Generalised Performance of Neural Network Controllers for Feedforward Active Noise Control of Nonlinear Systems

Xander Pike[1)], Jordan Cheer[2)]
[1,2)] Institute of Sound and Vibration Research, University of Southampton, Southampton, SO17 1BJ, UK

Abstract

Advances in digital technologies have allowed for the development of complex active noise and vibration control solutions that have been utilised in a wide range of applications. Such control systems are commonly designed using linear filters, which cannot fully capture the dynamics of nonlinear systems. To overcome such issues, it has been shown that replacing linear controllers with Neural Networks (NNs) can improve control performance in the presence of nonlinearities. Many real systems are subject to nonstationary disturbances where the magnitude of the system excitation time dependent. However, within the literature, the performance of single NN controllers across different excitation levels has not been thoroughly explored. In this paper, a method of training Multilayer Perceptrons (MLPs) for single-input-single-output (SISO) feedforward acoustic noise control is presented. In a simple time-discrete simulation, the performance of the trained NNs is investigated for different excitation levels. The effects of the properties of the training data and NN controller on generalised performance are explored. It is demonstrated that the generalised control performance of the MLP controllers falls as the range of magnitudes included in the training data is increased, and that this performance can be recovered by increasing the number of hidden nodes within the controller.

1 Introduction

Unwanted noise and vibration can be problematic in both engineering systems and in public and private spaces. Passive control solutions are capable of effectively reducing high frequency components of noise and vibration but are typically large and/or heavy for low frequency control, possibly exceeding the design constraints of a given application. Active control solutions, by contrast, are capable of effective control at low frequencies, and are typically lightweight and compact. Historically, feedforward active noise and vibration control systems have been implemented using linear control filters and linear plant models, commonly using the FxLMS algorithm. However, it is well understood that nonlinearities present in either the plant or primary path of the control system can have a significant impact on control performance [1], [2], [3], [4]. Many approaches have been proposed to overcome this limitation, including polynomial, cross-term or trigonometric expansion of the reference signal [5], [6], genetic algorithms [7] and fuzzy logic-based methods [8]. A further common approach, which has been applied to active control over the past few decades, is the application of machine learning methods. NNs in particular are known to possess the property of being 'universal approximators' [9] and are therefore an attractive black-box method for the modelling and control of unknown or uncertain nonlinear systems. The similarities in structure between NNs and linear filters provides good motivation for their use in both system modelling and feedforward controller design. Many different uses of NNs have been studied, including system modelling [4], [10], [11], [12], [13], feedforward controller design [4], [10], [14], inverse modelling [15], signal prediction and feedback control [16], [17], [18], [19], [20], linear filter selection [21], adaptive parameter estimation for linear controllers [20], [22], frequency-domain control [23], multichannel controller design [24], and signal classification [25]. In previous work utilising NNs as feedforward controllers, however, the ability for the controller or plant model NNs to generalise across a range of excitation levels of the studied system has not been thoroughly explored. This is clearly a desirable quality in any real implementations of such a control system where the properties of the excitation, and therefore the effect of the system nonlinearity, may change over time. In this paper, a simulation of a simple noise control system implementing a time-domain MLP controller is studied. Section 2 defines the simulated system, system parameters and simulation method. Section 3 explains the controller training methodology. Section 4 presents simulated results in the time and frequency domains. Finally, Section 5 discuss conclusions and presents possible future research directions.

2 Problem definition

2.1 Simulated System

The considered system consists of two acoustic sources. The primary source, which generates the acoustic disturbance, is modelled as a damped Duffing oscillator, assumed to radiate as a monopole acoustic source. The secondary acoustic source, which generates the cancelling acoustic signal, is modelled as a simple harmonic oscillator, also assumed to radiate as a monopole source. Figure 1 presents a diagram of the simulated system.

The displacement of the Duffing oscillator, $y_a(t)$, is caused by the motion of the floor to which it is coupled. The displacement, $x(t)$, of this floor is also taken to be the reference signal passed to the feedforward controller. The displacement, $y_b(t)$, of the mass, m_b, is caused by the control force, $F_c(t)$, produced by the controller.

The equations of motion for the total system are

$$m_a\ddot{y}_a(t) + k_a p(t) + k_a^{NL} p^3(t) + c_a\dot{p}(t) = 0 \qquad (1)$$

$$m_b\ddot{y}_b(t) + k_b y_b(t) + c_b\dot{y}_b(t) + F_c(t) = 0 \qquad (2)$$

where $p(t) = y_a(t) - x(t)$ and the remaining variables are defined in Figure 1 and their values are provided in Table 1. These parameter values were selected such that the two oscillators have unity mass, but distinct resonance frequencies of 60 Hz and 80 Hz. The damping coefficients c_a and c_b were selected such that each oscillator is subject to 20% of critical damping, and the oscillators are therefore not significantly underdamped or overdamped. Assuming for simplicity that the error sensor is equidistant from the two point monopole sources and that the constant amplitude scaling factors are equal both cases such that they can be neglected, the error signal is defined as

$$e(t) = y_a(t - \delta_a) + y_b(t - \delta_b) \qquad (3)$$

where δ_a and δ_b are the acoustic delays, in time, between the primary and secondary sources and the error microphone, respectively. In all cases, the signal $x[n]$ is Gaussian white noise band-limited to the frequency range $[0, 250]$ Hz. The motion of the sources is simulated in the time domain using a 4th order Runge-Kutta method at a sample rate of $f_s = 2$ kHz.

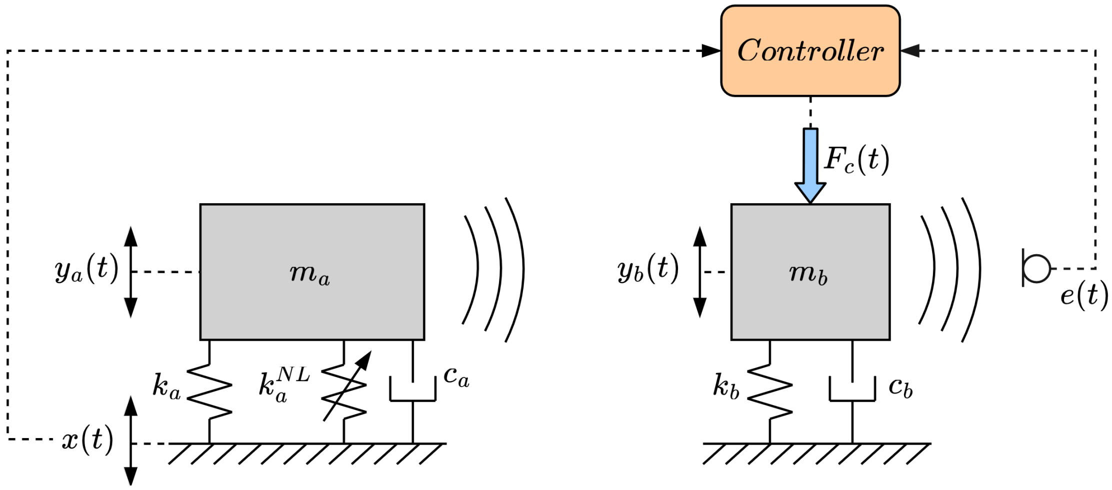

Figure 1: Diagram of the simulated system, consisting of a nonlinear primary acoustic source, and a linear control source. System parameter values are given in Table 1.

.

Table 1: Simulated system parameter values.

Parameter	**Symbol**	**Value**
Primary oscillator mass	m_a	1 kg
Secondary oscillator mass	m_b	1 kg
Primary oscillator linear stiffness	k_a	1.42×10^5 Nm^{-1}
Primary oscillator cubic stiffness	k_a^{NL}	1.42×10^{14} Nm^{-3}
Secondary oscillator stiffness	k_b	2.53×10^5 Nm^{-1}
Primary oscillator damping	c_a	151 Nsm^{-1}
Secondary oscillator damping	c_b	201 Nsm^{-1}

3 Controller design, training and testing

3.1 Controller architecture

A diagram of the NN architecture used to train the controller is presented in Figure 2. Similarly to the case of a linear controller, the NN controller and plant model each take as input a tapped delay line of the sampled reference signal, $x[n]$, and sampled control signal, $u[n]$, respectively. The plant model output, $\hat{y}[n]$, is linearly summed with the disturbance signal, $d[n]$, to generate the error signal estimate, $\hat{e}[n]$, which is used to update the weights and biases of the controller NN via backpropagation through the full network consisting of both the controller and plant model. Given a tapped delay line of length L of the reference signal, $x[n]$, given by

$$\mathbf{x}[n] = \begin{bmatrix} x[n] \\ x[n-1] \\ \vdots \\ x[n-L+1] \end{bmatrix} \tag{4}$$

the control signal, $u[n]$, can be generated by passing $\mathbf{x}[n]$ through the controller NN. If the NN architecture is that of an MLP with a single hidden layer, then the output of the NN is given by

$$u[n] = \sum_i w_i^o h_i + b^o \tag{5}$$

where w_i^o are the output weights of the NN, b^o is the NN output bias, and h_i are the NN hidden layer node values, given by

$$h_i = \sigma([\mathbf{W}\mathbf{x}[n]]_i + b_i^h) \tag{6}$$

where $\mathbf{W}$ is a matrix of weights between the input layer and hidden layer, $[\mathbf{W}\mathbf{x}[n]]_i$ is the i^{th} element of the vector $\mathbf{W}\mathbf{x}[n]$, $\sigma(\cdot)$ is the nonlinear activation function applied to the controller hidden layer, and b_i^h is the bias of the i^{th} hidden layer node. In total,

$$u[n] = \sum_i w_i^o \sigma([\mathbf{W}\mathbf{x}[n]]_i + b_i^h) + b^o \tag{7}$$

However, a full tapped delay line $\mathbf{u}[n]$ is required to infer the output of the plant model, $\hat{y}[n]$, and therefore the error estimate, $\hat{e}[n]$. It is therefore necessary to generate control signal values $u[n-1]$, $u[n-2]$, $\ldots$, $u[n-I+1]$ for a tapped delay line of length I. A previously presented solution to this problem [26] is to train the model using the training data sequentially, storing the values of the control signal in memory, and calling upon them when evaluating the output of the plant model at each timestep during the updating of the controller weights and biases. However, a result of this approach is that the error estimate, $\hat{e}[n]$, will not accurately reflect the control performance of the current iteration of the controller. The control signal tapped delay line is calculated from the outputs of the current and previous $L-1$ iterations of the controller, and therefore so is the error estimate. This could plausibly lead to stability and performance issues in the training of the controller NN. Perhaps more importantly, this sequential approach will face the issue of catastrophic interference [27], meaning it is unsuitable for training networks without compromising generalised control performance. As illustrated in Figure 2, an alternative approach is proposed here where all required previous controller outputs, $u[n-k]$, are generated using the current iteration of the controller. In general,

$$u[n-k] = \sum_i w_i^o \sigma([\mathbf{W}\mathbf{x}[n-k]]_i + b_i^h) + b^o \tag{8}$$

where all weights and biases in equation 8 are those of the current iteration of the controller during training. Irrespective of whether the values of $u[n-k]$ are called from memory or generated from the current iteration of the controller, standard backpropagation techniques can be used to update the weights and biases of the controller to minimise a given cost function of $\hat{e}[n]$. The approach presented in Figure 2 is clearly more computationally intensive than simply storing $u[n]$ in memory. It should be noted, however, that computing $u[n]$ is only required during the controller training. The controller NN is assumed here to be fixed during operation and so, for a NN controller with the same number of hidden nodes, the computational cost to produce $u[n]$ from $x[n]$ in operation is independent of the training method.

3.2 Controller training

The controllers were trained to minimise the Mean Squared Error (MSE) signal, defined as

$$J = \overline{\hat{e}^2[n]} \tag{9}$$

where the average is calculated over the samples in the training batch. The backpropagation used the Adam algorithm [28] with parameters $\alpha = 3 \times 10^{-5}$, $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\epsilon = 10^{-7}$. These parameters were selected through trial and error with a view to reaching a trade-off between controller performance and training speed.

In all cases, the plant model used for controller training was an FIR filter (equivalent to an MLP with no hidden layer) with 140 taps, which was capable of achieving high levels of modelling accuracy due to the linear nature of the simulated plant response.

3.3 Training data

For each instance of network training, two sets of 900 s of simulated data are generated. The first set is used

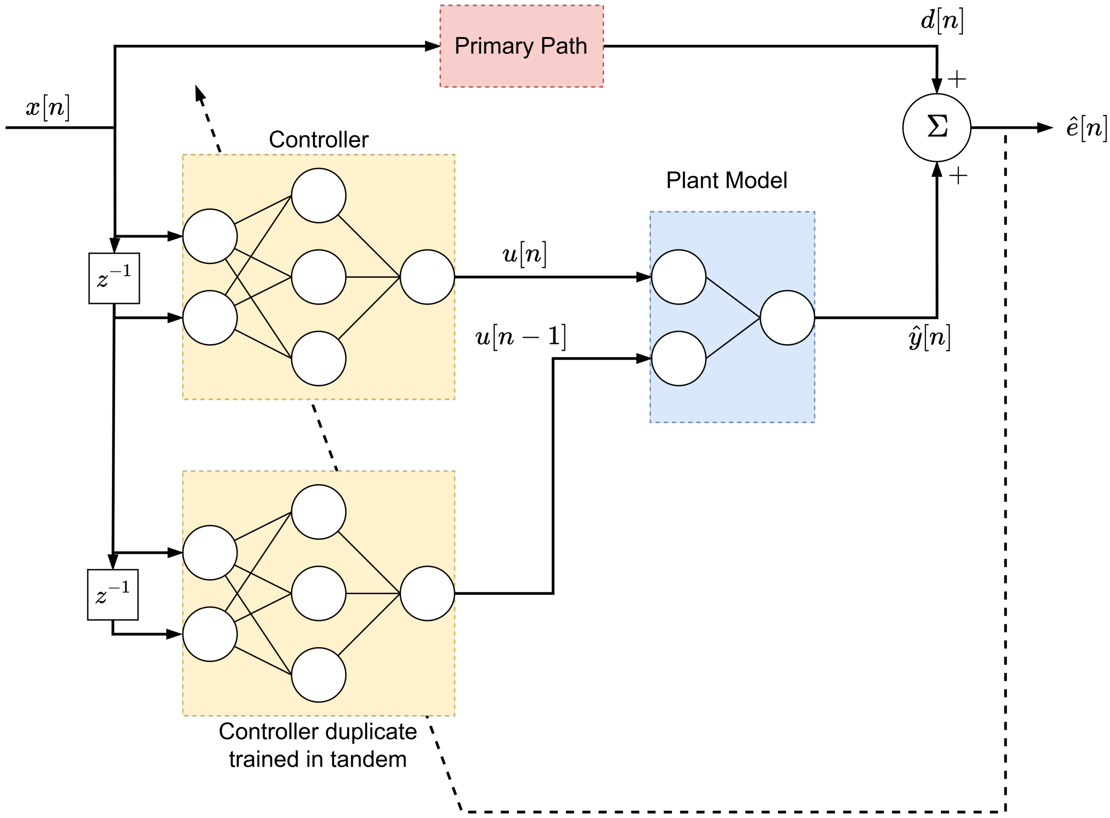

Figure 2: Block diagram of the controller training method, assuming an FIR plant model of order 2.

for training, and the second set is used as a validation set to assess model overfitting. Each dataset consists of the reference signal, $x[n]$, and the resultant displacement of the primary source, $y_a[n]$. In each 900 s simulation, the magnitude of the reference signal, $x[n]$, is increased linearly in time from the lower to the upper bound of the the training range. The training data is split into mini-batches of size 128, selected randomly from the data. In each iteration of the training, 1000 such mini-batches are employed. The full set of training data are therefore not used in each iteration. This is due to a training bias which is implemented to optimise network training for generalised performance, explained below. All networks are trained over 500 iterations. All signals used to generate training and validation data are first normalized to have a standard deviation of 1 to facilitate quick and stable training of the NNs. As expected, using small amounts of training data was observed to result in model overfitting, reducing the performance of the controllers and obscuring the underlying limits on controller performance. The relatively large amount of training data used was therefore chosen to minimize the effect of overfitting during network training, and therefore no regularization was applied to the networks.

During testing, it was initially observed that MLP controllers trained using data with equal weighting across all magnitudes of the reference signal produced MSE attenuation that was approximately equal across much of the training range. As shall be noted in Section 4, the maximum performance of the MLP controllers at each magnitude of the reference signal is not equal. As a result, training an MLP as a controller across a range of reference signal magnitudes results in control performance close to the maximum for the largest training magnitudes, but underperformance relative to the maximum at the lowest training magnitudes. To counteract this, the random selection of the training samples was weighted based on the magnitude of the reference signal. That is, for a set of N training examples with reference signal magnitudes x_{mag} in the range $a < x_{mag} < b$, the probability of training example q being included in a training batch is given (up to a normalizing factor) by

$$P(q) \propto 10^{-\gamma(x_{mag}-a)} \tag{10}$$

where γ is a factor controlling the training bias. The inclusion of this training bias affects the resultant performance of the MLP controllers across the training range. Appropriate selection of γ for a given training range results in generalised control performance that approaches the maximum MLP controller performance across the training range. This effect is illustrated in Section 4.2.

3.4 Testing

During the controller testing phase, the controller network is extracted and used to generate the control signal $u[n]$ in a new simulation. At each timestep, the controller is input with a tapped delay line of the reference signal, $\mathbf{x}[n] = [x[n], ..., x[n-L+1]]$, generating a control signal sample, $u[n]$. The testing is undertaken across a range of magnitudes of $x[n]$. However, this magnitude is kept constant within each testing simulation. Each testing simulation is undertaken over 60 s, and control performance is defined as MSE attenuation, measured in dB relative to the sampled disturbance signal, $d[n]$.

4 Results

4.1 Maximum control performance

To first establish an upper limit on the generalised control performance of the MLP controllers across a range of magnitudes of $x[n]$, a set of 90 networks with 100 hidden nodes were trained at 30 equally spaced magnitudes of $x[n]$ from 10^{-9} to 10^{-5} m with three controllers trained at each level to obtain an average of the control performance. Controller weights and biases were initialized with a random Gaussian distribution with zero mean and a standard deviation of 0.05. The performance of these controllers is presented in Figure 3. Also presented is the performance of FIR filter-based controllers trained using the same Adam algorithm, and the control performance of FIR controllers trained in-simulation using the FxLMS algorithm.

At very low magnitudes of the reference signal, both the FIR and MLP controllers achieve a similar level of performance of between approximately 55 and 60 dB. At the lowest magnitudes, the FIR controller outperforms the MLP controller slightly. However, as the magnitude of $x[n]$ increases and the degree of nonlinearity stimulated in the primary path increases, the MLP controller demonstrates a clear control advantage compared to the linear controller of up to 20 dB. At the highest levels of nonlinearity, the performance of both controller architectures falls. However, the MLP controller still achieves some control advantage of approximately 5 dB. Interestingly, the performance of the linear controller shows a standard deviation of 2.6 dB at the lowest magnitude of $x[n]$ tested. This is unexpected, and is absent in the results of the FxLMS-trained controller. It is reasonable to assume, therefore, that this variance is a result of the Adam algorithm used to train the FIR controllers.

Figure 4 presents Power Spectral Density (PSD) estimates of the disturbance and error signals at a range of values of x_{mag} for the FIR and MLP controllers. As observed in Figure 3, at the lowest value of x_{mag}, the FIR controller slightly outperforms the MLP controller. However, as x_{mag} increases, the MLP controller outperforms the FIR controller. Notably, the MLP controller outperforms the FIR controller across all frequencies present in the reference signal.

4.2 Effect of training bias γ

As explained in Section 3.3, the training of the MLP controllers for control across a range of x_{mag} is influ-

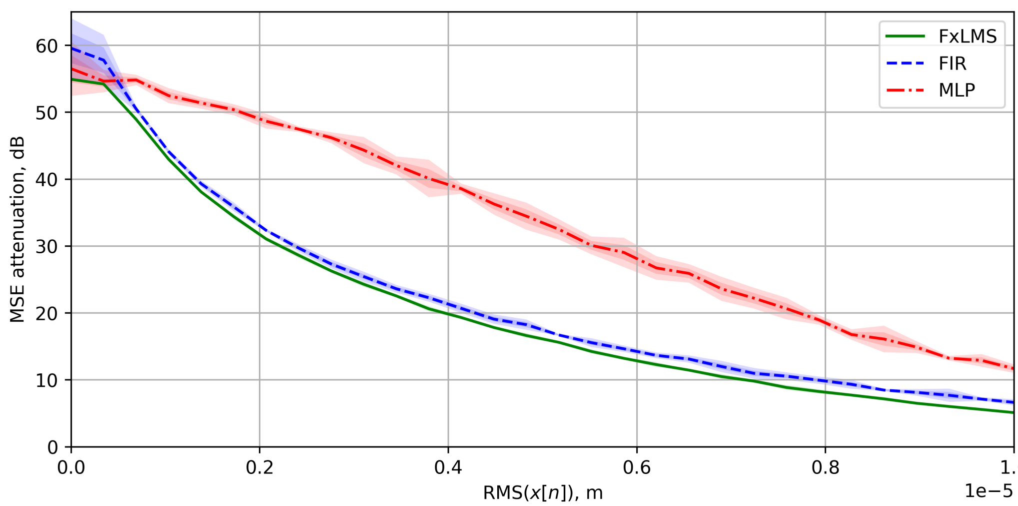

Figure 3: Average control performance of the MLP controller, and an FIR controller trained using the Adam algorithm and FxLMS. Averaging is undertaken over 5 instances of a trained controller. Solid lines represent the mean control performance. Shaded regions present 2 standard deviations of control performance around the mean.

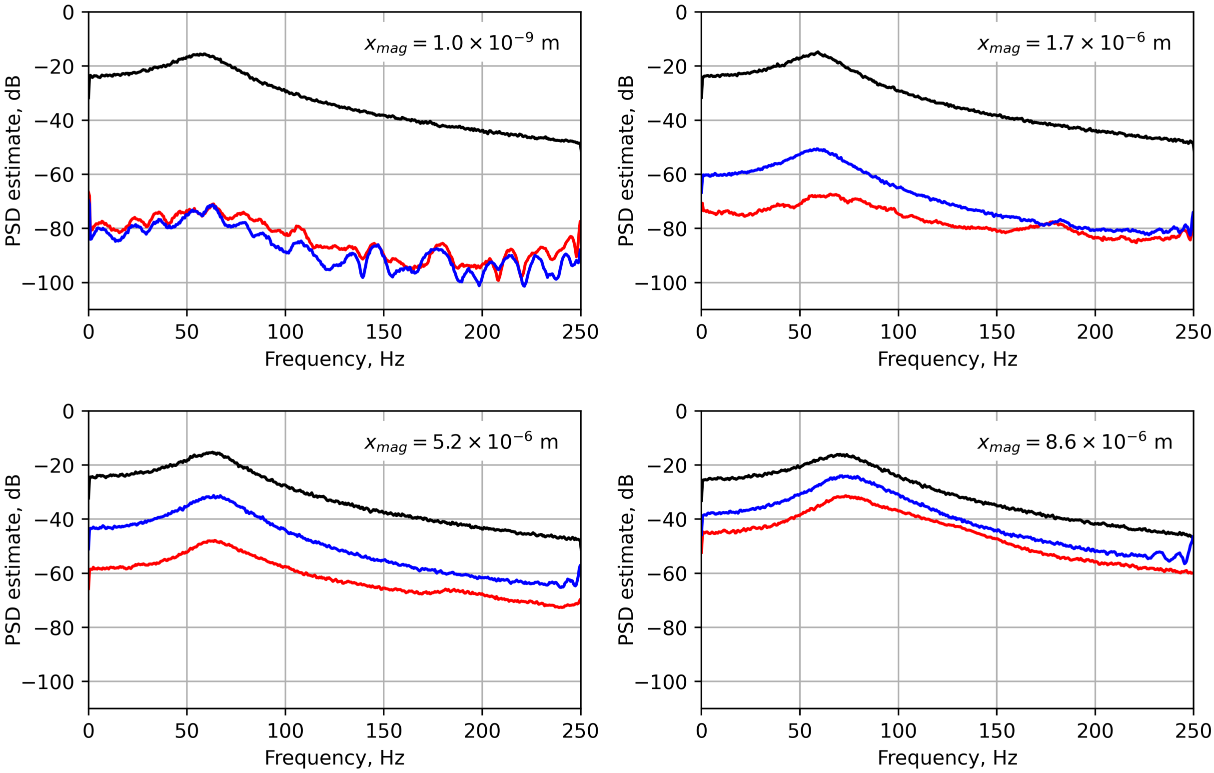

Figure 4: Frequency-domain errors corresponding to the MLP and FIR filter-based controllers over a range of reference signal magnitudes x_{mag}, averaged over 5 instances of controller training. Black: Disturbance; Blue: FIR controller error; Red: MLP controller error. All signals used to calculate PSD estimates are normalised with respect to the RMS of the disturbance signal.

enced by a training bias which is fully defined by a parameter γ. In effect, this parameter controls the slope of the generalised performance curve within the training region for a given controller. As the generalised performance of the controllers is limited to a maximum at each magnitude of $x[n]$, the tuning of γ allows for generalised performance to be maximised across the training region. Figure 5 presents the control performance of 3 MLP controllers trained with different values of the parameter γ. In each case, the controller contains a hidden layer of 100 nodes, and the training range is shown by the red shaded region. As observed in Section 4.1, the control performance of MLP networks is subject to variance. Therefore, to produce smooth generalised performance curves that illustrate the effect of γ more clearly, the training data used for each network was identical, and the reference signal used in each testing simulation was identical across tests, but scaled to the appropriate magnitude. As expected, the effect of increasing γ is to change the slope of the generalised performance curve within the training region. We can observe that, for $\gamma = 1.2 \times 10^6$, the generalised network performance is close to the maximum at the bottom of the training range, but falls away from the maximum near the top. We see the opposite effect for the network trained with $\gamma = 0$. For $\gamma = 6 \times 10^5$, the generalised performance of the controller relative to the maximum is approximately constant within the training range.

4.3 Effect of training range

For a fixed number of hidden nodes in the MLP controller, the effect of increasing the range of x_{mag} over which the controller is trained is investigated. Figure 6 presents the control performance of 4 networks with increasing training widths. The first network, trained at $x_{mag} = 5 \times 10^{-5}$ m, achieves control performance equal to the defined maximum at the trained magnitude, as expected. It also demonstrates some generalised performance capacity, with performance relative to the maximum curve dropping by approximately 3 dB within the range of $x_{mag} \in [4 \times 10^{-6}, 6 \times 10^{-6}]$. Increasing the range of the training to $[4 \times 10^{-6}, 6 \times 10^{-6}]$ slightly improves the control performance at the edges of this range at very little cost to performance at the centre of the range. Further increasing the training range contin-

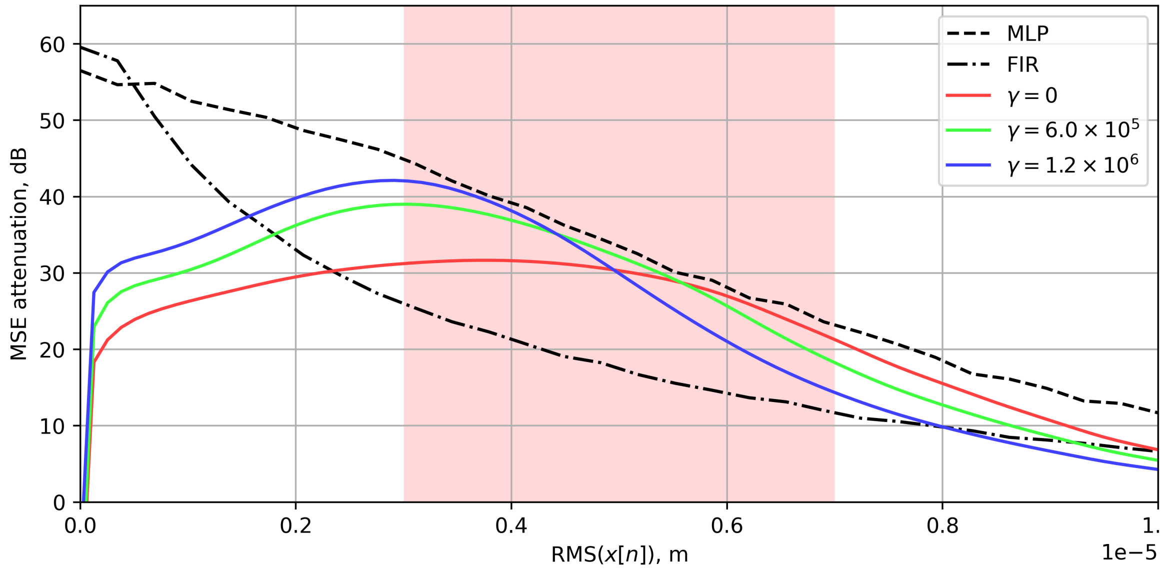

Figure 5: Performance of 3 MLP controllers trained with varying values of the training bias parameter γ. The red highlighted region represents the range of magnitudes of $x[n]$ over which the controllers were trained.

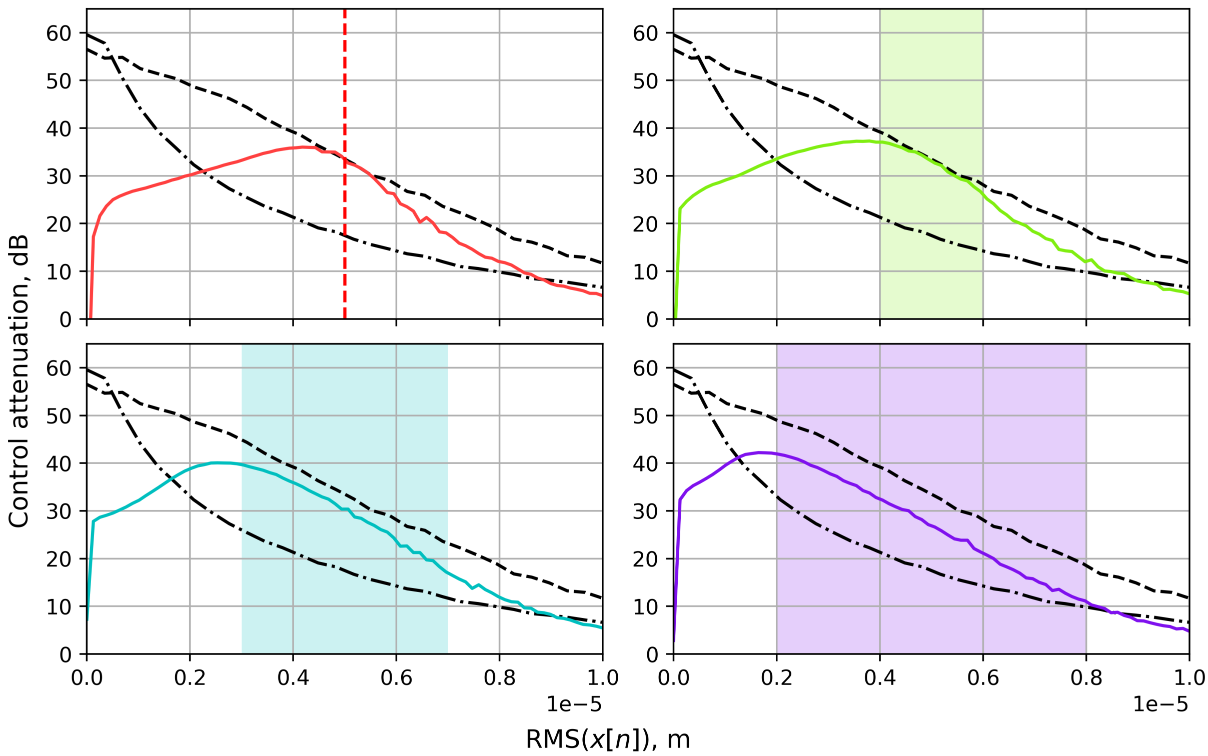

Figure 6: MSE attenuation of 4 MLP controllers trained with reference signal magnitude within the highlighted range. Each plot presents the averaged performance of 3 controller instances trained within the presented range.

ues to reduce the performance of the controller at the centre of the range whilst improving generalisability. This is expected, as increasing the range of the magnitude of the training data increases the range of nonlinear behaviour exhibited by the training data. We should therefore expect that keeping the number of hidden nodes in the networks fixed, but increasing the training range, should reduce the peak performance of the controller relative to the maximum curve as the controller network becomes increasingly under-powered to perform the required generalisation task.

4.4 Effect of network size

The ability of NNs to generate nonlinear mappings from input to output is derived from the nonlinear activation functions applied to the nodes of the hidden layer(s). It is natural, therefore, to assume that increasing the number of nodes in the hidden layer of the MLP controller will increase its generalised control performance across a range of reference signal magnitudes. Figure 7 shows the generalised control performance of 5 MLP controllers with a range of hidden nodes from 12 to 200. At each magnitude of $x[n]$ and number of hidden nodes, 3 controllers were trained and the performance averaged over the 3 controllers. Across the training range, the controller with 12 hidden nodes achieves a performance approximately 7 dB below the defined maximum MLP performance. Increasing the number of hidden nodes to 25 increases the performance of the controller relative to the maximum by approximately 2 dB. However, this approximately doubles the number of parameters to be learned within the network, and approximately doubles the number of operations required to infer the output of the network. Further doubling of the number of hidden nodes in the MLP controllers further increases the generalised control performance of the controllers. However, this increase becomes increasingly small as the performance of the networks approach the maximum. Noting that the controller containing 12 hidden nodes generates a control performance advantage over the linear controllers of up to 10 dB, the computational cost of increasing the number of hidden nodes from 12 to 200 to gain an additional 8 dB in control performance is considerable. In practical applications, the additional computational cost of both training and inferring the outputs of these networks may be prohibitive to their implementation for excessively large networks. However, it should be noted that increasing the number of hidden nodes in the MLP controller does recover the control performance lost by increasing the controller training width.

5 Conclusions

In this paper, a method of training MLP NNs for use as time-domain controllers for feedforward active noise control has been presented. For the simple case presented, the maximum achievable performance of linear and MLP controllers has been estimated, demonstrating that the control performance of both the linear and MLP controllers falls as the degree of nonlinearity in the system primary path increases. The ability of MLP controllers to achieve generalised performance across a range of system stimulation magnitudes has been investigated, and a training bias parameter γ has been introduced to balance the control performance across the training range. The effect of varying this parameter has been presented, demonstrating that, for a fixed number of hidden nodes in the controller, the parameter can control the relation between generalised controller per-

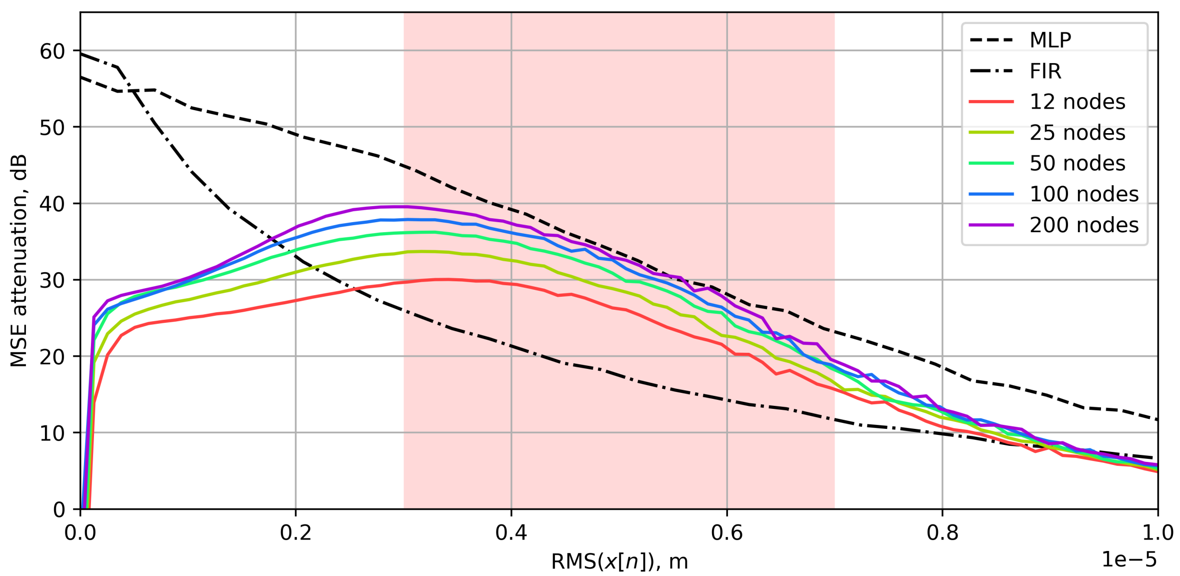

Figure 7: Generalised control performance of MLP controllers trained with varying numbers of nodes in the hidden layer, trained within the highlighted range. Each curve represents the average performance of 3 trained controller instances.

formance and reference signal magnitude. Increasing the controller training range has been shown to reduce controller performance over the same range. However, it has also been demonstrated that this performance may be recovered by increasing the number of hidden nodes in the MLP controller.

Additional contributions which would extend this work in the future could include the study of an extended set of NN architectures, a wider study of the NN hyperparameters, the study of a wider range of types of nonlinear behaviour, comparison of controller performance to other common nonlinear control strategies, as well as experimental validation of the results under similar nonlinear conditions.

Data Availability Statement

The data are available from the corresponding author on request.

Acknowledgements

This work was supported by an EPSRC Prosperity Partnership (No. EP/S03661X/1). The authors acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

References

[1] X. Pike and J. Cheer, "Limitations of FxLMS in feedforward active vibration control of a nonlinear two-degree-of-freedom system," *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, vol. 265, no. 2, pp. 5219–5229, Feb. 2023.

[2] M. Costa, J. Bermudez, and N. Bershad, "Stochastic analysis of the filtered-x LMS algorithm in systems with nonlinear secondary paths," *IEEE Transactions on Signal Processing*, vol. 50, no. 6, pp. 1327–1342, Jun. 2002.

[3] O. Tobias and R. Seara, "On the LMS algorithm with constant and variable leakage factor in a nonlinear environment," *IEEE Transactions on Signal Processing*, vol. 54, no. 9, pp. 3448–3458, Sep. 2006.

[4] S. Snyder and N. Tanaka, "Active control of vibration using a neural network," *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 819–828, Jul. 1995.

[5] L. Tan and J. Jiang, "Adaptive volterra filters for active control of nonlinear noise processes," *IEEE Transactions on Signal Processing*, vol. 49, no. 8, pp. 1667–1676, 2001.

[6] E. Reddy, D. Das, and K. Prabhu, "Fast exact multichannel FSLMS algorithm for active noise control," *Signal Processing*, vol. 89, no. 5, pp. 952–956, May 2009.

[7] C. Wangler and C. Hansen, "Genetic algorithm adaptation of non-linear filter structures for active sound and vibration control," in *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE.

[8] Q.-Z. Zhang and W.-S. Gan, "Active noise control using a simplified fuzzy neural network," *Journal of Sound and Vibration*, vol. 272, no. 1-2, pp. 437–449, Apr. 2004.

[9] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[10] M. Bouchard, B. Paillard, and C. L. Dinh, "Improved training of neural networks for the nonlinear active control of sound and vibration," *IEEE Transactions on Neural Networks*, vol. 10, no. 2, pp. 391–401, Mar. 1999.

[11] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, Mar. 1990.

[12] D. Hong, H. Lee, Y. Han, and B. Kim, "Numerical feasibility study for transverse vibration control of rotating shaft with a neural network-based tracking algorithm," *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, vol. 263, no. 5, pp. 1293–1298, Aug. 2021.

[13] H.-S. Kim, "Development of seismic response simulation model for building structures with semi-active control devices using recurrent neural network," *Applied Sciences*, vol. 10, no. 11, p. 3915, Jun. 5, 2020.

[14] C.-Y. Chang and F.-B. Luoh, "Enhancement of active noise control using neural-based filtered-x algorithm," *Journal of Sound and Vibration*, vol. 305, no. 1, pp. 348–356, Aug. 2007.

[15] Y. Yan, L. Dong, Y. Han, and W. Li, "A general inverse control model of a magneto-rheological damper based on neural network," *Journal of Vibration and Control*, vol. 28, no. 7, pp. 952–963, Apr. 2022.

[16] K. Na and S.-I. Chae, "Single-sensor active noise cancellation using recurrent neural network predictors," in *Proceedings of International Conference on Neural Networks (ICNN'97)*, vol. 4, Houston, TX, USA: IEEE, 1997, pp. 2153–2156.

[17] C. Chen and T.-D. Chiueh, "Multilayer perceptron neural networks for active noise cancellation," in *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96*, vol. 3, Atlanta, GA, USA: IEEE, 1996, pp. 523–526.

[18] M. Salmasi, H. Mahdavi-Nasab, and H. Pourghassem, "Comparison of feed-forward and recurrent neural networks in active cancellation of sound noise," in *2011 International Conference on Multimedia and Signal Processing*, Guilin, China: IEEE, May 2011, pp. 25–29.

[19] T. Matsuura, T. Hiei, H. Itoh, and K. Torikoshi, "Active noise control by using prediction of time series data with a neural network," in *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, vol. 3, Vancouver, BC, Canada: IEEE, 1995, pp. 2070–2075.

[20] K. Hiramoto and T. Matsuoka, "Active vibration control of structural systems with a preview of a future seismic waveform generated by remote waveform observation data and an artificial intelligence–based waveform estimation system," *Journal of Vibration and Control*, vol. 26, no. 17, pp. 1602–1613, Sep. 2020.

[21] Q. Wang, J. Wang, X. Huang, and L. Zhang, "Semiactive nonsmooth control for building structure with deep learning," *Complexity*, vol. 2017, pp. 1–8, 2017.

[22] J. Liu, X. Li, X. Zhang, and X. Chen, "Modeling and simulation of energy-regenerative active suspension based on BP neural network PID control," *Shock and Vibration*, vol. 2019, pp. 1–8, Jun. 18, 2019.

[23] H. Zhang and D. Wang, "Deep ANC: A deep learning approach to active noise control," *Neural Networks*, vol. 141, pp. 1–10, Sep. 2021.

[24] H. Zhang and D. Wang, "Deep MCANC: A deep learning approach to multi-channel active noise control," *Neural Networks*, vol. 158, pp. 318–327, Jan. 2023.

[25] R. Ranjan, J. He, T. Murao, L. Bhan, and W.-S. Gan, "Selective active noise control system for open windows using sound classification," 2016.

[26] Z. Qiu and W. Zhang, "Trajectory planning and diagonal recurrent neural network vibration control of a flexible manipulator using structural light sensor," *Mechanical Systems and Signal Processing*, vol. 132, pp. 563–594, Oct. 2019.

[27] M. McCloskey and N. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of Learning and Motivation*. Elsevier, 1989, pp. 109–165.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 2015.