30th International Congress on Sound and Vibration

ICSV30 AMSTERDAM 8-11 JULY 2024

NEURAL NETWORK-BASED CONTROL FOR VEHICLE SUSPENSION SYSTEMS WITH ACTIVE DAMPING

Archit Sharma
Birla Institute of Technology and Science, Pilani - Goa Campus, BE Mechanical Engineering
E-mail: f20190303g@alumni.bits-pilani.ac.inl
Pravin M Singru
Birla Institute of Technology and Science, Pilani - Professor, Department of Mechanical Engineering
E-mail: pmsingru@goa.bits-pilani.ac.in

The study presents a novel neural network-based method for adaptive control of active suspension systems. The neural net is initialized with random weights and initially relies on a traditional controller. The adaptive controller iteratively updates the weights using batch stochastic gradient descent (SGD), minimizing system deviation from the desired states. Results highlight an improvement in maximum overshoot and settling time compared to passive and Linear Quadratic Regulator (LQR) control algorithms under a step input. Furthermore, unlike traditional methods, the neural net controller exhibits adaptability across diverse road profiles.

Keywords: adaptive control, neural network, quarter car dynamics

1. Introduction

Vehicle suspension is a crucial system in any automobile to dampen the external forces and sudden movements that would otherwise damage internal components and be a detriment to drive quality. In most cases, the suspension system utilizes a damper or shock absorber. Damping systems are classified into passive, semi-active, and active, depending on whether they can exert an independent internal force. Semi-active systems such as magnetorheological dampers [1] actively modify damper parameters such as the damping constant to provide adaptive action. However, active systems, such as piezoelectric dampers [2], act on the system by applying a force. A controller is critical for such dampers. Adaptive control algorithms are commonly used for non-linear systems since they perform well for complex mathematical models diverging from the physical system. An adaptive controller reacts to the forces applied to the system to produce an optimal damping action.

Several methods, such as model reference adaptive, fuzzy logic, adaptive PID, adaptive skyhook, and neural network control, are used for adaptive damping. However, adaptive control has several significant disadvantages [3]. Adaptive controllers are generally not robust, and the system's stability is not rigorously maintained. Furthermore, the system may approach convergences slowly or suffer from high actuation cost.

The novel adaptive neural network control overcomes several limitations of neural networks and Adaptive controls in general. The control method proposed iteratively learns as the vehicle moves using a modified gradient descent algorithm[4]. Ideal network parameters and shallow architecture ensure

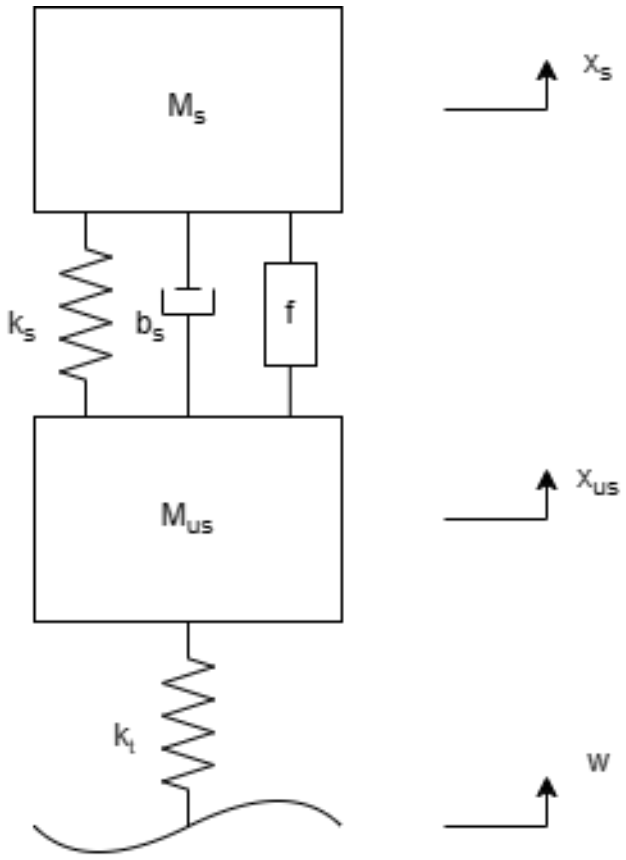

Figure 1: Quarter Car Model

that the required computational resources are low, the network does not overfit [5], and the controller converges rapidly. Thus, the controller has several improvements over traditional neural networks and other adaptive methods.

2. Model Description

The quarter car model, shown in figure 1, is the selected representation to analyse the behaviour of the vehicle suspension system. Equations 1 and 2 describe the 2 DOF system.

$$M_s\ddot{x}_s + b_s\dot{x}_s + k_s x_s - b_s\dot{x}_{us} - k_s x_{us} = f \tag{1}$$

$$M_{us}\ddot{x}_{us} + b_s\dot{x}_{us} + (k_s + k_t)x_{us} - b_s\dot{x}_s - k_s x_s = k_t w - f \tag{2}$$

Where M_s and M_{us} are the values of the sprung mass and suspension mass respectively, b_s is the damping constant of the suspension system, k_s and k_t are the spring constants for the suspension and the tire respectively [6].

Choosing state variable x to define the system,

$$x = \begin{bmatrix} x_s \\ x_{us} \\ \dot{x}_s \\ \dot{x}_{us} \end{bmatrix} \tag{3}$$

Table 1: Assumed values for model parameters

Parameter	Value
M_s	300 kg
M_{us}	60 kg
b_s	1000 Ns/m
k_s	16000 N/m
k_t	190000 N/m

Equations 1 and 2 can be rearranged to yield equation 4,

$$\begin{bmatrix} \dot{x_s} \\ \dot{x_{us}} \\ \ddot{x}_s \\ \ddot{x}_u s \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_s}{M_s} & \frac{k_s}{M_s} & -\frac{b_s}{M_s} & \frac{b_s}{M_s} \\ -\frac{k_s}{M_{us}} & -\frac{k_s+k_t}{M_{us}} & \frac{b_s}{M_{us}} & -\frac{b_s}{M_{us}} \end{bmatrix} \begin{bmatrix} x_s \\ x_{us} \\ \dot{x}_s \\ \dot{x}_u s \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{M_s} \\ \frac{k_t}{M_{us}} & -\frac{1}{M_{us}} \end{bmatrix} \begin{bmatrix} w \\ f \end{bmatrix} \tag{4}$$

Thus, the state space of the system is represented as,

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{5}$$

$$\forall \quad A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_s}{M_s} & \frac{k_s}{M_s} & -\frac{b_s}{M_s} & \frac{b_s}{M_s} \\ -\frac{k_s}{M_{us}} & -\frac{k_s+k_t}{M_{us}} & \frac{b_s}{M_{us}} & -\frac{b_s}{M_{us}} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{M_s} \\ \frac{k_t}{M_{us}} & -\frac{1}{M_{us}} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{k_s}{M_s} & \frac{k_s}{M_s} & -\frac{b_s}{M_s} & \frac{b_s}{M_s} \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{M_s} \end{bmatrix}$$

Since sprung mass position x_s and acceleration $\ddot{x}_s$ are readily observable, given y is chosen to describe state space output

$$y = \begin{bmatrix} x_s \\ \ddot{x}_s \end{bmatrix}$$

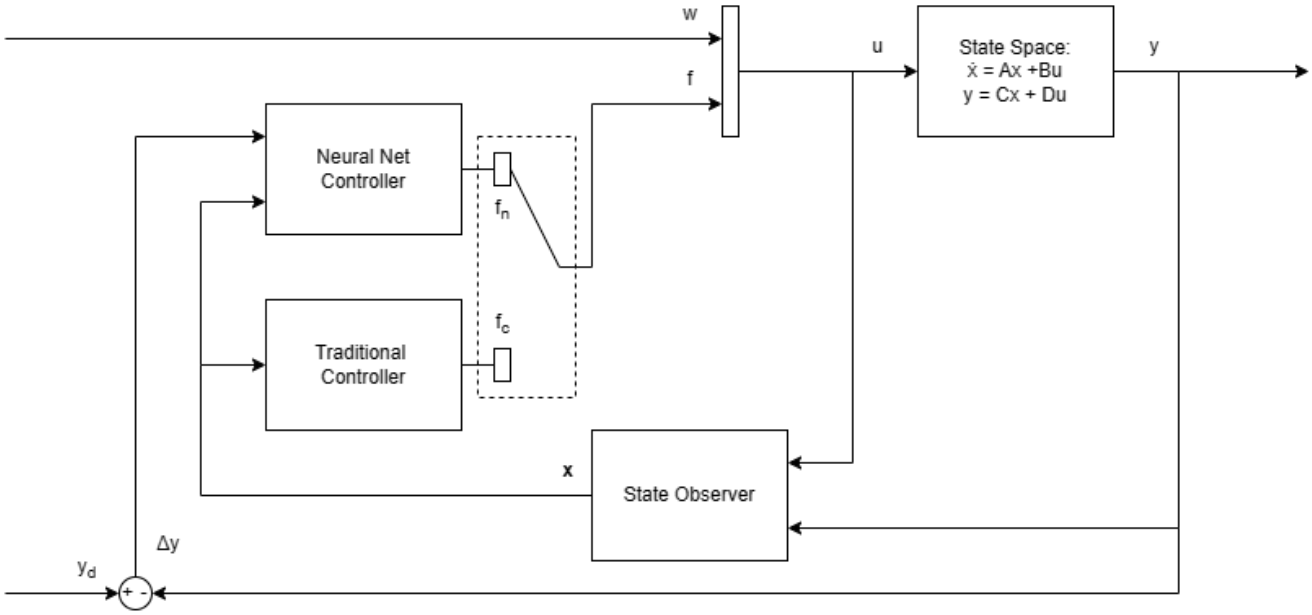

Figure 2: Controller Design

3. Controller Design

According to equation 5,

$$\dot{x} = g(x, u)$$
$$y = h(x, u)$$

where u and y are system input and output respectively. Objective of the controller is to minimize the difference between the desired observed output y_d and y. Thus, the loss function is chosen,

$$L = \frac{1}{2} Q(y - y_d)^2 \tag{6}$$

Where the output weight matrix Q is used to tune the loss function with respect to the output.
The control law uses a modified gradient descent algorithm to minimize the loss function L.

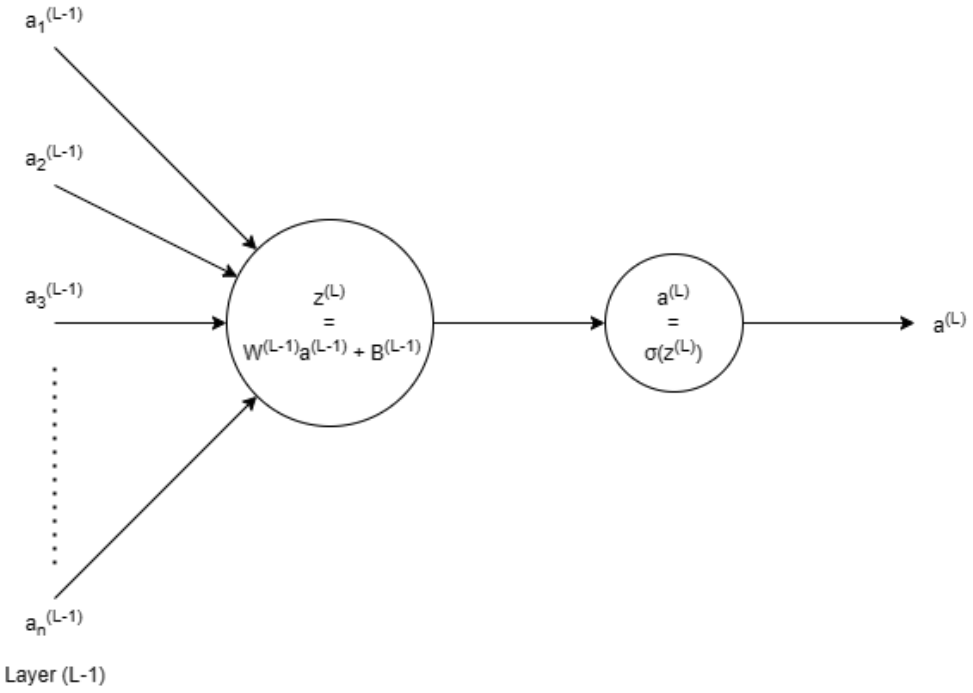

Figure 3: Neuron flow diagram

3.1 Output Law

The force applied by the active damper f_n is controlled by an adaptive multi-layered feed forward neural net (FNN). Output of the net is defined by set of equations 7 [4],

$$\begin{aligned} z^{(1)} &= W^{(i)} \cdot i + B^{(i)} \\ a^{(1)} &= \sigma(z^{(1)}) \\ \\ z^{(L)} &= W^{(L-1)} \cdot a^{(L-1)} + B^{(L)} \qquad \forall L \in [1, N] \\ a^{(L)} &= \sigma(z^{(L)}) \\ \\ z^{(o)} &= W^{(N)} \cdot a^{(N)} + B^{(N)} \\ f_n &= \sigma(z^{(o)}) \end{aligned} \tag{7}$$

Where W^l and B^l represent the weight and bias matrices respectively, z^l and a^l represent the hidden layer neuron and activation value respectively, and σ represent the activation function used to introduce non-linearity in the function.

3.2 Update Law

The network updates W^l and B^l matrices using a gradient descent algorithm. The algorithm iteratively approaches the minima of the loss function by taking steps towards the gradient of the function with respect to $\phi = w, b$ (w, b are elements of W^l and B^l matrices) according to [4],

$$\phi_{t+1} = \phi_t - \alpha \cdot \frac{\partial L}{\partial \phi}$$

Algorithm 1 Neural Net Learning Law

1: Create batch input state x and system error Δy using replay buffer
2: Update hidden layer values
3: **if** $\Delta y >$ Threshold **then**
4: Update leaky weight averages $V_{t+1} = \beta * V_t + (1 - \beta) * \frac{\partial C}{\partial \phi}$
5: Update weight and bias matrices $\phi_{t+1} = \phi_t - \alpha \cdot V_{t+1}$
6: **end if**
7: Calculate output f_n

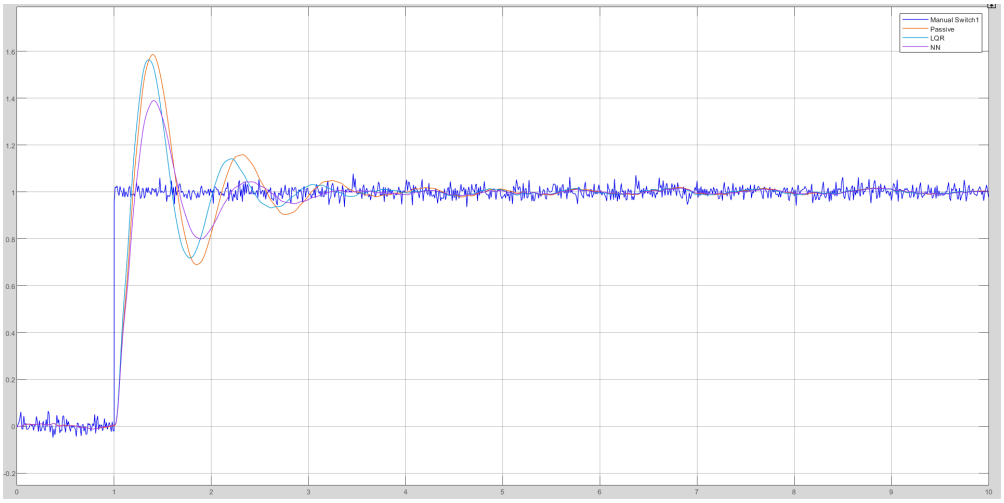

Figure 4: System response to unit step disturbance

$\frac{\partial L}{\partial \phi}$ is expanded using,

$$\frac{\partial L}{\partial \phi} = \frac{\partial z}{\partial \phi} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial L}{\partial a} \tag{8}$$

Where $\frac{\partial z}{\partial \phi}$ and $\frac{\partial a}{\partial z}$ are dependent on the hidden layer values and the network parameters, and computed accordingly. However, the value of $\frac{\partial L}{\partial a}$ is dependent on the gradient $\frac{\partial L}{\partial f_n}$ (14),

$$\frac{\partial L}{\partial f_n} = (y - y_d) \cdot (Ce^{tA} - CA^{-1}B_f + D_f)$$

Where,

$$B_f = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{M_s} \\ -\frac{1}{M_{us}} \end{bmatrix}$$

$$D_f = \begin{bmatrix} 0 \\ \frac{1}{M_s} \end{bmatrix}$$

4. Simulation and Results

A MATLAB Simulation of the model was implemented to verify the viability of the control strategy. The system was tested with passive, LQR, and the novel neural network controller.

Table 2: Assumed values for model parameters

Control Law	Overshoot (m)	Settling Time (t)
Passive	0.586	3.900
LQR	0.564	3.702
Neural Net	0.390	3.295

For the LQR control, the values of the weights were kept as[7],

$$Q_{LQR} = \begin{bmatrix} 1.2 \times 10^8 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10^6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_{LQR} = 1$$

The network architecture selected for the neural net control consists of 4 hidden layers with 16 neurons each.
The chosen batch size for the replay buffer was 1024. The learning rate α was 0.0025, and the leaky average parameter β was selected as 0.9. Additionally, the output weight Q was determined to be,

$$Q = \begin{bmatrix} 0.5 \times 10^{13} & 0.4 \times 10^8 \end{bmatrix}$$

The response of the system was observed under a unit step input with Gaussian noise of variance $0.5 \times 10^{-}3\ m^2$. Under passive control, figure 5 the model yielded a maximum overshoot of $0.586m$ and settling time of $3.900\ sec$.

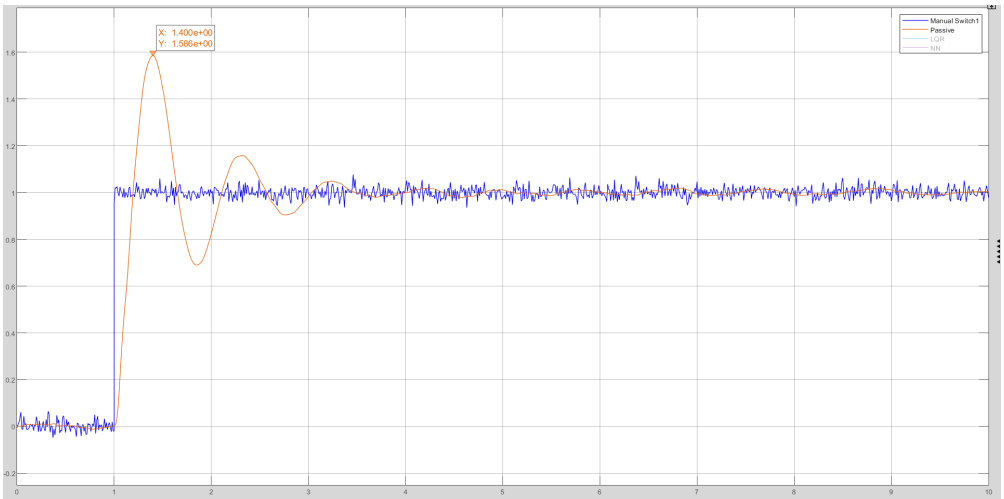

Figure 5: Sprung mass displacement under passive control

Comparing the system's response between LQR and the neural net control, figures 6 and 7, there's a 17.4% reduction in overshoot(%) and 0.407 sec reduction in the settling time. Figure 8 also studies the response of the controller against a unit step input without Gaussian noise to demonstrate settling error of the controller to be zero (within least count).

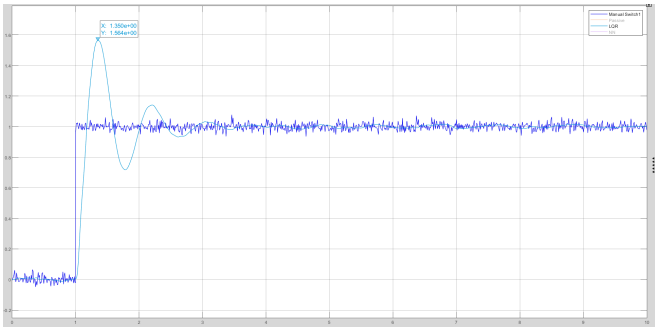

Figure 6: Sprung mass displacement(m) under LQR control

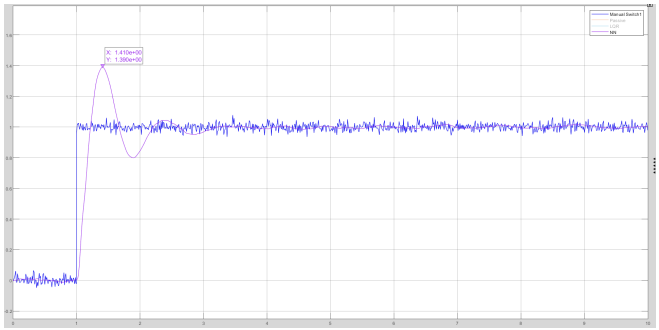

Figure 7: Sprung mass displacement(m) under neural net control

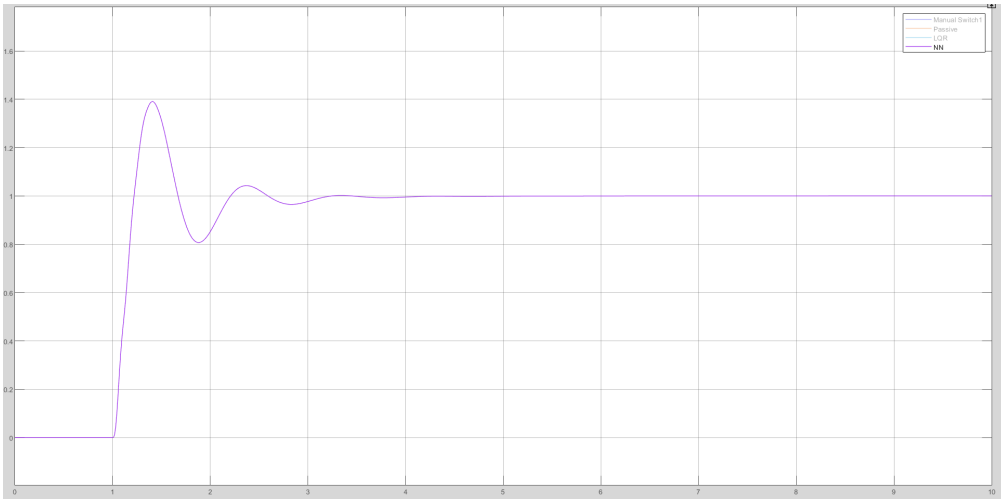

Figure 8: Neural net control response against unit step input (no Gaussian noise)

Furthermore, observing peak sprung mass acceleration in figures 9, the value of $\ddot{x}_s$ is $2.068 \times 10^2\ m/s^2$ for LQR control against $1.776 \times 10^2\ m/s^2$ for the neural net controller.

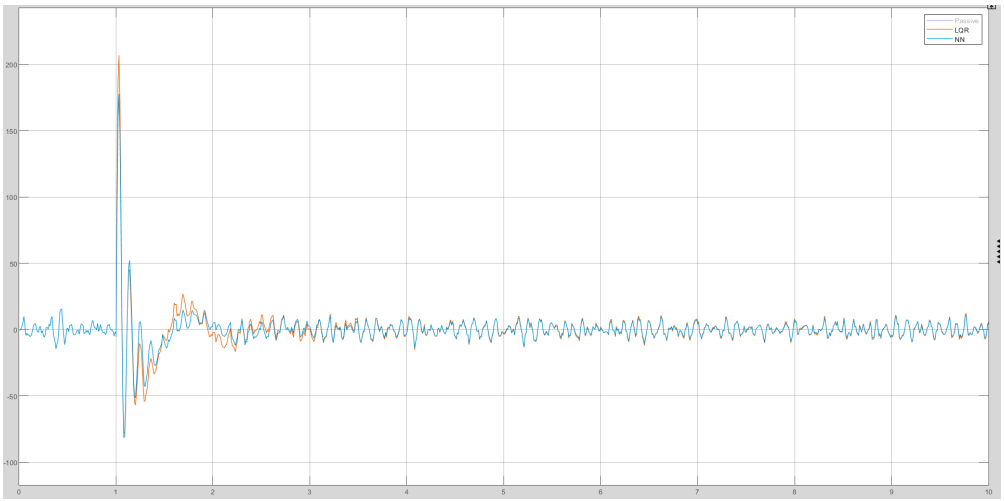

Figure 9: Sprung mass acceleration (m/s^2) under LQR and neural net control

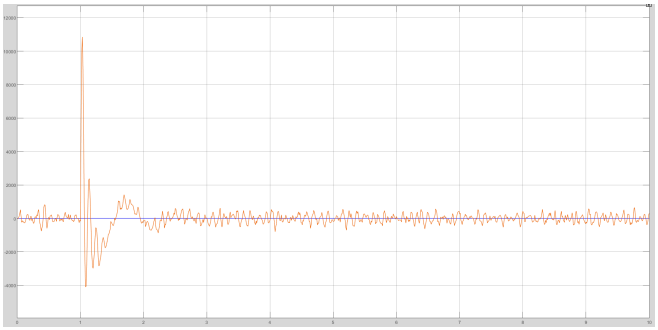

Figure 10: Damping force(N) applied under LQR control

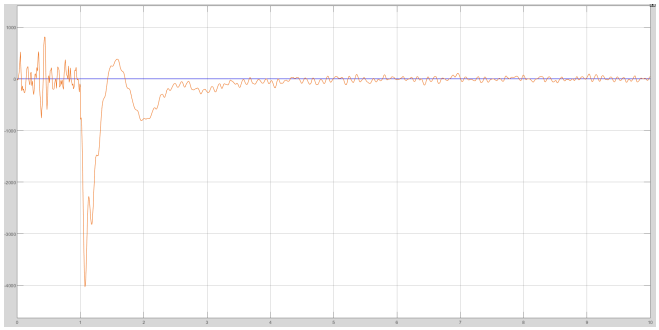

Figure 11: Damping force(N) applied under neural net control

The parameters for both controllers were selected while ensuring the force applied by the actuator does not exceed $10kN$. Despite of a more aggressive damping action, the damping force applied by the damper was much lower under the neural network controller. Figures 10 and 11, the peak damping force applied during LQR control is $10.85kN$ versus $3.99kN$ applied under the neural network control.

5. Conclusions

The paper proposes a neural network-based adaptive control method. The results obtained confirm the viability of the controller. The novel controller produces smaller overshoots and settling times for a step input response than a standard LQR controller. Furthermore, The suggested control method is less expensive regarding actuation cost, yielding a smaller actuation force for a more significant damping effect. The controller also shows rapid convergence, comparable to the LQR controller.

References

[1] G.Z. Yao et al. "MR damper and its application for semi-active control of vehicle suspension system". In: *Mechatronics* 12.7 (2002), pp. 963–973. ISSN: 0957-4158. DOI: https://doi.org/10.1016/S0957-4158(01)00032-0. URL: https://www.sciencedirect.com/science/article/pii/S0957415801000320.

[2] *Use of Piezoceramic Actuation for Automotive Active Suspension Mechanisms: A Feasibility Study*. Vol. 22nd Biennial Mechanisms Conference: Robotics, Spatial Mechanisms, and Mechanical Systems. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Sept. 1992, pp. 233–241. DOI: 10.1115/DETC1992-0220. eprint: https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/DETC92/09396/233/6670168/233_1_detc1992-0220.pdf. URL: https://doi.org/10.1115/DETC1992-0220.

[3] Necdet Özbek, Mert Önkol, and Mehmet Efe. "Feedback control strategies for quadrotor-type aerial robots: a survey". In: *Transactions of the Institute of Measurement and Control* 38 (Oct. 2015). DOI: 10.1177/0142331215608427.

[4] Joanne Quinn et al. *Dive into deep learning*. en. Thousand Oaks, CA: Corwin Press, Sept. 2019.

[5] Claudio Filipi Gonçalves Dos Santos and João Paulo Papa. "Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks". In: *ACM Comput. Surv.* 54.10s (Sept. 2022). ISSN: 0360-0300. DOI: 10.1145/3510413. URL: https://doi.org/10.1145/3510413.

[6] M.s Seong, S. Choi, and Kum-Gil Sung. "Control Strategies for Vehicle Suspension System Featuring Magnetorheological (MR) Damper". In: Sept. 2011. ISBN: 978-953-307-433-7. DOI: 10.5772/24556.

[7] https://github.com/GiacomoCorradini/Active_suspension/tree/main.. Accessed: 2024-1-11.

references

6. Derivation of Loss Gradient

Equation 8 expands the loss function, equation 6, using the chain rule. As mentioned, two terms of the function are calculated using parameters of the neural network itself but $\frac{\partial L}{\partial f_n}$ is required for the determination of the third term. We know,

$$L = \frac{1}{2}Q(y - y_d)^2$$

Expanding via chain rule again,

$$\frac{\partial L}{\partial f_n} = (y - y_d) \cdot \frac{\partial y}{\partial f_n} \tag{9}$$

from equation 5,

$$y = Cx + Du$$

Since,

$$D = \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{M_s} \end{bmatrix} \quad u = \begin{bmatrix} w \\ f \end{bmatrix}$$

$$D \cdot u = \begin{bmatrix} 0 \\ 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{1}{M_s} \end{bmatrix} f$$

Let,

$$D_f = \begin{bmatrix} 0 \\ \frac{1}{M_s} \end{bmatrix}$$

Thus,

$$y = Cx + D_f f \tag{10}$$

$f = f_n$ for the system after the time delay switches to the neural network control.

$$\frac{\partial y}{\partial f_n} = C\frac{\partial x}{\partial f_n} + D_f \tag{11}$$

Also from equation 5,

$$\frac{\partial x}{\partial t} = Ax + Bu$$

Similarly to equation 10,

$$\frac{\partial x}{\partial t} = Ax + B_f f_n \quad \forall \quad B_f = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{M_s} \\ -\frac{1}{M_{us}} \end{bmatrix} \tag{12}$$

Derivating the equation 12 with respect to f_n,

$$\frac{\partial^2 x}{\partial f_n \partial t} = A\frac{\partial x}{\partial f_n} + B_f$$

assuming $q = \frac{\partial x}{\partial f_n}$,

$$\frac{\partial q}{\partial t} = Aq + B_f \tag{13}$$

Equation 13, general solution to the differential of the form is,

$$q = e^{tA} - A^{-1}B_f$$

$$\frac{\partial x}{\partial f_n} = e^{tA} - A^{-1}B_f$$

Substituting in equation 11,

$$\frac{\partial y}{\partial f_n} = Ce^{tA} - CA^{-1}B_f + D_f$$

Substituting in equation 9,

$$\frac{\partial L}{\partial f_n} = (y - y_d) \cdot (Ce^{tA} - CA^{-1}B_f + D_f) \tag{14}$$