model-based approach. *Neurocomputing*, **70(4–6)**, pp. 762–769, 2007. doi: http://dx.doi.org/10.1016/j.neucom.2006.10.011

[12] Tikka, J., Simultaneous input variable and basis function selection for RBF networks. *Neurocomputing*, **72(10–12)**, pp. 2649–2658, 2009. doi: http://dx.doi.org/10.1016/j.neucom.2008.10.003

[13] Kourentzes, N. & Crone, S., Frequency independent automatic input variable selection for neural networks for forecasting. *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2010. doi: http://dx.doi.org/10.1109/ijcnn.2010.5596637

[14] Takens, F., Detecting Systems and Turbulence. *Lecture Notes in Mathematics, Warwick 1980*, **898**, pp. 366–381, 1980.

[15] Stark, J., Delay embeddings for forced systems. I. Deterministic forcing. *Journal of Nonlinear Science*, **9(3)**, pp. 255–332, 1999. doi: http://dx.doi.org/10.1007/s003329900072

[16] Caballero, V., On an embedding theorem. *Acta Mathematica Hungarica*, **88(4)**, pp. 269–278, 2000. doi: http://dx.doi.org/10.1023/A:1026753605784

[17] Stark, J., Broomhead, D., Davies, M. & Huke, J., Delay embeddings for forced systems. II. Stochastic forcing. *Journal of Nonlinear Science*, **13(6)**, pp. 519–577, 2003. doi: http://dx.doi.org/10.1007/s00332-003-0534-4

[18] Hirata, Y., Suzuki, H. & Aihara, K., Reconstructing state spaces from multivariate data using variable delays. *Physical Review E*, **74(2)**, p. 26202, 2006. doi: http://dx.doi.org/10.1103/physreve.74.026202

[19] Nerukh, D., Ryabov, V. & Glen, R., Complex temporal patterns in molecular dynamics: a direct measure of the phase-space exploration by the trajectory at macroscopic time scales. *Physical Review E*, **77(3)**, p. 36225, 2008. doi: http://dx.doi.org/10.1103/physreve.77.036225

[20] Monroig, E., Aihara, K. & Fujino, Y., Modeling dynamics from only output data. *Physical Review E*, **79(5)**, p. 56208, 2009. doi: http://dx.doi.org/10.1103/physreve.79.056208

[21] Laflamme, S., Slotine, J.E. & Connor, J., Self-organizing input space for control of structures. *Smart Materials and Structures*, **21(11)**, p. 115015, 2012. doi: http://dx.doi.org/10.1088/0964-1726/21/11/115015

[22] Kohonen, T., The self-organizing map. *Proceedings of the IEEE*, **78(9)**, pp. 1464–1480, 1990. doi: http://dx.doi.org/10.1109/5.58325

[23] Laflamme, S., Slotine, J. & Connor, J., Wavelet network for semi-active control. *J of Engineering Mechanics*, **137(7)**, pp. 462–474, 2011. doi: http://dx.doi.org/10.1061/(asce)em.1943-7889.0000248

[24] Fraser, A. & Swinney, H., Independent coordinates for strange attractors from mutual information. *Physical Review A*, **33(2)**, pp. 1134–1140, 1986. doi: http://dx.doi.org/10.1103/physreva.33.1134

[25] Kennel, M., Brown, R. & Abarbanel, H., Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review A*, **45(6)**, pp. 3403–3411, 1992. doi: http://dx.doi.org/10.1103/physreva.45.3403

[26] Overbey, L., Olson, C. & Todd, M., A parametric investigation of state-space-based prediction error methods with stochastic excitation for structural health monitoring. *Smart Materials and Structures*, **16**, pp. 1621–1638, 2007. doi: http://dx.doi.org/10.1088/0964-1726/16/5/016

[27] McNamara, R. & Taylor, D., Fluid viscous dampers for high-rise buildings. *The Structural Design of Tall and Special Buildings*, **12(2)**, pp. 145–154, 2003. doi: http://dx.doi.org/10.1002/tal.218

[28] Laflamme, S., Taylor, D., Abdellaoui Maane, M. & Connor, J., Modified friction device for control of large-scale systems. *Structural Control and Health Monitoring*, **19(4)**, pp. 548–564, 2012. doi: http://dx.doi.org/10.1002/stc.454