2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)
June 28-30, 2023. Seattle, Washington, USA

Deep Neural Network Design for Improving Stability and Transient Behavior in Impedance Control Applications

Jonathon E. Slightam
Unmanned Systems & Autonomy R&D
Sandia National Laboratories
Albuquerque, NM, USA

Antonio D. Griego
Department of Computer Science
University of New Mexico
Albuquerque, NM, USA

***Abstract*—Robot manipulation of the environment often uses force feedback control approaches such as impedance control. Impedance controllers can be designed to be passive and work well while coupled to a variety of dynamic environments. However, in the presence of a high gear ratio and compliance in manipulator links, non-passive system properties may result in force feedback instabilities when coupled to certain environments. This necessitates an approach that ensures stability when using impedance control methods to interact with a wide range of environments. We propose a method for improving stability and steady-state convergence of an impedance controller by using a deep neural network to map a damping impedance control parameter. In this paper, a dynamic model and impedance controlled simulated system are presented and used for analyzing the coupled dynamic behavior in worst case environments. This simulation environment is used for Nyquist analysis and closed-loop stability analysis to algorithmically determine updated impedance damping parameters that secures stability and desired performance. The deep neural network inputs utilized present impedance control parameters and environmental dynamic properties to determine an updated value of damping that improves performance. In a data set of 10,000 combinations of control parameters and environmental dynamics, 20.3% of all the cases result in instability or do not meet convergence criterion. Our deep neural network improves this and reduces instabilities and failed control performance to 2.29%. The design of the network architecture to achieve this improvement is presented and compared to other architectures with their respective performances.**

I. Introduction

Guaranteeing stability is a core technical challenge for robot manipulation in unstructured environments. Of particular interest, is the ability of a robot manipulator to rapidly attain stability while performing a manipulation task with an environment of yet to be determined dynamic properties.

This work was supported by Laboratory Directed Research and Development program at Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA000352. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. [emails: jslight@sandia.gov, deano505@unm.edu]

Early manipulators were directly controlled by human operators. For example, Goertz developed manipulators to facilitate interactions with dangerous radioactive materials for laboratory work [1]. These systems maintained stability due to the direct control of the human operators coupled with sensors in the manipulators that provided force reflection.

Eventually, control systems were developed for robotic arms that made use of computers in order to enable automatic control without human intervention. Many different types of force feedback control schemes exist in the literature including pure force control, impedance control, admittance control, hybrid position-force control, passive stiffness control with impedance control, and others [2]–[4].

Of primary concern with control schemes like impedance control is that the environment being interacted with needs to be precisely defined and known in advance of any manipulation task. In addition to this concern, many lightweight mobile manipulators have flexibility in links and high gear ratios which may result in non-passive behavior when implementing impedance control. With on-line system identification of the environment, it is possible to adapt to the environment. This motivates using prior knowledge of closed-loop coupled system performance to define new system parameters when the environment properties are identified. This paper presents a machine learning approach that updates a dissipative impedance control parameter (damping) to improve performance. The paper's objective is to demonstrate the feasibility of using neural networks as a tool for mapping control parameters that will attain desired control performance for contact rich manipulation tasks.

II. Background

Previous work has been established using non-machine learning methods for learning impedance control parameters. Kim proposed a recursive least-square filter-based episodic natural actor-critic algorithm in order to find optimal impedance control parameters [5]. Arimoto presented a physical interpretation of practice-based learning that steadily learns a desired task by monotonously increasing the grade of impedance matching pertaining to the dynamics of the

robot task with controller dynamics [6] . Cheah introduced a method for learning impedance control for robot manipulators [7]. Unlike other approaches, Cheah's method is implemented without the need to switch the learning controller between non-contact and contact tasks. Wang presented an iterative learning control law for the impedance control of robotic manipulators [8]. This method does not require a reference trajectory for training and instead the performance is determined by a target impedance.

This foundational work demonstrates the effectiveness of impedance control and various novel techniques for learning impedance control parameters. However, most of these approaches are brittle in the sense that they focus on teaching one specific task or interaction to the controller and they do not generalize to new environments without completely retraining the controller for each new scenario. It is the hope that this work can build off the ideas of these methods and develop an approach to generalize a way to adapt impedance control parameters for stability no matter what environment a robot may interact with.

In addition to traditional techniques described previously, there has also been a lot of work in using machine learning techniques to learn impedance control parameters. Tsuji developed a method to use an array of neural networks to regulate the impedance parameters of a manipulator's end-effector while identifying environmental characteristics through on-line learning [9]. Jung proposes a method using neural networks to compensate for uncertainties in the robot model by using a novel error signal for the neural network training [10]. Katic presents an application of connectionist structures for fast and robust online learning of internal robot dynamic relations used as part of impedance control strategies in the case of robot contact tasks [11]. Cohen presents an evaluation of the associative search network (ASN) learning scheme which is a stochastic scheme that uses a single scalar value as a measure of the system performance [12].

Previous work using machine learning methods has provided solid evidence that neural networks can be applied effectively to solve impedance control parameter learning problems. This work has either focused on using neural networks to train an actuator on learning specific tasks or focuses on using online learning methods.

A. Contributions

This paper presents a supervised machine learning method (neural networks) that guarantees stability using a passivity approach with additional transient criterion. In our proposed approach, a damping parameter in an impedance controller is mapped to meet this criterion as a function of current control parameters and estimated environment stiffness and mass. This approach is applied to a simulated 1-degree-of-freedom (DOF) model utilizing simple impedance control. A function fitting network and deep neural network are created using Matlab's Deep Learning Toolbox and the performance of these different neural network architectures are compared with a before and after use case scenario in simulation.

III. Modeling

In this section, we present a single DOF robot actuator and link model that we used in data generation and simulations with closed-loop impedance control.

A. System Model Dynamics

A high level diagram of a single DOF robot joint is illustrated in Fig. 1.

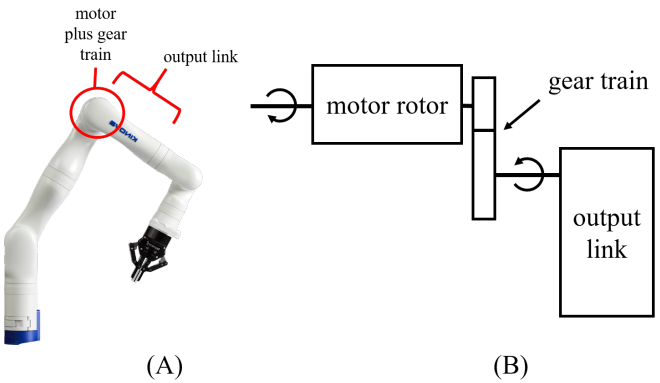

Fig. 1. High level representation of a robot arm segment as a 1-DOF robot actuator.

As described in the introduction, high gear train ratios and light-weight structures that introduce compliance in the robot links may potentially lead to non-passive properties and instability when coupled to certain environments. This paper considers systems with these properties.

A model of the single DOF robot actuator and link system is illustrated in Fig. 2.

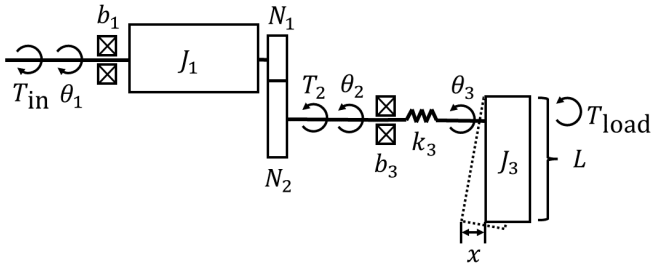

Fig. 2. Model diagram for a 1-DOF robot arm actuator.

In Fig. 2, θ_1 is the angle of the motor rotor, θ_2 is the output angle of the gearbox model, θ_3 is the angle of the robot link, L is the length of the output link, T_{in} is the input torque, b_1 is the viscous friction coefficient of the motor model, J_1 is the motor rotor inertia, N_1 and N_2 are the number of teeth on the input and output gears, respectively, b_3 is the viscous friction coefficient of the output shaft, k_3 is the combined angular stiffness of the output shaft and link, J_3 is the inertia of the output link, and T_{load} is the external torque load on the robot link. The values of system parameters are defined in Table I.

The resulting equations of motion for the motor and gear train model in Fig. 2 are derived using Newton's second law and result in Eq. 1 for motion at θ_2

$$\ddot{\theta}_2 = \left(\frac{N_1}{J_1 N_2}\right) T_{\text{in}} - \frac{1}{J_1}\left(\frac{N_1}{N_2}\right)^2 T_2 - \frac{b_1 \dot{\theta}_2}{J_1}, \tag{1}$$

where Eq. 1 is a consolidated form with thorough derivations shown in [13]. The output motion of θ_3 is defined as Eq. 2,

$$\ddot{\theta}_3 = \frac{T_{\text{load}}}{J_3} - \frac{b_3}{J_3}\left(\dot{\theta}_3 - \dot{\theta}_2\right) - \frac{k_3}{J_3}\left(\theta_3 - \theta_2\right). \tag{2}$$

Eq. 1 can be simplified by using the relation, $T_2 = b_3(\dot{\theta}_2 - \dot{\theta}_3) + k_3(\theta_2 - \theta_3)$. Thus, when substituting this relationship and simplifying we have Eq. 3,

$$\begin{aligned} \ddot{\theta}_2 = \frac{T_{\text{in}} N_1}{J_1 N_2} - \frac{b_1 \dot{\theta}_2}{J_1} - \\ \frac{1}{J_1}\left(\frac{N_1}{N_2}\right)^2 \left[b_3\left(\dot{\theta}_2 - \dot{\theta}_3\right) + k_3\left(\theta_2 - \theta_3\right)\right], \end{aligned} \tag{3}$$

where the resulting output translation is simply: $x = \theta_3 L$. The equations of motion can be put into state-space form as

$$\dot{x} = \mathbf{A}x + \mathbf{B}u, \tag{4}$$

where u is the control input vector. Eq. 4 can be translated to the S-domain in Matlab and be used with the connect function to synthesize the desired closed-loop impedance controller with the model.

B. Control System

A simple impedance control law was implemented for the 1-DOF dynamic model, where the positional error is defined by Eq. 5

$$e = x_{ref} - x, \tag{5}$$

with x_{ref} defined as the reference or desired position. The error can be used with the emulated stiffness, K_e, and when differentiated can be used with the emulated damping, B_e, and emulated inertia, M_e, along with the desired and actual forces on the actuation system, giving

$$u = F_D + F_{\text{load}} + M_e \ddot{e} + B_e \dot{e} + K_e e. \tag{6}$$

A proportional-derivative controller is then wrapped around the desired impedance behavior of the actuation system, defined by Eq. 7.

$$T_{\text{in}} = K_p u + K_d \dot{u} \tag{7}$$

The dynamic system and control parameters are listed in Table I, which includes the desired possible ranges of impedance parameters.

TABLE I
SYSTEM PARAMETERS

Parameter	*Value*
J_1	0.001 $[kg - m^2]$
B_1	1 $[N - m - s/rad]$
N_1	1 $[\#]$
N_2	100 $[\#]$
B_3	0.01 $[N - m/rad]$
K_3	5000 $[N - m/rad$
J_3	0.25 $[kg - m^2]$
M_e	(0.01 : 1 : 5) $[kg]$
B_e	$2\sqrt{M_e K_e}$ $[N - s/m]$
K_e	(31.25 : 125 : 625) $[N/m]$
K_p	100
K_d	1.0

C. Simulated Validation and Passivity Analysis

The model and control system were validated for a number of system parameters. Fig. 3 illustrates the ability of the closed-loop impedance controller to track desired response from a resulting step input force when $M_e = 1$, $B_e = 22.4$, and $K_e = 125$.

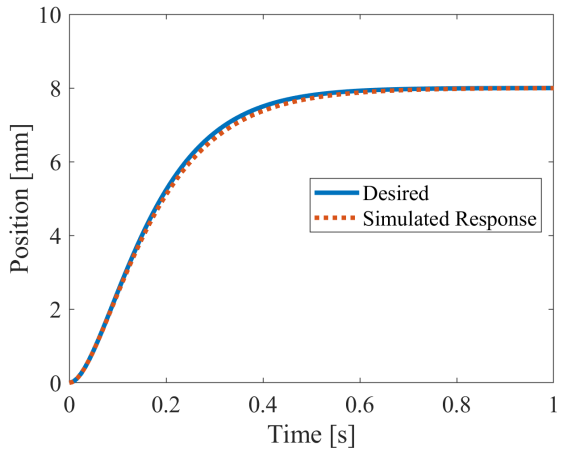

Fig. 3. Step force input response validation of model and controller. The blue solid line indicates the desired response as a result of a step input force and the red dashed line is the simulated response due to a step input force.

Fig. 3 indicates that the control parameters for the impedance controller provide sufficient performance to achieve the desired second-order impedance. However, from an analysis perspective and for the desired impedance parameters chosen in the case of Fig. 3, instabilities may arise when the system is coupled with a certain environment if the system is non-passive. By definition, a system's impedance is passive if all the real components of the impedance are greater than or equal to zero as defined by Eq. 8.

$$Re(Y(j\omega)) \geq 0 \quad and \quad Re(Z(j\omega)) \geq 0, \tag{8}$$

where $Y(j\omega)$ is the system's admittance and $Z(j\omega)$ is the system's impedance. By looking at the Nyquist plot of the desired impedance and the simulated model and control system's

impedance we can determine if the simulated control system is in fact passive. The Nyquist plot is shown in Fig. 4.

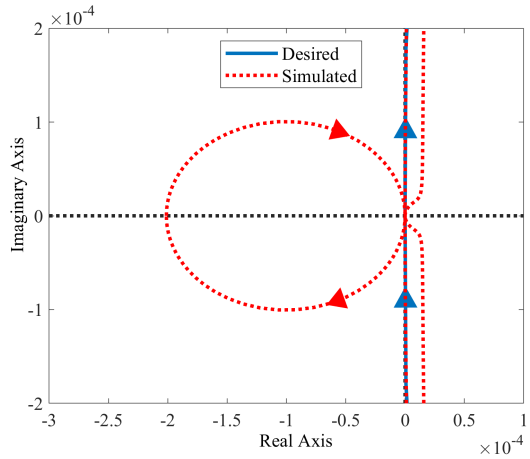

Fig. 4. Nyquist plot of desired impedance (solid blue line) and the impedance of the simulated dynamics and control system (red dashed line).

Fig. 4 identifies that our controlled system is non-passive and may result in instability when coupled to certain environments. This non-passivity is likely due to higher gear train ratios and flexibility in the output link. For the entire range of desired impedance parameters only 8% result in the system being passive. However, when decreasing drive train reduction to $N = 10$ and the output link stiffness to $K = 500,000N-m/rad$, 72% of the parameter combinations in the data set result in passivity. This suggests that many lightweight, high gear reduction drive robot arms may have difficulty in ensuring passivity and subsequently stability when coupled to the environment.

IV. Stability Analysis

Due to non-passivity of the system, it is important to consider the analysis of coupled stability with the environment. In this paper, we implement methods discussed in [14]–[16]. We consider the worst case scenario in our analysis, i.e. a mass-spring model of the environment. Fig. 5 illustrates the different block diagrams for the closed-loop and open-loop transfer functions included in the analysis.

In Fig. 5(a), the block diagram of the closed-loop impedance can be represented as Eq. 9.

$$G(s)_{CL} = \frac{Y(s)_{sys}Z(s)_{env}}{1 + Y(s)_{sys}Z(s)_{env}}, \tag{9}$$

where the subscript env denotes the environment that is comprised of the environment mass, M_{env}, and environment stiffness, K_{env}. Eq. 9 can be looked at in the open-loop form to check for stability as shown in Fig. 5(b). The open-loop form of the coupled system is defined by Eq. 10.

$$G(s)_{OL} = Y(s)_{sys}Z(s)_{env} \tag{10}$$

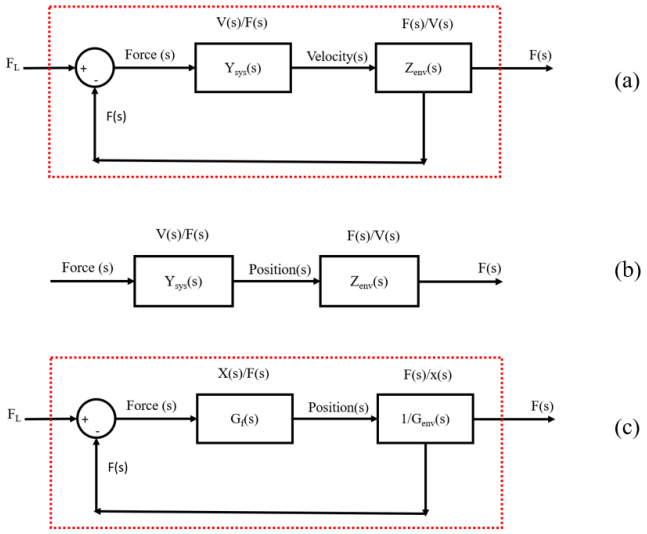

Fig. 5. Block diagrams for (a) closed-loop admittance and impedance of system and environment, (b) open-loop admittance and impedance of the system and environment, and (c) closed loop coupling of position-force of the system and environment.

where Eq. 10 can be used to identify instability of the coupled system by identifying if the critical point is encircled. The closed-loop coupled system can also be looked at in the position-force domain to identify marginal stability and additional controller transient constraints, e.g., settling time, etc. This closed-loop formulation as described in Fig. 5(c) is defined as Eq. 11.

$$G(s)_{CLp} = \frac{G(s)_{sys}(1/G(s)_{env})}{1 + G(s)_{sys}(1/G(s)_{env})} \tag{11}$$

V. Neural Network Design

In the neural network design, we aim to use the theoretical foundations in Fig. 5 and Eqs. 9-11 to suggest improved control parameters algorithmically and embed this analysis and ensured improvement in coupled control. We consider two different types of neural networks in Matlab's Deep Learning Toolbox: function fitting neural networks (FITNET) and deep learning networks (DL Networks), also referred to as DNNs herein. This section introduces the two types of networks and how we approached solving this problem with machine learning.

A. Function Fitting Neural Network

The function fitting neural network (FITNET) is a standard regression/function fitting neural network architecture available in Matlab. FITNET allows the user to completely decide the size and structure of the network. This includes the number of hidden layers and the number of neurons in each hidden layer. Different variations in number of neurons per layer and number of layers were tested to see which configuration produced the best training results. We trained and tested 1,

2, 4, 8, and 12 hidden layers with different combinations of neurons per layer, which included 1, 10, 100, 500, and 1000 neurons per hidden layer.

The benefits of this type of network over the Deep Learning Network is that the speed of training is generally faster with this system.

B. Deep Neural Network

The deep learning network or DNN is the other neural network that was tested in this work. Just as FITNET provided many variables and settings to adjust, the DL Network provides more flexibility and control than FITNET. The DL Network tool allows each individual layer in the network to be customized and adjusted. While this feature is available to FITNET, the ability to do so with the DL Network is significantly more streamlined and easier to use.

Each hidden layer was designed to be fully connected and the ReLu activation function is used between each layer. The same number of layers and neurons per layer were used as FITNET in order to provide a direct comparison of performance between the two neural network structures.

While extremely flexible, this comes at the cost of run time. This network typically takes more time to train than FITNET.

C. Data Generation and Problem Setup

The data set generation uses a combination of Eqs. 9-11 to analyze the simplified simulated system when it is coupled to a sweep of mass-spring environments while using a variety of different desired stiffness or inertia values in the controller, where the range of these desired impedance parameter values and environment values are in the algorithm described in 1. In algorithm 1, we incrementally adjust the damping parameter of the impedance controller to help ensure stability, which is then used for the training data set.

The core algorithm for training data generation can be described in four distinct steps. First, the ranges of values for the impedance controller and the environment being analyzed are generated including the inertial elements, stiffness values, and damping. Three sets of values are generated for B_e, which is the desired output of the neural network used in training: B_{e-MIN} which is calculated to be the critical damping for the robot system, B_{e-MAX} which will be the maximum stable damping calculated within a defined bound, and $B_{e-MEDIAN}$ which is the median value between B_{e-MIN} and B_{e-MAX}. It is also the value of B_e used for training. The training data input is derived by generating all possible combinations of the parameter vectors J_e, K_e, M_{env}, and K_{env} which define the robot system and its coupled environment.

Second, the algorithm verifies the minimum stable damping (B_{e-MIN}) for all parameter combinations in the data set. Using stability analysis, it is determined if the default value for the minimum damping remains stable when coupled to the defined environment. If it is not, the value of the damping is incrementally increased by 5% until a stable solution is found. It is important to note that we limited the settling time for the system to be less than ten seconds. Any set of parameters that takes longer than ten seconds to settle would not be feasible and are thus discarded and considered as unstable samples.

Algorithm 1: Data Set Generation

$J_e \leftarrow [0.01 : 1.0 : 5.0]$;
$K_e \leftarrow [31.25 : 125.0 : 625.0]$;
$M_{env} \leftarrow [0.01 : 10.0 : 100.0]$;
$K_{env} \leftarrow [0.0 : 2.5e^2 : 1e^4]$;
$B_{e-MIN} \leftarrow 2\sqrt{Ke \times Je}$;
$B_{e-MAX} \leftarrow Be_{MIN}$;
$B_{e-MEDIAN} \leftarrow Be_{MIN}$;
trainingData $\leftarrow$ combvec$(Je, Ke, M_{env}, K_{env})$;
for ***all*** $t(i) \in$ *trainingData* **do**
  while *isStable*$(t(i), B_{e-MIN}(i)) \rightarrow$ *false* **do**
    $B_{e-MIN}(i) \leftarrow B_{e-MIN}(i) \times 1.05$;
  end
  $B_{e-MAX}(i) \leftarrow B_{e-MIN}(i)$;
  $B_{e-WINDOW} \leftarrow \varnothing$;
  while *isStable*$(t(i), B_{e-MAX}(i)) \rightarrow$ *true* ***and*** $B_{e-MAX} < (10 \times B_{e-MIN}(i))$ **do**
    $B_{e-WINDOW} \leftarrow (B_{e-WINDOW}) \cup (B_{e-MAX}(i))$;
    $B_{e-MAX}(i) \leftarrow B_{e-MAX}(i) \times 1.05$;
  end
  $B_{e-MEDIAN}(i) \leftarrow$ median$(B_{e-WINDOW})$;
end

Third, the maximum stable damping values are next calculated using the minimum stable damping as the starting point. Once again, the damping is incremented by 5% upon each iteration and stability analysis is done to ensure each damping value produces a stable coupled system with a settling time less than ten seconds. Each calculated value is stored in an intermediary vector of stable damping values, $B_{e-WINDOW}$. The cutoff for B_{e-MAX} is either when stable samples are no longer generated or when the value of B_{e-MAX} exceeds $10 * B_{e-MIN}$.

Finally, once $B_{e-WINDOW}$ has been fully calculated, $B_{e-MEDIAN}$ is simply set to be the median value contained in the vector of stable samples in $B_{e-WINDOW}$, which is the newly desired B_e. The data set algorithmically determined B_e and was verified to ensure stability and desired control specifications in 100% of the data, whereas the baseline results in over 20% of the data set being unstable or not meeting our control criterion. Considering B_e as our desired output, the inputs to the neural network are J_e, K_e, M_{env}, and K_{env} which we can then design the neural network architecture around. The environment parameters and current impedance control parameters are needed in order to map a function to B_e, as $B_e = f(J_e, K_e, M_{env}, K_{env})$ as a result of using Eqs. 9-11.

D. Network Training

Settings and parameters for the FITNET and DL Networks were chosen in order to be reproducible and to provide

a reasonable baseline for comparison. Both networks were generated using consistent random number seeds, so that they could be reproduced. Additionally, both networks were trained to complete up to five hundred epochs. Unless otherwise noted, all parameters were set to their default values.

When training a FITNET neural network the following network specific parameters were used:

1) net.trainFcn — the backpropagation algorithm used in training (scaled conjugate gradient backpropagation).
2) net.divideFcn — the training data was divided randomly.
3) net.divideMode — the training data was divided into training, validation, and testing sets according to the defined ratios for training (70%), validation (15%), and testing (15%).
4) net.performFcn — the performance function used during training is the mean squared error.
5) net.trainParam.goal — the performance goal measured by the MSE, which is set to zero.
6) net.trainParam.min_grad — the minimum performance gradient, set to $1e^{-300}$.
7) net.trainParam.max_fail — the maximum number of validation failures, set to 1000.
8) net.trainParam.sigma — change in weight for ddot approximation.
9) net.trainParam.lambda — parameter for regulating the indefiniteness of the Hessian.
10) net.trainParam.time — the max time to train, set to infinite such that the network will take as long as necessary for training.

When training a DL Network the following network specific parameters were used:

1) solverName — solver for training the network, specificed as RMSProp (root mean square propagation).
2) InitialLearnRate — the initial learning rate for the RMSProp algorithm when training. Set to $1e-4$.
3) for the feature input layer, normalization of input data is set to none.

VI. Neural Network Design Results

After training the different network architectures for the DNN and FITNET, we analyzed their performance, by quantifying the total percentage of the data set that is stable with the network using the newly suggested B_e, as well as the mean-squared-error (MSE), and time it takes to execute the network in milliseconds. The results are tabulated in Table II and III in the appendix. The statistics of the network's performance are summarized in Fig. 6.

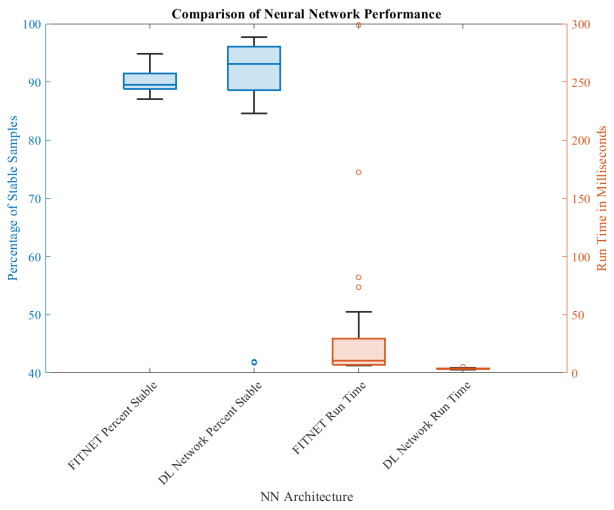

Fig. 6. Statistics of FITNET and DL Network neural networks.

Fig. 6 shows that the DNNs used with the DL Network tool outperformed FITNET, guaranteeing that 97.7% of the data set would meet our control requirements, all while executing the network in under 3 ms. While the DL Network had some outliers in how stable the dataset was, on average 93.1% was stable, while FITNET's average was 89.5%. FITNET's average time for execution was 10.3 ms with a minimum of 6.4 ms and maximum of 298 ms. DL Network's average execution time was 3.2 ms with a minimum of 3.0 ms and maximum of 5.2 ms.

The best performing DNN architecture was one using 2 hidden layers and 10 neurons for each layer. Fig. 7 shows this top performing network architecture with its respective inputs and output.

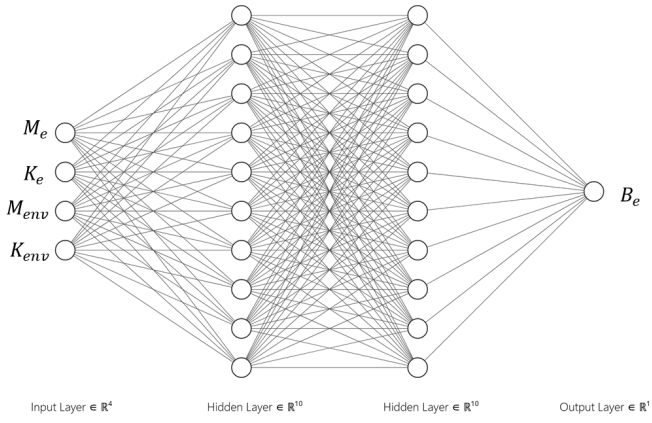

Fig. 7. Best performing NN, with 2 hidden layers and 10 neurons in each hidden layer.

To show the utility of the network trained we selected desired impedance parameters and environment that resulted in non-desired transient response, i.e., did not settle within a

certain time period specified (10s). We used the best performing DL Network to suggest a new damping parameter and compared a before and after step response of both systems using Eq. 11, where this formulation is the closed-loop system depicted in Fig. 5(c). The resulting step responses for this exemplar is displayed in Fig. 8. The initial impedance and adjusted impedance parameters are listed in the legend of Fig. 8.

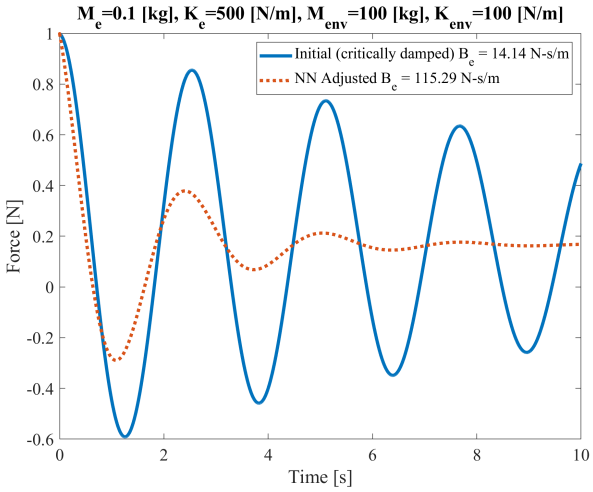

Fig. 8. Closed-loop step response of coupled system with original damping value (solid blue line) and NN adjusted damping value (red dashed line).

Fig. 8 shows that our ML approach can be used to parameterize control performance using closed-loop transient and open-loop stability properties of coupled systems. Implementation in real-time applications, whether simulation or hardware would require the NN to be integrated with system identification methods, such as those presented by Park et al., which can be used to identify environment model parameters that can be fed into the NN [17].

VII. Discussion

This paper presented a machine learning approach to improve control performance of impedance control systems when coupled to the environment to increase stability and achieve desired transient performance.

The machine learning approach presented utilizes analysis tools that have been presented in prior work in [14]–[16], while leveraging machine learning to look at a large number of different plausible scenarios in unstructured manipulation problems. It must be considered that this approach presents the machine learning algorithms and design, and in order to be implemented, this approach would need to be complimented with a system identification approach to feed environment parameters into the DNN [9], [17], [18]. We propose our created DNN to run in parallel with a parameter estimation tool that can update an impedance controller with parameters that can be varied in real-time.

Our proposed ML approach can also be used with other analysis tools, such as pole placement techniques for closed loop control design [19]. Other advanced nonlinear techniques for control design analysis such as those described by Babarahmati et al. and methods used by Ficuciello et al. for redundant manipulators can also be leveraged for our proposed machine learning framework [20], [21]. The FITNET and DNN architectures and training methods we proposed are agnostic to what approaches are used for data generation. In addition to this, our proposed approach can be used with different control architectures that utilize force feedback when analysis techniques can be used to ensure stability and specific performance criterion [3], [15], [21].

The designed DNN suggest that this approach can aid in control engineering problems that can leverage large amounts of data and consider many different scenarios an impedance control system may encounter in its lifetime.

VIII. Conclusions

In this work, two different approaches for neural network design were considered to aid in improving impedance control performance when coupled to the environment. It was found that using DNNs (DL networks) provided superior performance in ensuring stability when coupled to the environment and overall execution time. The best performing network had 2 hidden layers with 10 neurons in each hidden layer and ReLu activation functions between hidden layers. The percent stable was improved to 97.7% from 79.7%, while executing in 2.99 ms. It is thought that with additional data and tuned hyper-parameters, this performance can be improved. This proposed approach provides a way to update control parameters rapidly by using machine learning techniques with coupled and transient control analysis with impedance controllers. The results presented in this work are highly relevant to lightweight manipulators that have high gear ratio drive trains with non-passive characteristics.

References

[1] D. Whitney, "Historical perspective and state of the art in robot force control," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 262–268.

[2] N. Hogan, "Impedance control: An approach to manipulation," in *1984 American control conference*. IEEE, 1984, pp. 304–313.

[3] J. E. Slightam, M. Nagurka, and E. Barth, "Sliding mode impedance control of a hydraulic artificial muscle," in *Dynamic Systems and Control Conference*, 2018.

[4] J. E. Slightam, E. Barth, and M. Nagurka, "Sliding mode impedance and stiffness control of a pneumatic cylinder," in *Dynamic Systems and Control Conference*, 2019.

[5] B. Kim, J. Park, S. Park, and S. Kang, "Impedance learning for robotic contact tasks using natural actor-critic algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 2, pp. 433–443, 2009.

[6] S. Arimoto, P. Nguyen, and T. Naniwa, "Learning of robot tasks via impedance matching," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 4. IEEE, 1999, pp. 2786–2792.

[7] C.-C. Cheah and D. Wang, "Learning impedance control for robotic manipulators," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 3, pp. 452–465, 1998.

[8] D. Wang and C. C. Cheah, "An iterative learning-control scheme for impedance control of robotic manipulators," *The International Journal of Robotics Research*, vol. 17, no. 10, pp. 1091–1104, 1998.

[9] T. Tsuji and Y. Tanaka, "On-line learning of robot arm impedance using neural networks," *Robotics and Autonomous Systems*, vol. 52, no. 4, pp. 257–271, 2005.
[10] S. Jung and T. C. Hsia, "On neural network application to robust impedance control of robot manipulators," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 1995, pp. 869–874.
[11] D. Katic and M. Vukobratovic, "Learning impedance control of manipulation robots by feedforward connectionist structures," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE, 1994, pp. 45–50.
[12] M. Cohen and T. Flash, "Learning impedance parameters for robot control using an associative search network," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 382–390, 1991.
[13] K. Ogata, *System dynamics / Katsuhiko Ogata.*, 4th ed. Upper Saddle River, NJ: Pearson/Prentice Hall, 2004.
[14] J. E. Colgate and N. Hogan, "Robust Control of Dynamically Interacting Systems," *International Journal of Control*, vol. 48, no. 1, pp. 65–88, 1988.
[15] N. Hogan and S. Buerger, "Impedance and Interaction Control," in *Robotics and Automation Handbook*. Berlin, Heidelberg: CRC Press, oct 2005, ch. 19, pp. 1–24. [Online]. Available: http://www.crcnetbase.com/doi/abs/10.1201/9781420039733.ch19
[16] J. E. Slightam, D. R. McArthur, S. J. Spencer, and S. P. Buerger, "Passivity analysis of quadrotor aircraft for physical interactions," in *2021 Aerial Robotic Systems Physically Interacting with the Environment (AIRPHARO)*, 2021.
[17] C.-W. Park, J. Lee, M. Park, and M. Park, "Fuzzy model based environmental stiffness identification in stable force control of a robot manipulator," in *Modeling Decisions for Artificial Intelligence*, V. Torra, Y. Narukawa, and S. Miyamoto, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 240–251.
[18] N. Diolaiti, C. Melchiorri, and Stramigioli, "Contact Impedance Estimation for Robotic Systems," *International Journal of Control*, vol. 21, no. 5, pp. 925–935, 2005.
[19] C. Wang, G. Yang, C.-Y. Chen, and Q. Xin, "Impedance control system analysis and gain tuning of robot joints with flexibility," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2018, pp. 2159–2164.
[20] K. Babarahmati, C. Tisco, J. Smith, H.-C. Lin, M. Erden, and M. Mistry, "Fractal Impedance for Passive Controllers: a Framework for Interaction Robotics," *Nonlinear Dynamics*, vol. 110, pp. 2518–2533, 2022.
[21] F. Ficuciello, A. Romano, L. Villani, and B. Siciliano, "Cartesian impedance control of redundant manipulators for human-robot co-manipulation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2120–2125.

APPENDIX

TABLE II
FITNET RESULTS

Layers	*Neurons*	*% Stable*	*MSE*	*Run Time [ms]*
1	1	94.62%	1.82E+02	6.37
1	10	90.05%	2.83E+02	6.41
1	100	88.59%	7.44E+01	6.51
1	500	89.54%	2.56E+01	6.56
1	1000	88.20%	6.74E+02	6.64
2	1	94.74%	1.69E+02	6.97
2	10	88.89%	4.20E+02	7.06
2	100	88.98%	3.23E+02	7.41
2	500	88.39%	3.63E+02	13.87
2	1000	87.05%	1.14E+03	34.30
4	1	94.66%	4.26E+02	8.01
4	10	90.62%	1.17E+01	8.12
4	100	90.67%	1.32E+02	8.68
4	500	88.13%	9.44E+02	28.13
4	1000	88.25%	6.55E+00	73.67
8	1	94.29%	3.32E+01	10.14
8	10	92.13%	7.31E+02	10.28
8	100	89.45%	4.82E+01	11.39
8	500	89.77%	3.54E+01	52.54
8	1000	88.82%	5.16E+01	172.31
12	1	94.87%	1.58E+02	12.46
12	10	91.21%	9.23E+02	12.60
12	100	90.33%	1.29E+02	14.31
12	500	89.42%	4.45E+02	82.10
12	1000	89.20%	1.66E+03	298.95

TABLE III
DL NETWORK RESULTS

Layers	*Neurons*	*% Stable*	*MSE*	*Run Time [ms]*
1	1	96.15%	6.74E+04	3.52
1	10	97.15%	3.13E+04	3.07
1	100	90.76%	1.66E+06	3.16
1	500	89.76%	1.22E+06	3.18
1	1000	86.19%	2.46E+06	3.45
2	1	96.00%	7.92E+04	3.02
2	10	97.71%	3.17E+03	2.99
2	100	89.20%	1.83E+06	3.16
2	500	84.58%	4.32E+05	3.75
2	1000	92.23%	3.54E+05	3.10
4	1	41.80%	4.94E+07	3.24
4	10	95.93%	1.95E+05	3.17
4	100	97.11%	1.74E+06	3.09
4	500	95.05%	3.16E+05	5.20
4	1000	94.81%	9.23E+05	3.66
8	1	41.92%	4.94E+07	3.06
8	10	93.72%	2.99E+05	3.19
8	100	96.54%	7.53E+05	3.00
8	500	95.31%	1.55E+06	4.13
8	1000	88.58%	3.16E+06	3.76
12	1	41.73%	4.94E+07	3.23
12	10	96.73%	3.77E+05	3.21
12	100	93.07%	6.51E+04	3.16
12	500	88.46%	1.33E+05	4.37
12	1000	93.06%	2.98E+05	4.42