Application of Neural Networks for Control of Self Balancing Bicycle

1st Pushkar Hiray
Department of Avionics
Indian Institute of Space Science and Technology
Thiruvananthapuram, India
pushkarhiray@gmail.com

2nd Neelabh Dev
Department of Avionics
Indian Institute of Space Science and Technology
Thiruvananthapuram, India
neelabh.dev22@gmail.com

***Abstract*—In this paper we present a method to dynamically control an autonomous bicycle with a balancer using a neural network controller. The balancer is a type of an inverted pendulum that is attached to the main frame of the bicycle. We have generated our own data set using LQR controller and used it to train a feed forward neural network controller. Optimization techniques, number of hidden layer neurons, and number of epochs were varied and necessary results and conclusions were drawn. The neural network controller has a final training error of 10E-5. We also have analysed the performance of the controller with the variation in the input given to the system.**

***Index Terms*—Autonomous bicycle, balancer, neural network controller, LQR controller.**

I. Introduction

Research on the stabilization techniques for autonomous bicycles has become popular in this age where technology just wants to automatize everything. It is well known that the balancing control of bicycle with steering at zero or slow linear velocity is very difficult therefore, we have chosen to follow the method of stabilization using an inverted pendulum as used by the researcher of [1]. The researchers of [1] have used an LQR controller for the stabilization of their bicycle. But the major inconveniences with LQR controller is that we have to model the Q and R matrices every time we need to update the controller. If there are any changes in the plant dynamics which we need to account for, we need to update the LQR controller. Therefore, we propose a method of stabilizing using the neural network controller. The research on using a feed-forward neural network as a controller has been done by the researcher of [2]. The researchers of [2] but have only implemented on a simple inverted pendulum problem. In our case we propose the neural network controller for a bicycle having a balancer. Note that bicycle itself is an inverted pendulum, we are balancing it by attaching it another inverted pendulum (balancer) therefore making it a two link manipulator problem.

II. Modeling of the Bicycle

All the bicycle parameters necessary for our simulations were derived from the research paper [2] as we can see in the following table. A state space model of the plant dynamics was made. Four states are defined according to the multi-body dynamics. The four states are:-

Parameter descriptions	Par.	Value
Bicycle mass	m	45 kg
Height of the bicycle center of mass	h	0.5 m
Moment inertia of bicycle at COG	I	0.81 kgm^2
Distance between ground and balancer	l	0.87 m
Balancer mass	m_b	11.72 kg
Moment inertia of balancer at COG	I_b	0.49 kgm^2

Fig. 1. Bicycle parameter values

1) α : Roll angle of the bicycle
2) β : Balancer angle
3) $\dot{\alpha}$: Rate of change of alpha
4) $\dot{\beta}$: Rate of change of beta

These states can be visualized from the following diagram.

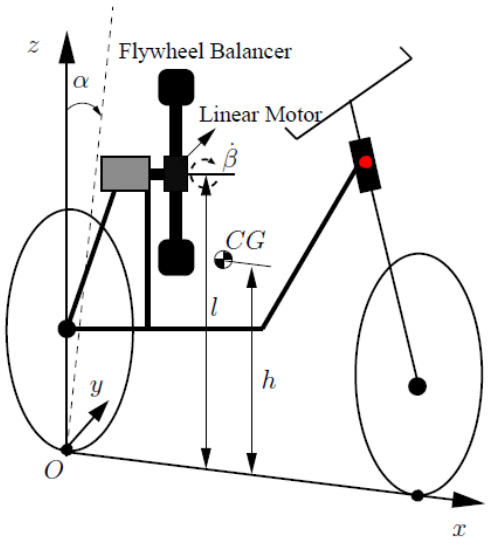

Fig. 2. Bicycle model with the balancer

The input to the plant dynamics is the torque given to the balancer, while the four states can be observed from the

plant dynamics. Therefore, the controller has to provide the necessary torque to the plant, and since it is closed loop control, the state information is fed as input to the controller.

III. LQR CONTROLLER AND GENERATION OF THE DATA SET

We have used LQR algorithm to make a controller to keep the bicycle in an upright position.

- The Linear Quadratic Regulator (LQR) is a method to optimally controlled feedback gains to enable the closed-loop stability.
- It is a methodology based on the calculation of variations.
- It has an infinite gain margin.

We have thought of LQR as a neural network without any hidden layers. 'x' is the 4×1 input and 'u' is 1×1 output.We find 'K' matrix (1×4) by minimizing the loss function.
In state space form this 'K' matrix can be incorporated as

$$\dot{x} = (A - BK)x + BKx_{desired}$$

$x_{desired}$ is the vector of desired states, here it acts like external inputs for the closed loop system.

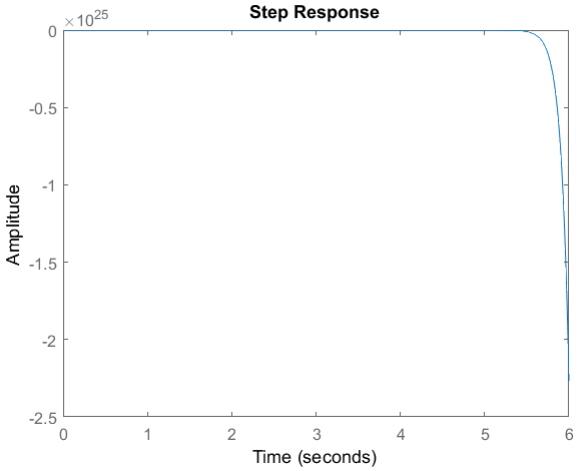

Fig. 3. Step response of uncompensated system

Once the LQR controller is incorporated in our system the instability in (figure 3) is removed. plotted the transient response of α and $\dot{\alpha}$ (figure 4) and β and $\dot{\beta}$ (figure 5). We plotted the output torque (figure 6) for the labels of our inputs We took samples of α, $\dot{\alpha}$, β, $\dot{\beta}$ and torque at 950 samples per second to generate our dataset along with the labels.

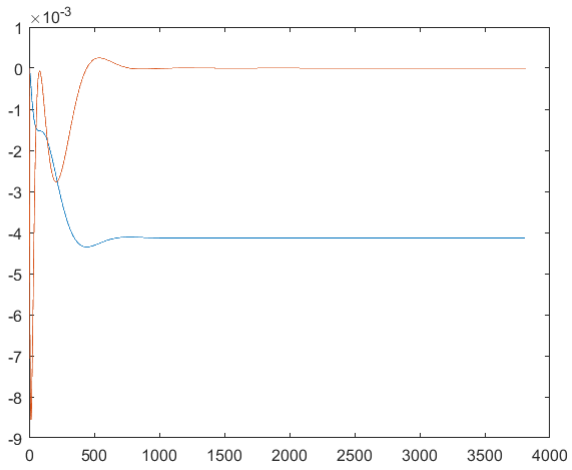

Fig. 4. α and $\dot{\alpha}$ vs time

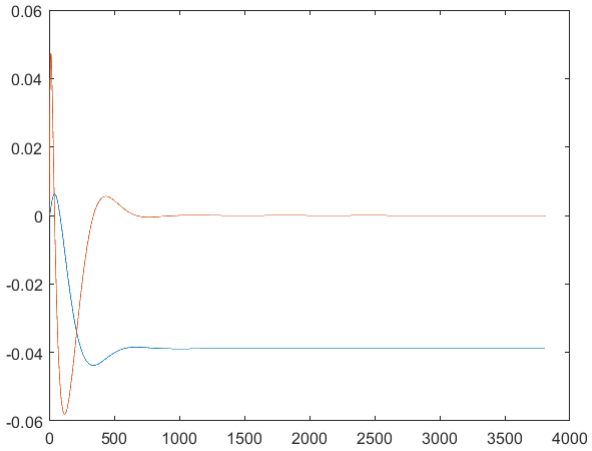

Fig. 5. β and $\dot{\beta}$ vs time

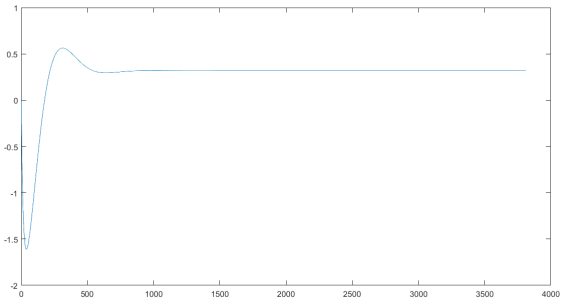

Fig. 6. Torque without neural network controller

IV. TRAINING AND TESTING OF THE NEURAL NETWORK

We have used the Matlab Neural network toolbox, for our simulations. We have a feed forward neural network which

has 4 input neurons and 1 output neuron. Unlike the standard controllers for state space where there are no hidden layers, we have used a one hidden layer containing 10 neurons. The initial learning rate is 0.05.We have used the Matlab Neural network toolbox, for our simulations. We have a feed forward neural network which has 4 input neurons and 1 output neuron. Unlike the standard controllers for state space where there are no hidden layers, we have used a one hidden layer containing 10 neurons. The initial learning rate is set to 0.05 . Since the data set positive as well as negative real values, we have used a pure linear activation function for the output and tan-sigmoid activation function as a non-linearity for the input to hidden layer. From the total of 3812 samples generated, we have used:

1) Random 3000 samples for training data set.
2) Remaining 812 samples for testing data set.

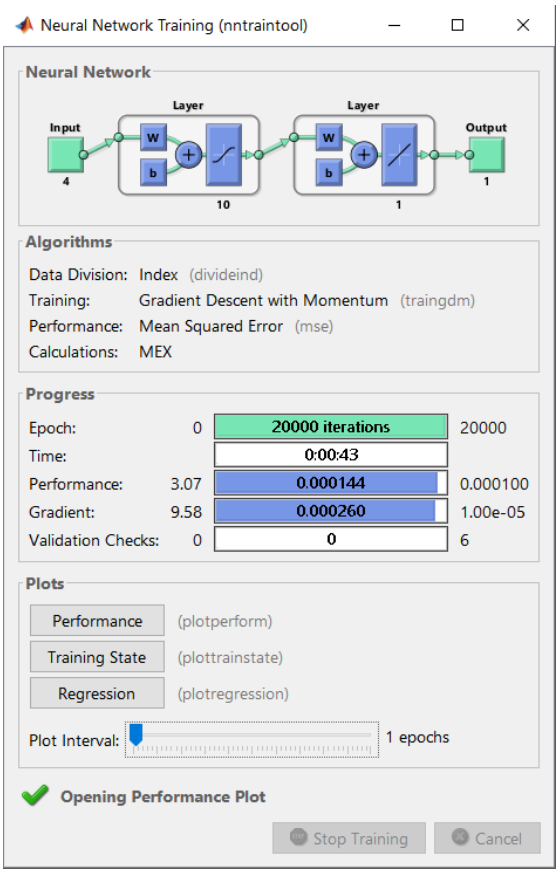

Fig. 7. The neural network representation in Matlab

V. Results and Analysis

A. For step input

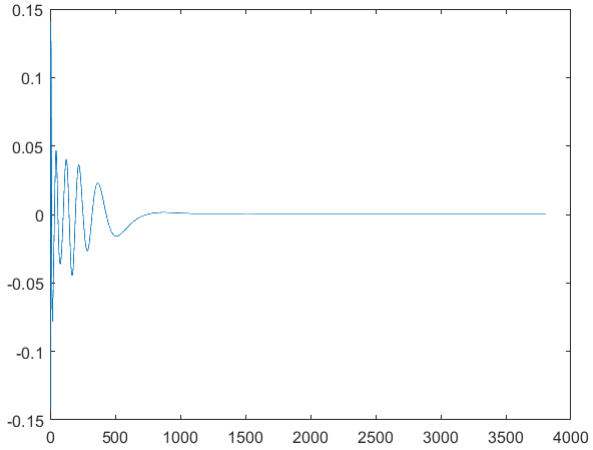

Fig. 8. Error vs Time for neural network controller

Figure 8 shows the error vs time response for the neural network controller. It can be seen that initially even though the error was quite high but gradually the error decreased and settled down to a low value of order 10^{-6}. We need the error to be as low as possible otherwise the cycle may fall down.

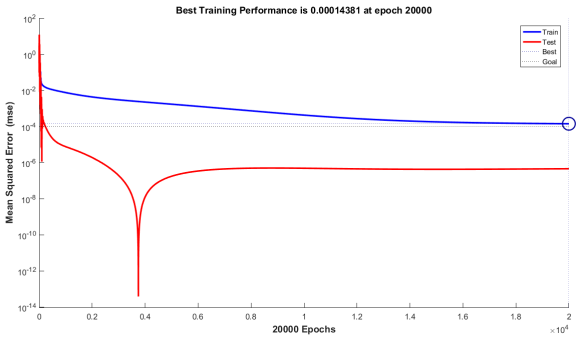

Fig. 9. Performance Index

The training and testing performance index can be seen in figure 9.

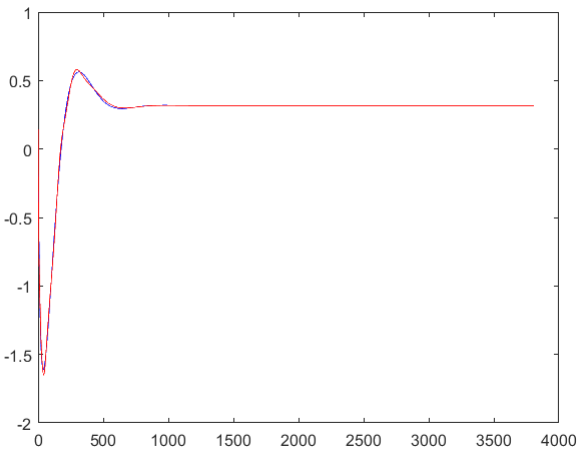

Fig. 10. Torque outputs for LQR and neural network controller

Figure 10 show the torque output for LQR (blue) and neural network (red) controller for a step input. It can be seen that they both coincide after certain time.

B. For other inputs

Once our neural network got trained we gave different input signals to the system to check if it can work for the inputs which it was not trained for.

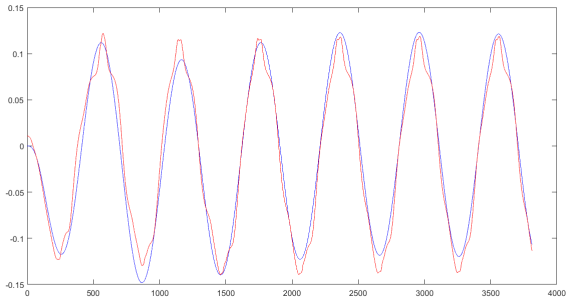

Fig. 11. Torque outputs for LQR and neural network controller for sinusoidal input

Figure 11 shows that even if we change the input type to sinusoidal neural network is performing with very small deviations from the LQR controller. As for the ramp input figure 12 tells us that the neural network follows the LQR but when the input becomes very high it becomes unable to balance the bicycle which is the limitation of our neural network controller.

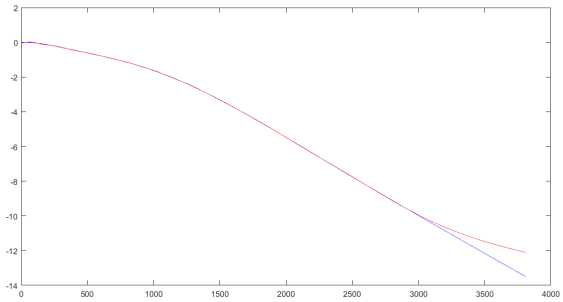

Fig. 12. Torque outputs for LQR and neural network controller for ramp input

VI. Conclusions

LQR controller is based on linear state space system thus, if any non linearity comes then we would need to model that non-linearity as a linearity and then add it to our state space model which is not possible for any dynamic system. In this paper we have seen that a neural network controller will balance the cycle for any input type even if it is not trained for that. However, if the input disturbance is very high our neural network controller will fail which can be seen in case of ramp input. But this case is of less importance since if we give a very high input we will need a very high mass of the pendulum to balance it which is not always physically possible. In that case even if our controller works properly our system will fail physically. Thus, we can say that our neural network controller is one step ahead of LQR controller since it can incorporate non-linearities and will work for different types of inputs once it is trained.

References

[1] L. Keo M. Yamakita “Dynamic Model of a Bicycle with a balancer and its control.”

[2] Mladenov, Valeri. (2011). “Application of neural networks for control of inverted pendulum.” WSEAS Transactions on Circuits and Systems. 8 .