Neural Network Based Observer Design

Anirban Sinha

December 15, 2017

Abstract

In this report we synthesized a nonlinear system-state-observer for two single-input-single-output nonlinear systems. We use neural network to capture nonlinearity of the observer. No linearity with respect to the unknown system parameters is required for the designed observer. The observer stability and boundedness of the state estimates and NN weights are proven. In order to show the effectiveness of the proposed observer, two simulations are carried out. First we modeled NN based observer for a single-link robot, rotating in vertical plane with two sets of example data sets to train the NN embeded into the observer. The second observer is designed for Van der Pol oscillator. For the single-link observer, tests are performed with no-noise training data set and with noisy dataset as well. It has been observed that NN based observer learned more accurately and faster from the no-noise data set as compared to noisy data set which leads to longer learning time.

1 Motivation

A system can be thought of as a function that takes in control input from the outside world and returns measurable states to the user. However all system internal states are not measurable, yet necessary for controlling and stabilizing the system. A system observer is a modeled-function that can estimate a system's internal states. An observer estimates system states based on the measured states from the actual system. Usually a system observer is a vector valued function that approximates actual governing differential equation of the real system.

In practice most of the system that we deal with are nonlinear in nature. When we fed in control inputs to the system, only a few of the system states are measurable. If we want to estimate internal system states of a nonlinear system using measured output from the system, we need to design nonlinear observer. Knowing system dynamic parameters, such as inertia-matrix, coriolis-matrix and gravity matrix one can derive the governing differential equations of any system. However accurate knowledge (numerical values and model) of those dynamic parameters are not always available. Further it is hard to always model all possible dynamic relationships between different parts of a system. For instance a two link manipulator has spring joints attached at each joint, is difficult to model mathematically. Most of the time we use empirical model to approximate these kind of features.

The introduction of Luenberger [1] observer led to design nonlinear observer by linearization technique of nonlinear functions. Nonlinear adaptive observer was first proposed in [2]. Marino [3]

presented a simple but restricted observer based on the satisfaction of *strictly-positive-real* (SPR) conditions. Adaptive observers with arbitrary exponential rate of convergence were considered by Marino et.al. [4, 5]. However their assumption of linearity with respect to any unknown system parameters and their conditions on transforming the original system into special canonical form are not often met for many physical systems.

Neural networks are meant to approximate nonlinear functions which are hard to model using closed form equations. We use this power of neural networks to approximate un-modeled dynamics of a system. In fact in this report we use neural network to design an observer that assumes no any prior knowledge of the system parameters. However we have been able to apply model-free neural network based observer to very simple single-input-single-output dynamical systems to achieve satisfactory results. Levin and Narendra [6] has addressed the problem of estimating unknown system state for certain discrete time nonlinear systems.

Accurate system dynamics help understand a system better and hence to control. This allows one to design a low gain controller. If nonlinearities are not properly captured a high gain control need to be employed to suppress system nonlinearities which could be infeasible in practice or very expensive. Our goal here is to capture those complex phenomena in system dynamical model using Neural Network in order to model system dynamics accurately.

2 Problem Statement

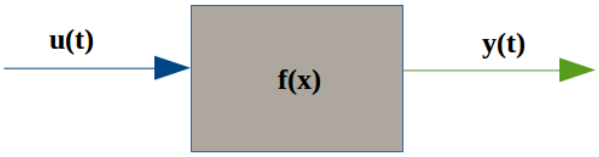


Figure 1: A nonlinear system is modeled using a vector-valued function $\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$ that maps control inputs to the system measurable states

Suppose we have a nonlinear *multi-input-multi-output* (MIMO) system that takes in $\boldsymbol{u}(t)$ as input and $\boldsymbol{y}(t)$ is the measured output (see figure 1). The nonlinear behavior of the system is assumed to be governed by the nonlinear vector valued function $\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$. The internal states of the system is denoted as $\boldsymbol{x}(t)$. The partial measurement of all the input states is denoted as $\boldsymbol{y}(t)$. Please note that we do not know the function $\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$ and internal states $\boldsymbol{x}(t)$ yet. Therefore systems dynamic behavior can be represented as following,

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{y}(t) &= \boldsymbol{C}^T \boldsymbol{x}(t) \end{aligned} \tag{1}$$

In the above equations, $\boldsymbol{x} \in \mathbb{R}^n$ is state vector, $\boldsymbol{u} \in \mathbb{R}^m$ is control input vector, and $\boldsymbol{y} \in \mathbb{R}^m$ is the observed state vector. Our objective is to approximate the vector-valued-function $\boldsymbol{f}$ by designing

a neural network based nonlinear observer, using known time series values of $\boldsymbol{u}$ and target output $\boldsymbol{y}$.

An observer for a system of the form as in equation (1) can be represented as following,

$$\begin{aligned} \dot{\hat{\boldsymbol{x}}}(t) &= \boldsymbol{A}\hat{\boldsymbol{x}}(t) + \hat{\boldsymbol{g}}(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t)) + \boldsymbol{G}(\boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t)) \\ \hat{\boldsymbol{y}}(t) &= \boldsymbol{C}^T\hat{\boldsymbol{x}}(t) \end{aligned} \tag{2}$$

where, the matrix $\boldsymbol{G} \in \mathbb{R}^{n\times m}$ is known as observer gain matrix selected by the designer. In practice we consider $\boldsymbol{G}$ to be a diagonal matrix in order to maintain linear modification of error states defined by $\boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t)$.

The reason behind nonlinear observer in equation (1) has that form is as following,

- The entity $\boldsymbol{A}\hat{\boldsymbol{x}}(t) + \boldsymbol{G}(\boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t))$ works as an observer of a linear system. The matrix $\boldsymbol{A}$ is chosen in such a manner that $\boldsymbol{A} + \boldsymbol{GC^T}$ becomes *Hurwitz*. In control theory *Hurwitz* matrix indicates that real parts if all its eigen values are strictly negetive.
- The term $\boldsymbol{g}(\hat{\boldsymbol{x}}(t))$ approximates the behavior of $\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) - \boldsymbol{A}\boldsymbol{x}(t)$

We design nonlinear observer by approximating $\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) - \boldsymbol{A}\boldsymbol{x}(t)$ using neural networks, with an update rule that ensures stability of the observer according to *Lyapunov stability criterion*. In other words the weights of the neural network is updated in such a manner that the observer output $\hat{\boldsymbol{y}}(t)$ is always bounded.

3 Problem Formulation

The problem formulation is carried out for a particular type of SISO nonlinear system as in equation (3). However this particular choice of system model dose not hurt the generality of problem formulation.

We want to design an observer for the system which has the form as below. The functions $\boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ are assumed to be continuous and nonlinear. The matrix $\boldsymbol{A}$ is a *Hurwitz matrix*, selected by the designer so that the pair $(\boldsymbol{C}, \boldsymbol{A})$ is observable.

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}\left[\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}\right] \\ \boldsymbol{y} &= \boldsymbol{C^T}\boldsymbol{x} \end{aligned} \tag{3}$$

Observer for this system is defined as

$$\begin{aligned} \dot{\hat{\boldsymbol{x}}} &= \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{b}\left[\hat{\boldsymbol{f}}(\hat{\boldsymbol{x}}) + \hat{\boldsymbol{g}}(\hat{\boldsymbol{x}})\boldsymbol{u} + \boldsymbol{G}(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k)\right] \\ \hat{\boldsymbol{y}} &= \boldsymbol{C}^T\hat{\boldsymbol{x}} \end{aligned} \tag{4}$$

Our next step is to find a good approximation of two continuous nonlinear functions $\boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ using two independent neural networks.

3.1 NN Observer Design

The continuous nonlinear functions in the system (3) can be represented by NNs with constant 'ideal' weight $\boldsymbol{W}$ and sufficient number of basis functions as follow,

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}) &= \boldsymbol{W}_f^T \boldsymbol{\sigma}(\boldsymbol{V}_{\boldsymbol{f}}^{\boldsymbol{T}} \boldsymbol{x}) + \boldsymbol{\varepsilon}_f \\ \boldsymbol{g}(\boldsymbol{x}) &= \boldsymbol{W}_g^T \boldsymbol{\sigma}(\boldsymbol{V}_{\boldsymbol{g}}^{\boldsymbol{T}} \boldsymbol{x}) + \boldsymbol{\varepsilon}_g \end{aligned} \tag{5}$$

We assume that the 'ideal' weights $\boldsymbol{W}_f$ and $\boldsymbol{W}_g$ are bounded by known positive values so that,

$$\|\boldsymbol{W}\|_{i,F} \leq \|\boldsymbol{W}\|_{i,M}, \quad i = f, g$$

where $\|\boldsymbol{W}\|_{i,M}$ are known values.

Let the NN functional estimates for the nonlinear functions of $\boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ be given by,

$$\begin{aligned} \hat{\boldsymbol{f}}(\hat{\boldsymbol{x}}) &= \hat{\boldsymbol{W}}_f^T \boldsymbol{\sigma}(\hat{\boldsymbol{V}}_{\boldsymbol{f}}^{\boldsymbol{T}} \hat{\boldsymbol{x}}) \\ \hat{\boldsymbol{g}}(\hat{\boldsymbol{x}}) &= \hat{\boldsymbol{W}}_g^T \boldsymbol{\sigma}(\hat{\boldsymbol{V}}_{\boldsymbol{g}}^{\boldsymbol{T}} \hat{\boldsymbol{x}}) \end{aligned} \tag{6}$$

where the current weights $\hat{\boldsymbol{W}}_f$ and $\hat{\boldsymbol{W}}_g$ are provided by the weight tuning algorithms.

After certain manipulations it can be showed that,

1. Proposed observer system is,

$$\begin{aligned} \dot{\hat{\boldsymbol{x}}} &= \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{b}\left[\hat{\boldsymbol{W}}_f^T \boldsymbol{\sigma}(\hat{\boldsymbol{V}}_{\boldsymbol{f}}^{\boldsymbol{T}} \hat{\boldsymbol{x}}) + \hat{\boldsymbol{W}}_g^T \boldsymbol{\sigma}(\hat{\boldsymbol{V}}_{\boldsymbol{g}}^{\boldsymbol{T}} \hat{\boldsymbol{x}})\boldsymbol{u}\right] + \boldsymbol{G}(\boldsymbol{y} - \boldsymbol{C}\hat{\boldsymbol{x}}) \\ \hat{\boldsymbol{y}} &= \boldsymbol{C}^T \hat{\boldsymbol{x}} \end{aligned} \tag{7}$$

2. Observation error dynamics is,

$$\begin{aligned} \dot{\tilde{\boldsymbol{x}}} &= (\boldsymbol{A} + \boldsymbol{G}\boldsymbol{C}^{\boldsymbol{T}})\tilde{\boldsymbol{x}} + \boldsymbol{b}\left[\tilde{\boldsymbol{W}}_f^T \boldsymbol{\sigma}(\tilde{\boldsymbol{V}}_{\boldsymbol{f}}^{\boldsymbol{T}} \hat{\boldsymbol{x}}) + \boldsymbol{w}_f(t) + \{\tilde{\boldsymbol{W}}_g^T \boldsymbol{\sigma}(\tilde{\boldsymbol{V}}_{\boldsymbol{g}}^{\boldsymbol{T}} \tilde{\boldsymbol{x}}) + \boldsymbol{w}_g(t) + \boldsymbol{\varepsilon}_g\}\boldsymbol{u} + \boldsymbol{\varepsilon}_f\right] \\ \tilde{\boldsymbol{y}} &= \boldsymbol{C}^T \tilde{\boldsymbol{x}} \end{aligned} \tag{8}$$

   where, for $i = f, g$

$$\begin{aligned} \tilde{\boldsymbol{W}}_i &= \boldsymbol{W}_i - \hat{\boldsymbol{W}}_i \\ \boldsymbol{w}_i(t) &= \boldsymbol{W}_i^T \tilde{\boldsymbol{\sigma}}_i \end{aligned}$$

3.2 Weight Update Rule

The neural network weight update rules are derived from Lyapunov stability criterion for a specific choice of Lyapunov candidate function. Here we will just mention the main theorem describing weight update rule such that at any point of time the estimated states $\hat{\boldsymbol{x}}(t)$ is always *Uniformly Ultimately Bounded*. The detailed proof of the theorem can be found in [7].

Theorem : Assume the control input $\boldsymbol{u}(t)$ is bounded by positive constant $\boldsymbol{u}_d$ and that for the given vector $\boldsymbol{b}$, for the transfer function $\boldsymbol{W}(s)$ realized by tuple $(\boldsymbol{A}+\boldsymbol{G}\boldsymbol{C}^T)$ is an *strictly positive real* (SPR) transfer function. let the robustifying terms be given by,

$$\boldsymbol{\nu}_i(t) = -\boldsymbol{D}_i\tilde{\boldsymbol{y}}/|\tilde{\boldsymbol{y}}| \tag{9}$$

with $\boldsymbol{D}_f \geq a_f$ and $\boldsymbol{D}_g \geq a_g$, weight tunings be provided by,

$$\begin{aligned} \dot{\hat{\boldsymbol{W}}}_f &= \boldsymbol{F}_f\hat{\boldsymbol{\sigma}}_f\tilde{\boldsymbol{y}} - \boldsymbol{\kappa}_f\boldsymbol{F}_f|\tilde{\boldsymbol{y}}|\boldsymbol{W}_f \\ \dot{\hat{\boldsymbol{W}}}_g &= \boldsymbol{F}_g\hat{\boldsymbol{\sigma}}_g\tilde{\boldsymbol{y}}\boldsymbol{u} - \boldsymbol{\kappa}_g\boldsymbol{F}_g|\tilde{\boldsymbol{y}}|\hat{\boldsymbol{W}}_g \end{aligned} \tag{10}$$

where $\boldsymbol{F}_i = \boldsymbol{F}_i^T > 0$ (*symmetric positive definite*) is any constant design matrix governing the speed of convergence and $\boldsymbol{\kappa}_i > 0$ is a design parameter with $i = f, g$. Then the state estimates $\hat{\boldsymbol{x}}(t)$ and the NN weight estimates $\hat{\boldsymbol{W}}(t)$ are UUB.

4 Learning form Data Scheme

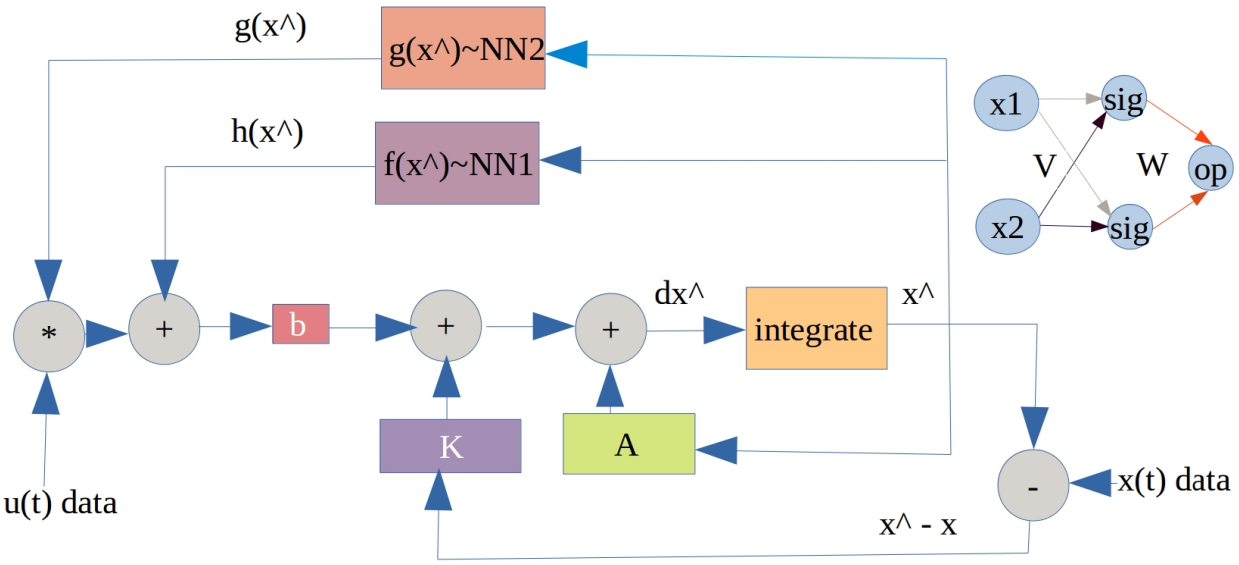


Figure 2: Learning from data scheme. (Top right) Basic neural block in **NN1** and **NN2**

The figure 2 shows the basic outline of training the NN's (NN1 and NN2 approximating functions $\boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ respectively). A simple two-layer neural network has been used as building block of the proposed observer (shown in top right). In the figure above, the symbols $+, -, *$ are used to represent *addition*, *subtraction* and *multiplication* respectively. The arrow heads denote directions of data flow in the block diagram.

In order to maintain simplicity, we assume that input layer weight matrix $\boldsymbol{V}$ is a constant matrix with all its elements as 1. This assumption leave us only to determine hidden layer weight matrix $\boldsymbol{W}$ only. As a first step we randomly initialize the weight $\boldsymbol{W}$ and observer states as $\hat{\boldsymbol{x}}(0) = \boldsymbol{0}$.

5 Simulation Results

Our simulation results are based on two SISO systems, namely, a single-DoF robot arm rotating in a vertical plane and Van der Pol oscillator. Learning dataset is created synthetically by using some nonlinear dynamic equation of motions.

5.1 One-DoF Vertically Rotating Robot

We considered a single-link robot rotating in a vertical plane whose equations of motion are,

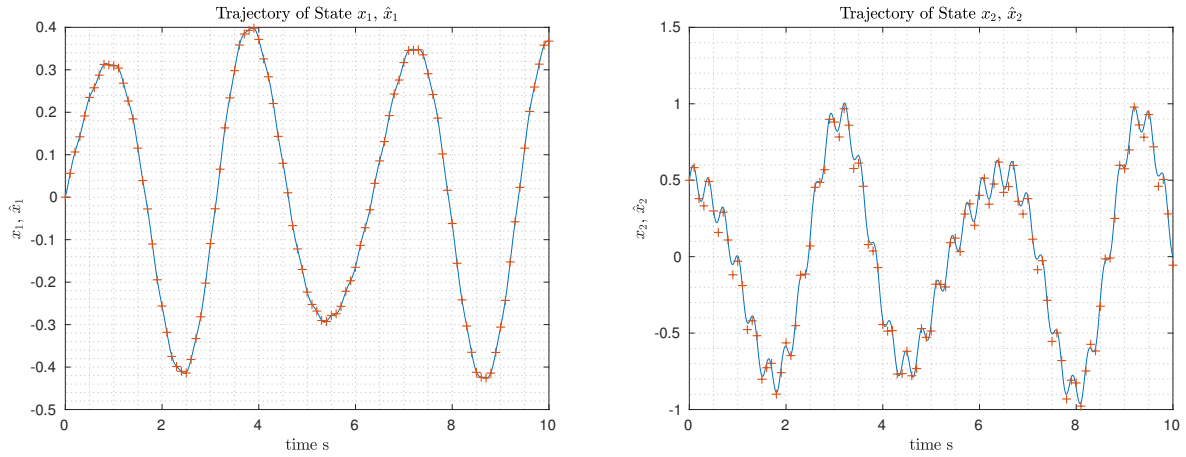
$$\begin{aligned} M\ddot{q} + \frac{1}{2}mgl\sin(q) &= u \\ y &= q \end{aligned} \tag{11}$$

In state-space form of equation (11) can be represented by assuming $x_1 = q$ and $x_2 = \dot{q}$ as following,

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} (u - \frac{1}{2}mgl\sin x_1)/M \\ y &= x_1 \end{aligned} \tag{12}$$

Using the above dynamic equation, we generated two data sets by feeding in two different control inputs into equation (12) and recorder the corresponding joint states. The first control input is, $u(t) = \sin(t)$ whereas the second control input is $u(t) = \sin(2t) + \cos(2t)$. The corresponding data files are named as *Lewis_data*1.*txt* and *Lewis_data*2.*txt* respectively. The required system parameter values are taken from [7].

At the validation step, the trained neuro-observer is asked to estimate joint states corresponding to $u(t) = \sin(2t) + \cos(20t)$ which is unseen control input into the observer.



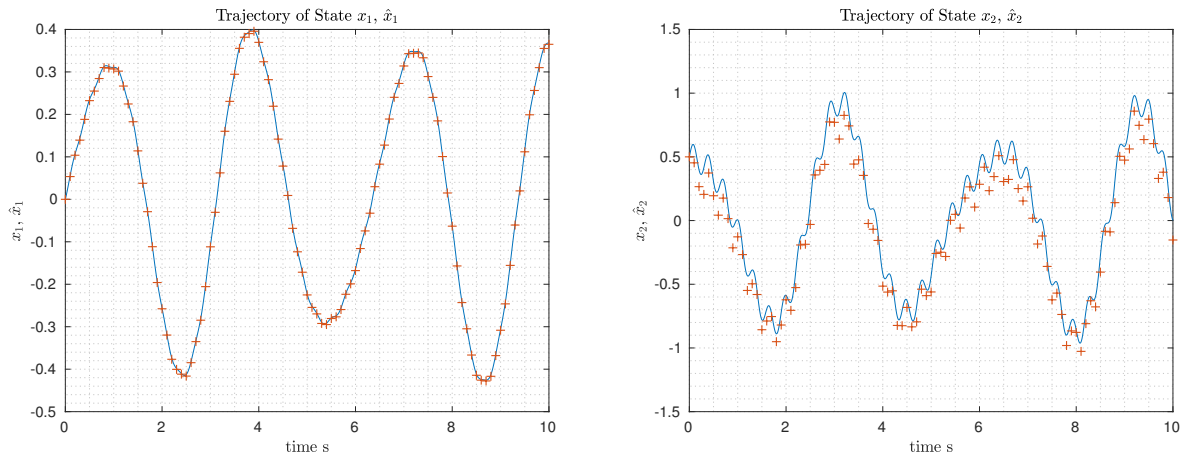
(a) Actual and estimated first state

(b) Actual and estimated second state

Figure 3: State output of the trained NN-observer for input $u(t) = \sin(2t) + \cos(20t)$.

The figures 3a and 3b shows estimate of the system states of the trained neuro-observer to unseen control input $u(t) = \sin(2t) + \cos(20t)$. The final weight vectors are found to be $\boldsymbol{W}_f = [1.0267 \quad -$

5.9631] and $\boldsymbol{W}_g = [-2.7019 \quad -2.3718]$ respectively. Notice that, the trained NN-observer is capable of estimating states with very less amount of observation error.



(a) Actual and estimated first state

(b) Actual and estimated second state

Figure 4: State output of the trained NN-observer with noisy dataset for input $u(t) = \sin(2t) + \cos(20t)$.

In order to see the performance of the NN learning method, we next train the network with the same two examples but this time corrupted with Gaussian noise. The state output to the validation control input $u(t) = \sin(2t) + \cos(20t)$ are shown in figure 4a and 4b. The final weight vectors in this case are found to be $\boldsymbol{W}_f = [1.8416 \quad -31.1121]$ and $\boldsymbol{W}_g = [0.1532 \quad -7.8790]$ respectively. Unlike previous case, NN-observer trained with noisy data does not perform well in estimating the second state correctly, although the first state estimation looks satisfactory.

For both the cases (uncorrupted and noisy data-sets) during the NN observer training the following parameter values has been used, $\hat{\boldsymbol{x}}(0) = [0 \quad 0]^T$, $\boldsymbol{G} = [40 \quad 5e3]^T$, $\boldsymbol{\kappa}_f = \boldsymbol{\kappa}_g = 0.001$, $\boldsymbol{F}_f = \text{diag}[5e4 \quad 5e4]$, $\boldsymbol{F}_g = \text{diag}[5e3 \quad 5e3]$.

5.2 Vander Pol Oscillator

We here consider a Van der Pol oscillator whose equations of motion are,

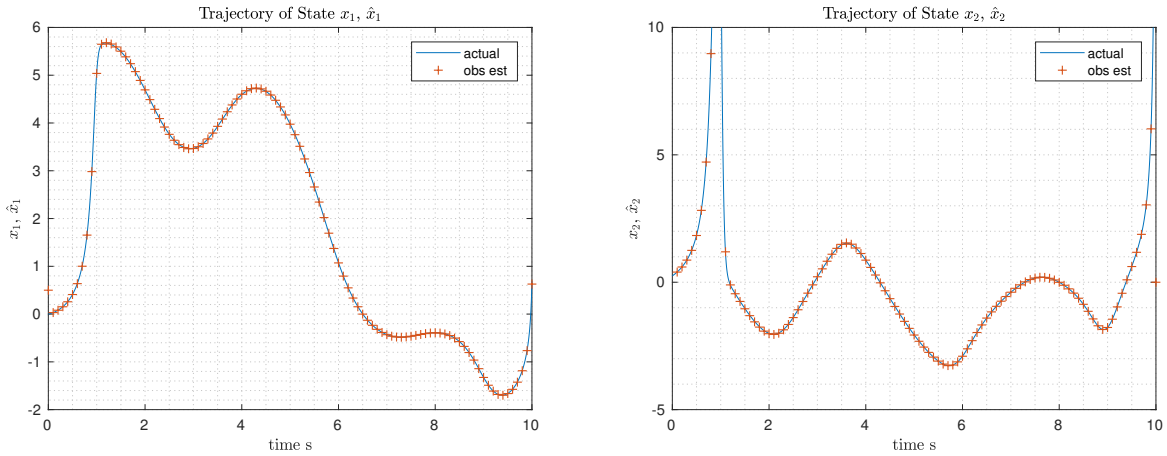
$$\begin{aligned} \ddot{q} + (q^2 - 1)\dot{q} + q &= (1 + q^2 + \dot{q}^2)u \\ y &= x \end{aligned} \tag{13}$$

The state-space representation is,

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \{(1 - x_1^2)x_2 - x_1 + (1 + x_1^2 + \dot{x}_2^2)u\} \\ y &= x_1 \end{aligned} \tag{14}$$

Using the above dynamic equation, we generated two data sets by feeding in two different control inputs into equation (14) and recorder the corresponding joint states. The first control input is, $u(t) = \sin(4t) + \cos(10t)$ whereas the second control input is $u(t) = \sin(4t) + \cos(4t)$.

Once the training is over, the trained observer is asked to estimate states corresponding to $\sin(2t) + \cos(2t)$ and the corresponding observer response is shown in figure 5a and 5b. The final weight vectors are found to be $\boldsymbol{W}_f = [0.1215 \quad 0.6897]$ and $\boldsymbol{W}_g = [1.6610 \quad 2.3310]$ respectively.



(a) Actual and estimated first state

(b) Actual and estimated second state

Figure 5: State output of the trained NN-observer with noisy dataset for input $u(t) = \sin(2t) + \cos(2t)$.

6 Conclusion

- We designed a neural network based observer to estimate system states. Weight updating scheme is formulated in a manner so that it respects system stability as per **Lyapunov**'s stability criterion.
- Two sets of time series data is used to train the neural network. When uncorrupted data is used to train the network, the output looks satisfactory for the choice of validation set. On the other hand, when we trained the NN-observer with noisy data, output at the validation stage does not seem very satisfactory.

7 Future Work

Here we have used simplest kind of neural network to design a nonlinear observer. We believe the proposed observer may not be robust enough to estimate states of complex systems with many inputs and outputs. In future we would like to implement this observer designing methodology to more complex systems like redundant degree of freedom robots. Further instead of using simple NN we can use recurrent neural network with an LSTM cell which seems to have potential to remember output of most recent states to estimate the future states.

References

[1] D. G. Luenberger, "Observing the states of linear systems," *IEEE Trans. Military Electron*, vol. 8, pp. 74–90, 1964.

[2] G. Bastin and M. R. Gevers, "Stable adaptive observer for nonlinear time varying systems," *IEEE Trans. Auto. Ctrl.*, vol. 33, no. 7, pp. 650–657, 1988.

[3] R. MArino, "Adaptive observer for single output nonlinear systems," *IEEE Trans. Auto. Ctrl.*, vol. 35, no. 9, 1990.

[4] R. Marino and P. Tomei, "Global adaptive observer for nonlinear systems via filtered transformations," *IEEE Trans. Auto. Ctrl.*, vol. 37, no. 8, pp. 1239–1245, 1992.

[5] R. Marino and P. Tomei, "Adaptive observers with arbitrary exponential rate of convergence for nonlinear systems," *IEEE Trans. Auto. Ctrl.*, vol. 40, no. 7, pp. 1300–1304, 1995.

[6] A. U. Levin and K. S. Narendra, "Control of nonlinear dynamical systems using neural networks-part ii," *IEEE Trans. Neural Networks*, vol. 7, no. 1, pp. 30–42, 1996.

[7] Y. H. Kim, F. L. Lewis, and C. T. Abdallah, "Nonlinear observer design using dynamic recurrent neural networks," *Proceedings of the 35th Conference on Decision and Control, Kobe, Japan*, 1996.

8 Appendix

8.1 Useful Terminologies and Definitions

8.1.1 Observer

In control theory, a state observer is a system that provides an estimate of the internal state of a given real system, from measurements of the input and output of the real system.

A system is observable means, it is possible to fully reconstruct the system states from its output measurements using the state observer.

8.1.2 $L_\propto$ Space

This space consists of all those functions which takes in $\boldsymbol{x} \in X$ and returns $\boldsymbol{y} \in \mathbb{R}^m$ such that $\inf_{\boldsymbol{x} \in X} sup|\boldsymbol{y}| < + \propto$, where X is the domain of an element in the space.

8.1.3 Vector Valued Function

A vector valued function is such that it takes in a scalar or vector $\boldsymbol{x} \in \mathbb{R}^m$, $m \geq 1$ as input but outputs a finite or infinite dimensional vector $\boldsymbol{y} \in \mathbb{R}^n$, where $1 \leq n <\propto$.

$$\boldsymbol{f}(\boldsymbol{x}) = \{\boldsymbol{f} : \mathbb{R}^m \rightarrow \mathbb{R}^n\} \tag{15}$$

8.1.4 Hurwitz Matrix

Given a polynomial,

$$p(x) = a_0 x^n + a_1 x^{n-1} + \ldots \cdots + a_{n-1} x + a_n \tag{16}$$

we can construct a $N \times N$ matrix H of the form

$$H = \begin{bmatrix} a_1 & a_3 & a_5 & \cdots & \cdots & \cdots & 0 & 0 & 0 \\ a_0 & a_2 & a_4 & & & & \vdots & \vdots & \vdots \\ 0 & a_1 & a_3 & & & & \vdots & \vdots & \vdots \\ \vdots & a_0 & a_2 & \ddots & & & 0 & \vdots & \vdots \\ \vdots & 0 & a_1 & & \ddots & & a_n & \vdots & \vdots \\ \vdots & \vdots & a_0 & & & \ddots & a_{n-1} & 0 & \vdots \\ \vdots & \vdots & 0 & & & & a_{n-2} & a_n & \vdots \\ \vdots & \vdots & \vdots & & & & a_{n-3} & a_{n-1} & 0 \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & a_{n-4} & a_{n-2} & a_n \end{bmatrix} \tag{17}$$

is called Hurwitz matrix corresponding to the polynomial P.
It was established by Adolf Hurwitz in 1895 that a real polynomial is stable (i.e., all its roots have strictly nonnegative real part) iff all the leading principle minors of the matrix H are positive. Mathematically,

$$\Delta_1(p) = |a_1| \implies a_1 > 0$$

$$\Delta_2(p) = \begin{vmatrix} a_1 & a_3 \\ a_0 & a_2 \end{vmatrix} = a_2 a_1 - a_0 a_3 > 0$$

$$\Delta_3(p) = \begin{vmatrix} a_1 & a_3 & a_5 \\ a_0 & a_2 & a_4 \\ 0 & a_1 & a_3 \end{vmatrix} = a_3 \Delta_2 - a_1(a_1 a_4 - a_0 a_5) > 0$$

and so on.
In engineering and stability theory, a square matrix A is called stable matrix (or sometimes Hurwitz matrix) if every eigenvalue of $\boldsymbol{A}$ has strictly negative real part, i.e.,

$$Re[\lambda_i] < 0 \quad \forall i$$

$\boldsymbol{A}$ is also called a stability matrix, because then the differential equation,

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$$

is asymptotically stable.