Using Denoising Autoencoders to Predict Behavior of an Inverted Pendulum on a Cart System

J. Khalid[1], A. Nasir[2], U. T. Shami[3], A. Baig[4]

[1,2,4]Electrical Engineering Department, University of Central Punjab, Lahore
[3]Electrical Engineering Department, University of Engineering and Technology, Lahore
[2]a.nasir@ucp.edu.pk

Abstract-This paper presents a method for precise prediction of the behavior of an inverted pendulum on a cart system. We have improved the accuracy of prediction beyond what can be achieved through traditional model-based simulation. This improvement has been achieved through learning of the differences between simulation and experimental results. Specifically, a three layered neural network known in the literature as denoising autoencoder has been used for learning. The proposed method consists of three steps. First step is to design linear controller for the inverted pendulum using text book methods and perform simulations. Second step is to perform experiments on the actual hardware of the inverted pendulum in the laboratory using the same controller as in first step. Third step is to learn the difference between simulation results and the results from the experiments using neural networks. Now the learned neural network is used to predict lab experiment results based on simulations with different initial conditions and reference values than the ones used to train the network. We have designed Linear Quadratic Regulator for demonstration of the proposed method. Results from the autoencoder have been reported. It is found that the autoencoder can predict the actual behavior of the pendulum with reasonable accuracy.

Keywords-Denoising Autoencoder, LQR Controller, Controller Implementation

I. INTRODUCTION

TRADITIONALway of implementing a controller for a dynamical system involves four steps. First the controller gains are computed based on the mathematical model. Second, the designed controller is simulated to predict the performance of actual hardware. If the results in the simulations are satisfactory, then in the third step, the designed controller is implemented on the hardware and the performance is observed. This performance usually differs from the simulation results thus leading to the fourth step which is to "adjust" the controller gains to achieve the performance depicted by the simulations. Adjusting of the gains is an iterative and sometimes costly process because it involves testing of the actual hardware. Sometimes much iteration is required before satisfactory performance is achieved.

Goal of this paper is to remove the need of hardware in the iterative process of the fourth step of controller implementation. Main idea is that if we can somehow predict how the hardware shall behave without having to perform experiments, then we can achieve our goal. In this regard, we have proposed a neural network based approach to learn the behavior of actual hardware through learning the difference between hardware and simulation based results. With our proposed approach, the controller implementation involves the following. First three steps of controller implementation shall remain the same as explained before. After the third step, the simulation results and the results of the testing of hardware are used to train the neural network. Once the neural network is trained, it can predict the behavior of the hardware using the results of the simulations. This enables the "adjustment" of gains using the simulations and the trained neural network only.

We have used inverted pendulum on a cart as a bench mark for demonstrating our approach. A number of researchers have demonstrated various controller designs and sensing applications using this benchmark. One of the most popular controller designs implemented on the inverted pendulum is back stepping based nonlinear controller design [i, ii, iii, iv]. Another popular approach is neural network based controller design [v, vi, vii, viii]. Sliding mode controller design approach has also been explored e.g. [ix]. A comparison between sliding mode and periodic control law has been presented in [x]. Computer vision based control has also been designed [xi]. Influence of parameters on the control on inverted pendulum is presented in [xii]. MATLAB simulation of the inverted pendulum control has been discussed in [xiii]. There have been some review papers on the inverted pendulum control as well [xiv, xv]. Fuzzy logic based controller design has been discussed in [xvi]. Linear controllers such as PID and LQR for inverted pendulum have been described in [xvii, xviii, xix]. A

commercial form of inverted pendulum on a cart is mobile inverted pendulum [xx, xxi, xxii]. Literature is available on the approaches for the design of controller of this application [xxiii, xxiv].

Denoising autoencoders [xxv] on the other hand, arise from the artificial intelligence community. Specifically, the concept of deep learning [xxvi] has motivated the invention of these autoencoders. Autoencoders are basically layers of affine sigmoid encoding followed by affine decoding to recover the output from an input signal. In this paper, the input is the simulation results including variations in angle and angular rate of the pendulum as well as cart velocity andposition. Output of the autoencoders in our case is the expected experimental results corresponding to all the input signals. In order to be able to recover reasonable output, denoising autoencoders have to be trained using the training dataset. Training involves learning of the weights corresponding to neurons in the autoencoders. Larger the training set better is the expected performance of the autoencoders. But the size of the training set is limited by the cost of performing experiments to collect the training data. This is a limiting factor in the application of autoencoders. Denoising autoencoders have found many applications. One of the common applications is speech enhancement [xxvii, xxviii, xxix, xxx]. Another application is emotion recognition [xxxi]. Denoising autoencoders have also been used for feature learning from EEG[xxxii]. Another interesting application is in fault detection and diagnosis [xxxiii].

In this paper, multiple LQR controller designs have been simulated and tested on the inverted pendulum hardware. The comparison of the theoretical and practical results has been presented. The gaps in the theoretical and practical results have been discussed. A denoising autoencoder based approach has been used to depict the actual behavior of the inverted pendulum based on the simulation results. The depiction of the denoising autoencoders has been analyzed and is found to be pretty close to the actual behavior. Hence, the gap between the theoretical and practical results has been learnt by using samples of practical behavior as seeding. This type of learning enables the controller designer to tweak his design without having to repeat the experimentation. For example, we design a couple of controller gains and test on real hardware. Then the result is used to learn the gap between actual and simulation results. From thereon, no matter how many controller gains are designed, the actual behavior can be depicted without actually performing the experiment on the hardware.

In section II of the paper, the experimental setup and mathematical model of the inverted pendulum is described. Design of LQR controller is discussed in section III. Section IV presents the learning methodology and introduction to denoising autoencoders. Section V includes simulation setup and results along with experimental results and results from using denoising autoencoders. Section VI concludes the paper.

II. Mathematical Model of Inverted Pendulum on A Cart

As discussed in the introduction section, there is a huge literature on the control and dynamics of the inverted pendulum on a cart system. Now a day, this system is part of almost every control systems laboratory at technical institutes and engineering universities. The equations presented in this section can be found in standard textbooks and laboratory manuals. The inverted pendulum used for the work in this paper is shown in Fig. 1. The computer screen in the background shows the graphical user interface for the apparatus that is developed in SIMULINK/MATLAB. On the left side of the CPU in Fig. 1 is the casing that encloses digital signal processing card for the apparatus. This card is interfaced with the CPU through peripheral component interconnect slot.

Fig. 2 shows the force analysis of cart and rod system. *N* and *P* denote interactive force of cart and rod in horizontal and vertical direction respectively.

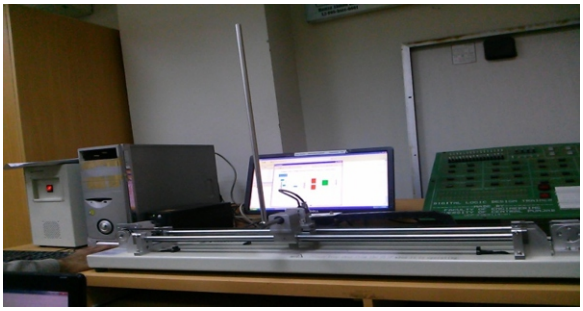

Fig. 1. Inverted Pendulum Hardware

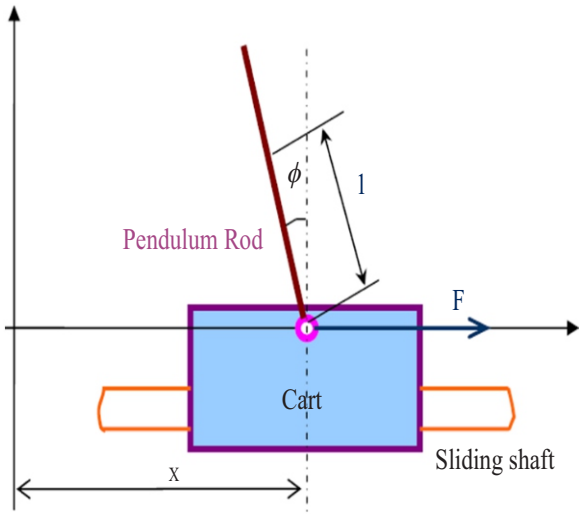

Fig. 2. Visualization of variables in the system

From forces in horizontal direction (for the cart), the equation below can be obtained

$$M\ddot{x} = F - f\dot{x} - N \quad (1)$$

where,
F = Force acting on cart
x = Cart position
f = damping coefficient
N = Force exerted on the cart in horizontal direction due to motion of the pendulum
M = mass of the cart

From the force acting on the rod in horizontal direction we get

$$N = m\frac{d^2}{dt^2}(x - lsin\phi) \quad (2)$$

l = length of the pendulum rod
ϕ = Angle between the rod and vertically upward direction.
m = mass of the pendulum rod
Equation (2) can be written as,

$$N = m\ddot{x} - ml\ddot{\phi}cos\theta + ml\dot{\phi}^2 sin\phi \quad (3)$$

Substituting equation (3) into equation (1), first equation for non-linear system obtained is

$$(M+m)\ddot{x} + f\dot{x} - ml\ddot{\phi}cos\phi + ml\dot{\phi}^2 sin\phi = F \quad (4)$$

Similarly, combining the forces acting on the rod in vertical direction, we obtain second equation of motion

$$(I + ml^2)\ddot{\phi} - mglsin\phi = ml\ddot{x}cos\phi \quad (5)$$

Linearization of equations (4) and (5) about the equilibrium point $[x \quad \dot{x} \quad \phi \quad \dot{\phi}]^T = [0 \quad 0 \quad 0 \quad 0]^T$ results in the following state space equation in matrix form.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix} u \quad (6a)$$

where u=F and the output equations can be written as,

$$y = \begin{bmatrix} x \\ \phi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u \quad (6b)$$

III. LQR Controller Design

LQR controller is a linear feedback controller in which system is defined using set of linear equations and cost is defined by a quadratic function. Purpose of the controller is to stabilize system at minimum possible cost.

Considering a system

$$\dot{x} = Ax + Bu \quad (7)$$

Minimum cost function of the controller is calculated by approximating Q and R values.

$$J = \int_0^\infty (x^t Qx + u^t Ru)dt \quad (8)$$

where

Where, A and B are state dynamics and input matrices respectively. Q is the cost of the deviation of the state variable values from the equilibrium point. R is the control effort cost. It consists of either unit value or symmetric matrix. Matrix Q and R determine relative importance of error and energy loss respectively. Q and R values are estimated as per expected performance criterion

Optimal control vector is given by matrix 'K' such that performance index can be minimized

$$u(t) = Kx(t)$$
$$K = -R^{-1}B^TP \quad (9)$$

To get the value of P, algebraic Riccati equation below is solved (assuming infinite horizon problem)

$$PA + A^TP + Q - PBR^{-1}B^TP = 0 \quad (10)$$

K matrix can also be obtained by using following MATLAB command

$$K = lqr(A, B, Q, R) \quad (11)$$

By Changing Q, different system responses are obtained. The system will behave more robust to disturbance and the settling time will be shorter if Q is higher.

IV. Learning with Denoising Autoencoder

It is quite clear from the above discussion that simulation results cannot authentically represent the practical real world systems. That being said, simulations still hold an important place in the whole design and development process as they provide a mathematical basis for the whole system to be designed. Therefore, if we are somehow able to design a model or a simulation that can very accurately represent the practical system we would have the best of both worlds. The fundamental reason for such a huge discrepancy between simulation and real world results is due to the inherent nonlinearities in the real world

systems that can never be modeled rigorously and thoroughly by a simulation. This means that it would be quite difficult to design a model that can simulate the results closer to the real world values.

An alternate approach to solving the problem of this discrepancy between simulated and real world results would be to find patterns in these discrepancies and adjust the simulation results accordingly. If we are able to do so this would allow us to provide simulation results which are quite close to the real world results and thus make these simulations very effective.

It should be remembered that since the mathematical models used in simulations are designed based on the real world systems they both have the same basic structure. They should ideally provide similar results. The results vary due to the difference in the model and real world that are nonlinear in nature. In order to improve the accuracy of the simulations we need to capture this nonlinearity and compensate for it.

Neural Networks have long being used as a system to detecting and modeling nonlinear class boundaries. They have been able to learn nonlinear boundary between classes based on a set of labeled data. This ability to learn nonlinear boundaries make them quite an interesting and effective tool.

Recent advancements in neural network research, especially deep learning, have introduced a new kind of neural network called Autoencoder. An Autoencoder is a neural network with a single hidden layer and where the output layer and the input layer have the same size. The weights of autoencoder can be trained with gradient decent algorithm also known as back propagation algorithm [xxxiv]. Denoising autoencoder is straight forward variant of the basic autoencoder. Denoising autoencoder is trained to reconstruct a clean input from its corrupted version. They are created by establishing a hidden layer which is larger in size then the input and output layers (both are of same size). The arrangement makes the whole system sparse which in turn allows us to detect and remove noise present in the input data. Denoising autoencoder does two things i.e. it tries to encode the input as well as undo the effect of corruption process stochastically applied to the input of autoencoder. Hence denoising autoencoder is trying to predict the corrupted value and refine it, to remove distortion. Denoising autoencoders are commonly used in deep learning applications to develop a more generalized model of the input data. Fig. 3 shows the overall structure of a denoising autoencoder whereas Fig. 4 shows how neurons in each of the layers are connected. Specifically, we have shown three layers i.e. input layer (with neurons I_1, I_2), hidden layer (with neurons H_1, H_2, and H_3), and output layer (with neurons O_1, O_2). In general number of layers and number of neurons per layer may vary from case to case.

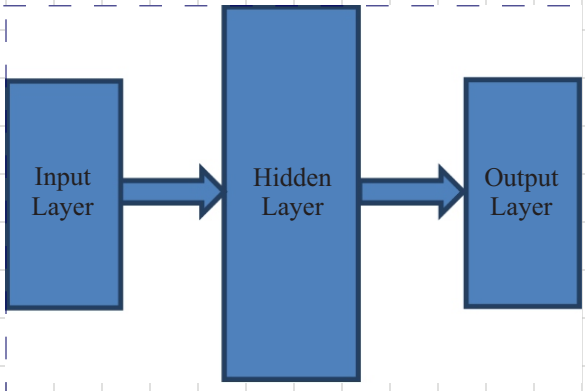

Fig. 3. Overall structure of a Denoising Autoencoder

The input to the autoencoder is mapped in terms of a sigmoid function

$$f_{W,b}(x) = s(Wx + b), \qquad s(y) = \frac{1}{1 + e^{-y}} \quad (12)$$

Where W is a weighting matrix and b is an offset vector. The output is reconstructed as

$$z(x) = W'x + b' \quad (13)$$

Where W' and b' are appropriately sized parameters. Backpropagation is used to modify the values of connected weights to achieve desired level of distortion removal through successive iteration. So it will result in an error signal, if we keep on comparing the resulted outputs of network with the target values. Then the error signal is fed back in the network and the values of layer's weights are reset accordingly. The most commonly used method to find value of error signal(s) is the mean square error function (MSE) represented hereunder

$$E = \frac{1}{n}\sum_{p=1}^{n}\sum_{i=1}^{m}(y_{pi} - t_{pi})^2 \quad (14)$$

Where n is number of training samples, m is output vector dimension, y_{pi} is network output of i^{th} neuron for pattern p, and t_{pi} is target value of i^{th} component for pattern p.

Back propagation produces a gradient descent procedure to minimize the value of error signal E as follows,

$$\Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}} \quad (15)$$

Where W_{ij} indicates connection weight between neurons i and j. η represents learning rate.

The above mention two stages perform their functions alternatively till target output of the network can be approximated with the smallest error.

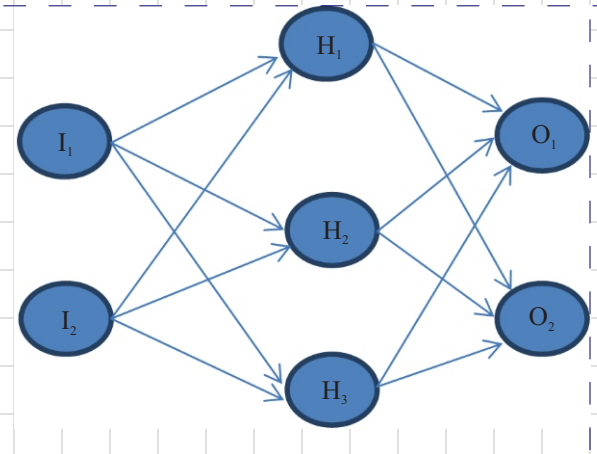

Fig. 4. Internal Connections within a Denoising Autoencoder (2-3-2 example)

It is this ability of the autoencoders to detect and model the noise in the input signal that is being exploited here. The basic idea is that if the denoising autoencoder can detect and model the discrepancy between the simulated and actual results we use them to transform the simulated results and bring them closer to the actual results.

V. Simulations and Experimental Results

A. *Simulations of proposed LQR controller model*

Fig. 5 presents the proposed LQR controller model developed in MATLAB GUI environment. Initially, cart and rod's reference positions are used and compared with cart and rod's actual positions. Swing up controller swings the rod from downward position to upright position using the energy efficient variable.

In addition, the rod is kept in the upright position by correct calculation of instantaneous rod angle and cart position.

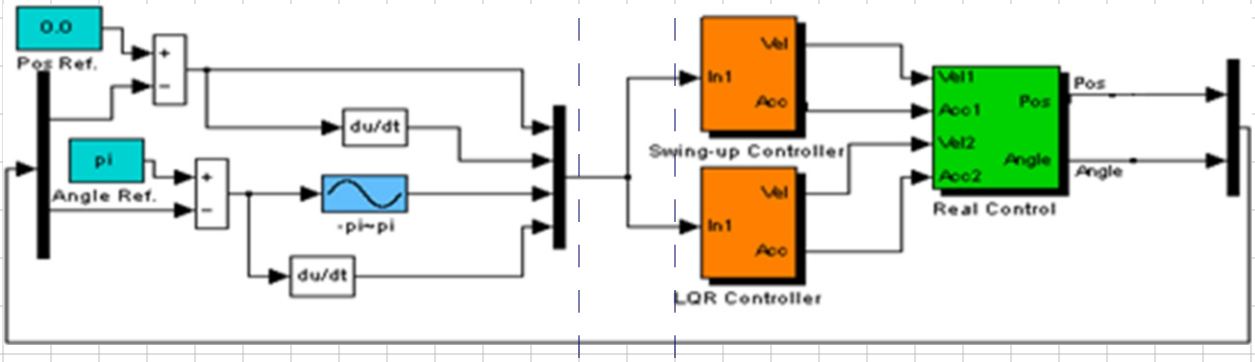

Fig. 5. Snapshot of the part of SIMULINK Model

Inputs are fed into LQR controller to control pendulum angle and cart position. Since it is a feedback system, instantaneous position of cart and rod angle is fed into input of the system which is compared with reference position and angle. Pendulum angle φ should be stabilized at 0º and cart position should drift to stabilize the pendulum system.

B. *Hardware Results*

Table I shows the parameter values for the hardware used. These values are for the pendulum and cart system shown in Fig. 1. Table II shows the controller gains and Q/R ratios for the designed controllers. The gain values have been computed using equations (9) and (10). Q/R ratios have been selected carefully in order to obtain results for a fairly broad range of values. Note that Q and R are not of same size therefore here by Q/R we mean the ratio of magnitudes of Q and R. By changing the gain, different responses are observed which are presented later in the section. Q is selected as a diagonal matrix with equal weights along the diagonal. Therefore the value of the weight in Q is taken as its magnitude. This choice may not be the best for all applications but since the objective here is educational, this simplified selection is not too constraining. Note that Q is a 4 x 4 matrix and R is a scalar for the state space linear model presented in (6a) and (6b)).

When gains 1, 2, and 3 are tested on MATLAB, the cart drifts towards a direction. Pendulum rod achieves 0° and maintains this angle throughout. However, when these values are tested on hardware structure, the responses appear to be different from MATLAB simulation results. Cart slides towards one end at a very high speed due to which pendulum angle does not stabilize.

TABLE I
PARAMETER VALUES FOR THE INVERTED PENDULUM HARDWARE

Symbol	Description	Actual value in hardware used
M	Cart mass	1.096 kg
m	Rod mass	0.109 kg
$b = f$	Friction coefficient of cart	0.1 N/m/sec
L	Rod distance from rotation axis center to the rod mass center	0.25 m
I	Rod Inertia	0.00223 kgm^2

TABLE II
CONTROLLER GAINS AND CORRESPONDING Q/R RATIO

Sr. Number	Kx	$K\dot{x}$	Ka	$K\dot{a}$	Q/R
Gain 1	-0.3162	-3.4201	24.833	4.7469	0.1
Gain 2	-1	-4.0709	26.5105	5.185	1
Gain 3	-3.1623	-6.4282	33.8787	7.3334	10
Gain 4	-10	-14.7898	64.9447	16.43	100
Gain 5	-22.36	-30.40	125.68	33.98	500
Gain 6	-31.82	-42.16	171.74	47.24	1000

When gain 4 is used as input parameters, MATLAB simulation results show cart drift slightly to attain stability and pendulum rod gets stabilized in 2 seconds. When these parameters are tested on hardware structure, cart drifts and cover some distance to stabilize the rod in upright position. Pendulum rod stabilizes and slight jitters are observed in motor once stabilizing the rod. The responses of gain 4 tested on both MATLAB and hardware is shown in Fig. 6.

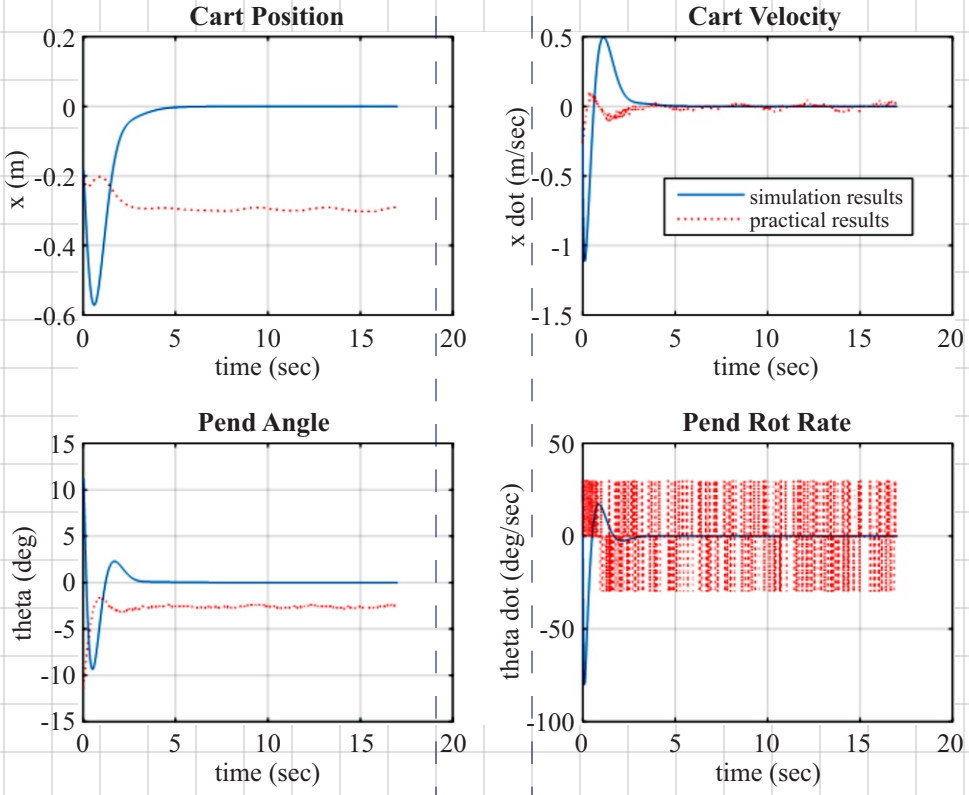

Fig. 6. Gain 4 responses tested on MATLAB and hardware

When gain 5 is used as input parameters, MATLAB simulations and hardware results are shown in Fig. 7. Pendulum rod stabilizes at 3.14 radians. Notice the oscillations observed in pendulum rod angular velocity.

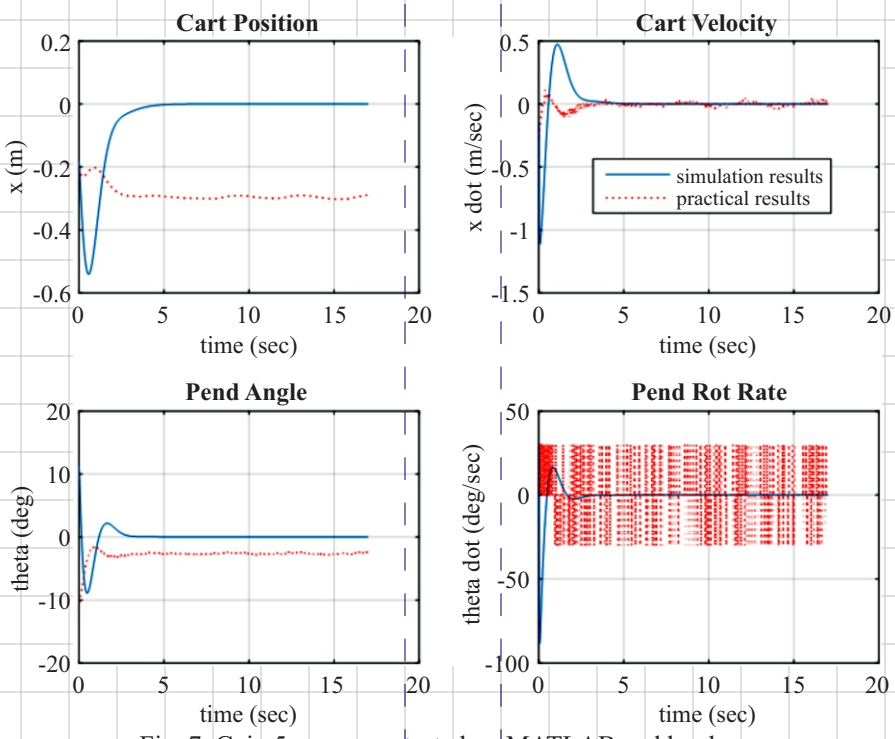

Fig. 7. Gain 5 responses tested on MATLAB and hardware

When gain 6 is used as LQR parameters, the responses of MATLAB simulation and hardware are shown in Fig. 8. When these values are tested on hardware structure, swing up controller swings the rod by drifting cart and move the rod to upright position and rod gets stabilized.

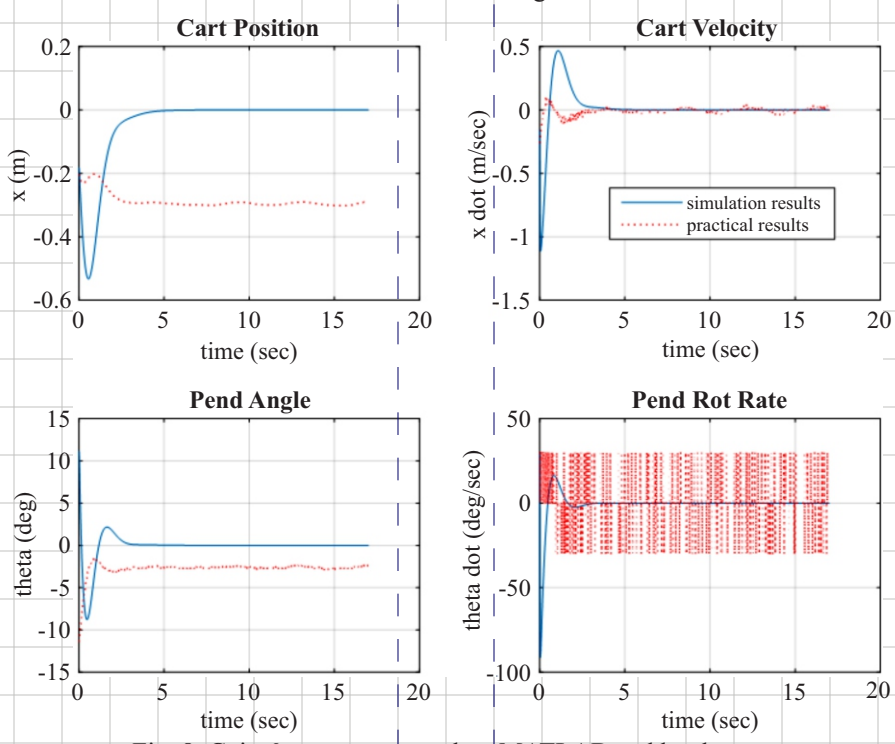

Fig. 8. Gain 6 responses tested on MATLAB and hardware

Based on the results obtained, there are many issues to be discussed related to the differences that are evident in the plots above.

I. It may be noticed that there is a 3 degrees steady state error in the pendulum angle in all three cases. The reason for this error is not easy to find since there are more than one possibility. First, the pendulum hardware may not be exactly horizontal, second, there may be a bias error in the encoder, third, there can be a steady state error due to unknown dynamics of the system, fourth, the signal processing performed on the encoder reading or overall coding involved in the project may have some logical error.

ii. Another major difference between simulation and experimental results evident in the plots is in the rate of change of pendulum angle. This may be problematic when the practical application requires the pendulum to be steady after some time e.g. in Segway.

iii. The steady state of the cart position in practical and simulation results also do not match. This may be solved by iterating on the Q matrix selection. But the main issue here is that with the current selection of Q, simulations have shown good results for the cart position. How would one know if the designed controller is going to work well on the real hardware or not?

In general, the results indicate that the simulations are almost of no use for precise practical applications in case of an inverted pendulum. This finding places a question mark on the authenticity of the simulation based research work and on the simulation based design methodology i.e. design a controller using mathematical model, perform simulations, and apply the controller on real hardware if the simulation results are satisfactory. There is a remarkable difference in practical and simulation based results; clearly, the methodology described is not in general going to work right away (there may be exception of course). This poses a huge problem for systems such as satellites and maybe rocket launchers where obtaining practical results and iterating on them is very costly. Hence there is a need for rigorous research in this area and to devise a method to minimize the differences and gaps between simulations and hardware implementation. The above discussion prompts a need to check whether feedback control properties such as robustness and disturbance rejection hold with simulation based designs, and if yes, to what extent?

C. *Results with denoising autoencoders*

A denoising autoencoder has been developed to test this hypothesis with 4 neurons each in input and output layers and 16 neurons in the hidden layer. One set of simulated and actual results on the designed controller have been used for training the autoencoder. Two set of simulated results have been used for testing the autoencoder with the actual results as the ground truth for the experimentation. During training the simulated results have been used as input and the actual results have been used as the output that the autoencoder is supposed to achieve.

Once the autoencoder has been trained the results from the other two simulations are provided to the neural network and the output was recorded and compared to the actual results. Fig. 9 and Fig. 10 show the simulated, actual and the adjusted results. It is quite clear from the graphs that the prediction from the autoencoder is much closer to the actual results than the simulation results.

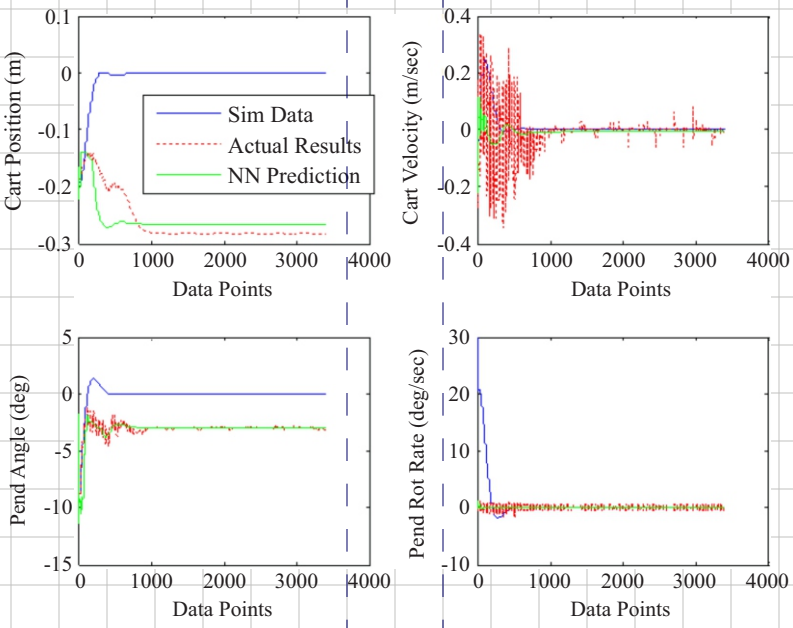

Fig. 9. Training Results from Gain 5

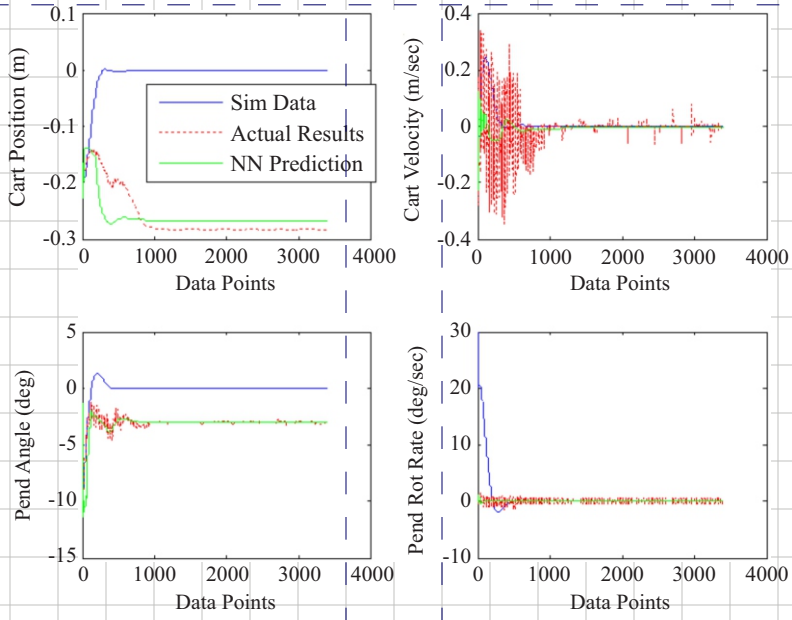

Fig. 10. Training Results from Gain 6

Table III presents quantitative results from the autoencoder predictions. Second column of the table presents the mean value of the difference in state variable values from simulation results and from experimental results. Third column of the table presents the mean value of the difference in state variable values from neural network based predictions and from experimental results. Fourth column of the table presents the value of the variance of the difference in state variable values from simulation results and from experimental results. Fifth column of the table presents the value of the variance of the difference in state variable values from neural network based predictions and from experimental results. Note that the mean and variance of almost all the errors is orders of magnitude smaller in case of Neural Network predictions.

TABLE III
QUANTITATIVE RESULTS FROM AUTOENCODER PREDICTION

State	Mean error between simulations and actual results	Mean error between NN prediction and actual results	Variance of error between simulations and actual results	Variance of error between NN prediction and actual results
x (m)	-0.2528	-0.0026	0.0044	0.0009
$\dot{x}$ *(m/sec)*	-0.0168	0.0021	0.0057	0.0040
ϕ *(deg)*	-2.9491	0.0472	0.3920	0.5385
$\dot{\phi}$ *(deg/sec)*	-0.6491	0.0074	11.5442	0.0728

VI. CONCLUSIONS

Prediction of the behavior (dynamics) of actual hardware of an inverted pendulum has been discussed. It has been demonstrated through simulations and experimental results that the proposed denoising autoencoders based method for prediction is more accurate than traditional simulation method.

The additional accuracy comes at the cost of having to perform some experiments on the hardware (in order to generate the training data set). Whereas in traditional simulations, the behavior of the hardware is predicted without performing any experiment on the actual hardware.

Furthermore, the accuracy of the prediction is proportional to the size of the training data set. But for locally linear systems such as inverted pendulum, a fairly small training set may be enough for sufficient accuracy in the prediction. Suchis the case in this paper where we have trained the network on one set of gains (gain 4 in Table II) and it is able to predict the results of gain 5 and gain 6 with reasonable accuracy. This means that there is no need to retrain the neural network in the event of controller redesign. This is an encouraging

result.

The approach presented in this paper can be extended in two ways. One way is to enhance the internal structure of the autoencoders and find out if the learning accuracy can be improved. Other learning methods may also be used such as reinforcement learning and its variants. Second approach to extend this work is to prove some analytical properties such as independence of the approach from the control law used. Another interesting property may be to find out the upper bound on the prediction error for general applications or specific ones.

REFERENCES

[I] Ö.T.Altınöz, 2007. Adaptive integral backstepping motion control for inverted pendulum. *eec*, *1*, p. 22.

[ii] S. Rudra and R.K. Barai, 2012. Robust adaptive backstepping control of inverted pendulum on cart system. *International journal of control and automation*, *5*(1).

[iii] F. K. Tsai and J. S. Lin, 2003. Backstepping control design of 360-degree inverted pendulum systems.

[iv] V. K. Singh, V. Kumar, 2014. Nonlinear Design for Inverted Pendulum using Back stepping Control Technique. *International Journal of Scientific Research Engineering & Technology (IJSRET)*, Vol. 2 Issue 11, (2014): 807-810

[v] C. W. Anderson, 1989. Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, *9*(3), pp.31-37.

[vi] S. Jung and S. S. Kim, 2008. Control experiment of a wheel-driven mobile inverted pendulum using neural network. *IEEE Transactions on Control Systems Technology*, *16*(2), pp. 297-303.

[vii] Z. Li and C. Yang, 2012. Neural-adaptive output feedback control of a class of transportation vehicles based on wheeled inverted pendulum models. *IEEE Transactions on Control Systems Technology*, *20*(6), pp.1583-1591.

[viii] C. Yang, Z. Li, R. Cui, and B. Xu, 2014. Neural network-based motion control of an under actuated wheeled inverted pendulum model. *IEEE transactions on neural networks and learning systems*, *25*(11), pp.2004-2016.

[ix] R. J. Wai and L. J. Chang, 2006. Adaptive stabilizing and tracking control for a nonlinear inverted-pendulum system via sliding-mode technique. *IEEE Transactions on Industrial Electronics*, *53*(2), pp.674-692.

[x] A. Chakraborty and J. Dey, 2015, March. Performance comparison between sliding mode control and periodic controller for cart-inverted pendulum system. In *Industrial Technology (ICIT), 2015 IEEE International Conference on* (pp. 405-410). IEEE.

[xi] H. Wang, A. Chamroo, C. Vasseur, and V. Koncar, 2008, June. Stabilization of a 2-DOF inverted pendulum by a low cost visual feedback. In *American Control Conference, 2008* (pp. 3851-3856). IEEE.

[xii] J. Jie and T. Wei, 2012, May. Influence analysis of initial state parameters on linear inverted pendulum system performance. In *Control and Decision Conference (CCDC), 2012 24th Chinese* (pp. 3498-3501). IEEE.

[xiii] K. Mahmud, 2013. Design and Analysis of Control of an Inverted Pendulum System by MATLAB. *Global High Tech Congress on Electronics (GHTCE),* IEEE (2013): 207-211.

[xiv] K. Perev, 2011. Inverted Pendulum Control; an Overview.*Information Technologies and Control*, (2011) 34-41

[xv] V. Kurdekar, and B. S.Borkar, 2013. "Inverted Pendulum Control: A Brief Overview." *International Journal of Modern Engineering Research (IJMER)* 3.5, 2924- 2927.

[xvi] T. Yamakawa, 1989. Stabilization of an inverted pendulum by a high-speed fuzzy logic controller hardware system. *Fuzzy sets and Systems*, *32*(2), pp.161-180.

[xvii] W. Li, H. Ding, and K. Cheng, 2012, September. An investigation on the design and performance assessment of double-PID and LQR controllers for the inverted pendulum. In *Control (CONTROL), 2012 UKACC International Conference on* (pp. 190-196). IEEE.

[xviii] L. B. Prasad, B. Tyagi, and H.O. Gupta, 2012, May. Modelling and simulation for optimal control of nonlinear inverted pendulum dynamical system using PID controller and LQR. In *Modelling Symposium (Ams), 2012 Sixth Asia* (pp. 138-143). IEEE.

[xix] H. Wang, A. Chamroo, C. Vasseur, and V. Koncar, 2008, June. Stabilization of a 2-DOF inverted pendulum by a low cost visual feedback. In *American Control Conference, 2008* (pp. 3851-3856). IEEE.

[xx] H. Leeand J. Lee, 2012, November. Driving control of mobile inverted pendulum. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2012 9th International Conference on* (pp. 449-453). IEEE.

[xxi] H. Lee and S. Jung, 2012. Balancing and navigation control of a mobile inverted pendulum robot using sensor fusion of low cost sensors. *Mechatronics*, *22*(1), pp.95-105.

[xxii] J. Lee and J. M. Lee, 2013. A study on the visual servoing of autonomous mobile inverted pendulum. *Journal of Institute of Control, Robotics and Systems*, *19*(3), pp.240-247.

[xxiii] D. Choi and J. H. Oh, 2008, May. Human-friendly motion control of a wheeled inverted

pendulum by reduced-order disturbance observer. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* (pp. 2521-2526). IEEE.

[xxiv] V. Muralidharan and A. D. Mahindrakar, 2014. Position stabilization and waypoint tracking control of mobile inverted pendulum robot. *IEEE Transactions on Control Systems Technology*, *22*(6), pp. 2360-2367.

[xxv] G. Zhou, K. Sohn, and H. Lee, 2012. Online incremental feature learning with denoising autoencoders. *Ann Arbor*, *1001*, p.48109.

[xxvi] L. Deng, 2011, October. An overview of deep-structured learning for information processing. In *Proceedings of Asian-Pacific Signal & Information Processing Annual Summit and Conference (APSIPA-ASC)*.

[xxvii] X. Feng,Y. Zhang, and J. Glass, 2014, May. Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on* (pp. 1759-1763). IEEE.

[xxviii]D. Liu, P. Smaragdis, and M. Kim, 2014, September. Experiments on deep learning for speech denoising. In *INTERSPEECH* (pp. 2685-2689).

[xxix] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, 2013, August. Speech enhancement based on deep denoising autoencoder. In *Interspeech* (pp. 436-440).

[xxx] B. Xia and C. Bao, 2013. Speech enhancement with weighted denoising auto-encoder. In *INTERSPEECH* (pp. 3444-3448).

[xxxi] R. Xia, J. Deng, B. Schuller, and Y. Liu, 2014, May. Modeling gender information for emotion recognition using denoising autoencoder. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on* (pp. 990-994). IEEE.

[xxxii]J. Li, Z. Struzik, L. Zhang, and A. Cichocki, 2015. Feature learning form incomplete EEG with denoising autoencoder.*Neurocomputing*, *165*, pp.23-31.

[xxxiii]T. Tagawa, Y. Tadokoro, and T. Yairi, 2014. Structured Denoising Autoencoder for Fault Detection and Analysis. In *ACML*.

[xxxiv]S. J. Russell and P. Norvig, 2002. Artificial intelligence: a modern approach (International Edition).