Analysis of System Dynamics Approximation by Neural Network

Automation

How can we mesure the performance of a Neural Network as a perturbation approximator ?

Viozelange Matis

date : September 9, 2025

École Centrale Nantes

Contents

List of Figures

List of Tables

Problem statement

1.1 Context

The SuperTwisting algorithm is a known robust control algorithm based on sliding mode control theory. It can be used on dynamical systems with great uncertainties and disturbances. The counterpart of such theory is the apparition of chattering phenomenon, which is a high frequency oscillation of the control signal. This phenomenon can be harmful for the system and can lead to mechanical failures. The goal of this project is to add a new type of observer based on ANN to substract the inpact of unknown perturbation on the system.

We base our work on the following article [?] : This report presents an overview and a performance analysis of the controler presented bellows. What are the effect of the Neural Network parameters on the approximation of the perturbation ? How the performance are affected by the perturbation ?

1.2 Problem statement

We are working on a Python simulation (available on this GitHub repository). Run the GUI.py file to lanch the interface. The results are simulated on two dynamical systems:

- A simple perturbed system:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = u + a\sin(t) \end{cases} \tag{1.1}$$

  where $a = 5$.

- A more complex pendulum system with a time-varying length:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{2\dot{l}(t)}{l(t)}x_2 - \frac{g}{l(t)}\sin(x_1) + 2\sin(t) + \frac{1+0.5\sin(t)}{m\cdot l(t)^2}u \end{cases} \tag{1.2}$$

  where:

  - $g = 9.81\,\mathrm{m/s^2}$ is the gravitational constant,
  - $m = 2\,\mathrm{kg}$ is the mass of the pendulum,
  - $l(t) = 0.8 + 0.1\sin(8t) + 0.3\cos(4t)$ is the length of the pendulum as a function of time,
  - $\dot{l}(t)$ represents the time derivative of $l(t)$.

To evaluate the performance of the Neural Network in approximating the perturbation, we :

1. Discretize the signals of the perturbation and its Neural Network approximation with a sampling time of $\Delta t = 0.0001$ s.

2. Compute the metrics shown below.

- **Mean Squared Error (MSE)**: The mean squared error between the perturbation and the Neural Network approximation is given by:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (p_i - \hat{p}_i)^2 \tag{1.3}$$

  where p_i is the true perturbation value, $\hat{p}_i$ is the Neural Network's approximation, and N is the total number of samples.

- **Standard Deviation of the Error**: To evaluate the stability of the Neural Network, we calculate the standard deviation of the error:

$$\sigma_e = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (e_i - \bar{e})^2} \tag{1.4}$$

  where $e_i = p_i - \hat{p}_i$ is the error at each sample, and $\bar{e}$ is the mean error.

- **Correlation Coefficient**: To evaluate the quality of the approximation, we compute the correlation coefficient between the true perturbation and the Neural Network approximation:

$$r = \frac{\sum_{i=1}^{N}(p_i - \bar{p})(\hat{p}_i - \bar{\hat{p}})}{\sqrt{\sum_{i=1}^{N}(p_i - \bar{p})^2 \sum_{i=1}^{N}(\hat{p}_i - \bar{\hat{p}})^2}} \tag{1.5}$$

  where $\bar{p}$ and $\bar{\hat{p}}$ are the mean values of the true perturbation and the Neural Network approximation, respectively.

We are going to evaluate the performance of the Neural Network in approximating the perturbation with different values of the following parameters:

1. **n** : the number of neurons in the hidden layer of the Neural Network.
2. γ : the learning rate of the Neural Network.

Neural Network parameters analysis

2.1 Basic system

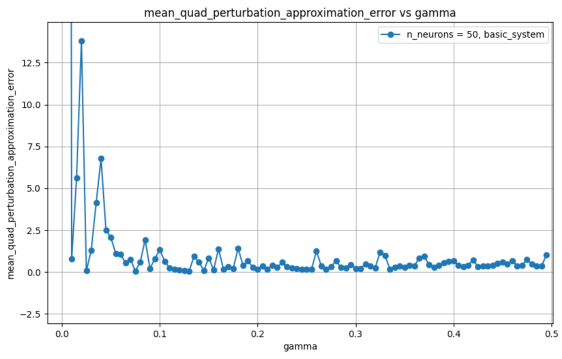
2.1.1 First results

We will see the performance of the Neural Network in approximating the perturbation of the basic system. The perturbation is given by $a\sin(t)$ with $a = 5$.

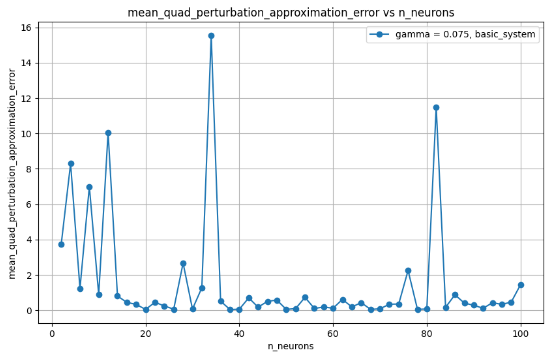
The default parameters of the Neural Network are as follows:

- **Number of neurons in the hidden layer**: n = 50
- **Learining rate**: $\gamma = 0.075$

We can plot the mean squared error for each default parameter :



(a) MSE, basic system with $n = 50$



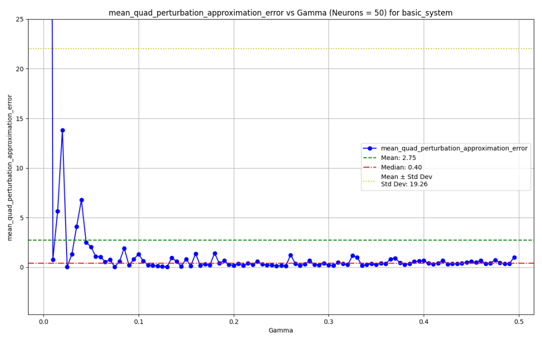
(b) MSE, basic system with $\gamma = 0.075$

Figure 2.1: MSE for the basic system

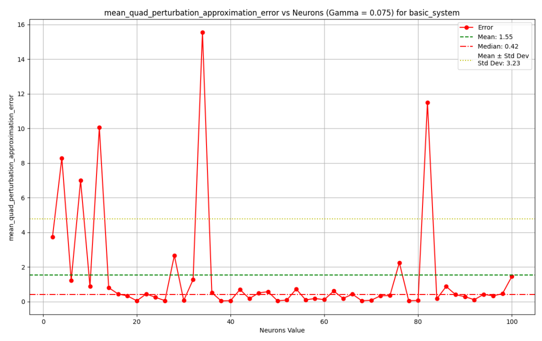
We can see that the mean squared error is decreasing with the number of neurons and the increase of the learning rate. On one hand, the error is stabilising on 0.7 ± 0.5 for $\gamma > 0.1$. Although, it would be intersting to see the same metrics with different values of n. To assure this statement. On the other hand, it's difficult to see the impact of the neurons number with $\gamma = cst$. Indeed the error seems to be decreasing with the number of neurons but the presence of outliers can alert on the overfit issue. It also safe to say that we want more than 25 neurons to have a good approximation.

These plot give some insights on the performance of the approximation regarding certain parameters. Unfortunately, we can only see a part of the dataset and not well. We can compute the mean, median and standard divation of the curve to get quantitative results. We can go further and plot these results for each constant parameters. To resume, we chose a type of error (MSE, standard deviation or correlation), we also chose the *"varying"* parameter and the dynamic system. Then we can plot the mean, median and standard deviation of the error curve of the *"varying"* parameter for each *"constant"* parameter.

First, here is a plot of the mean, median and standard deviation of the MSE curve of the figure. **??** :



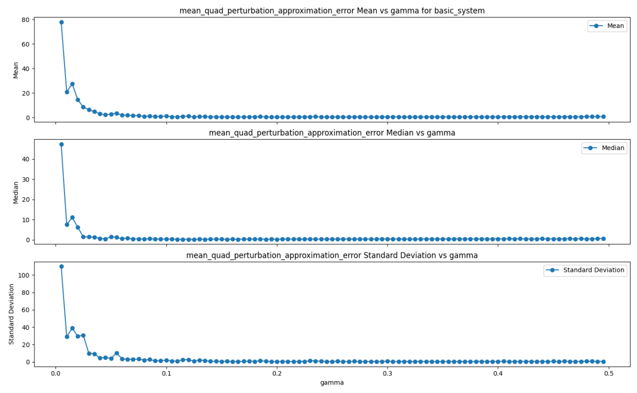
(a) MSE, basic system with $n = 50$



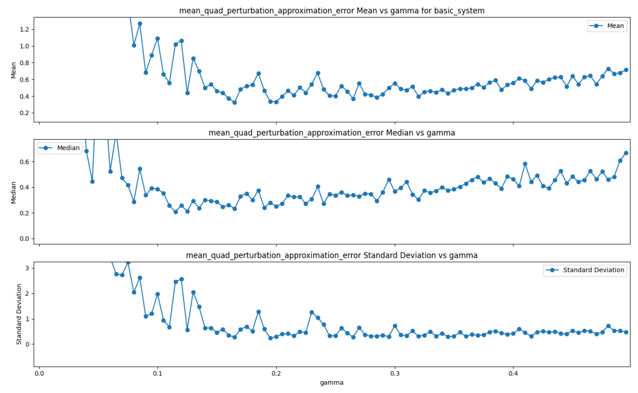
(b) MSE, basic system with $\gamma = 0.075$

Figure 2.2: MSE for the basic system with its statistics

Now we can see the impact of the number of neurons and the learning rate on the performance of the Neural Network approximation more quantitatively.



(a) MSE, *gamma*



(b) MSE, *gamma* zoomed on y-axis

Figure 2.3: MSE for all neurons curve for each $\gamma = cst$

2.1.2 Best tuning region

We can now see all the plot of median, mean and standard deviation for every error index and start to see for which parameters of the Neural Network the approximation is the best.

We must be careful not to make hasty conclusions. As the standard deviation increase with the number of neurons and γ, that means that the error is more spread. So can have good results to with high values of γ and n. But the simulation showed quite intersting results.

2.1.3 Neural Network questionings

We can now ask ourselves some questions about the Neural Network :

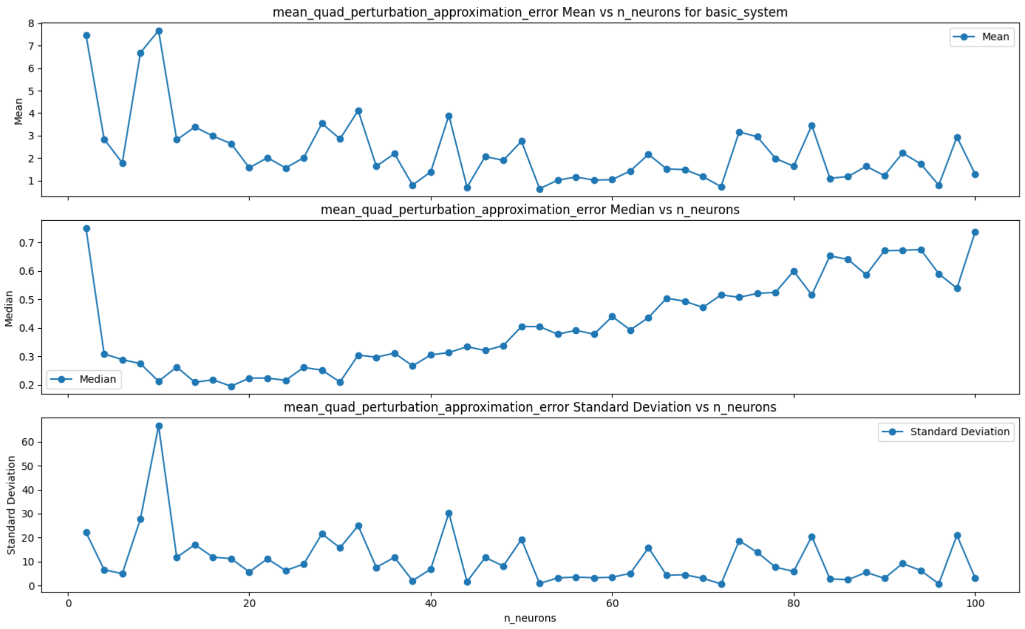
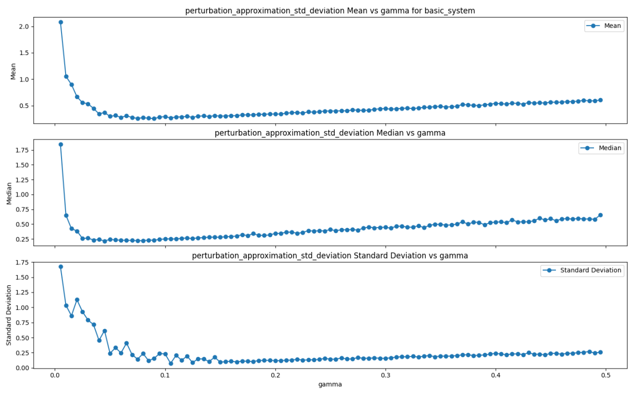
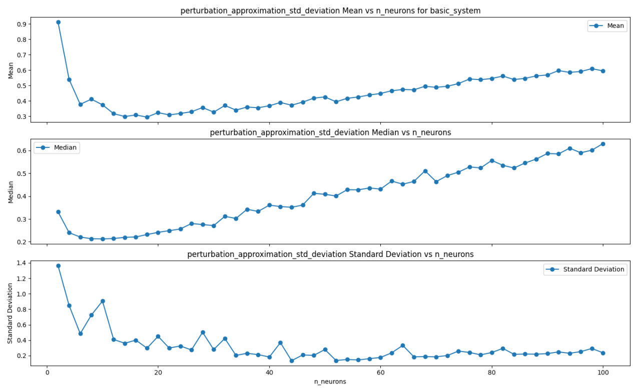


Figure 2.4: MSE for all γ curve for each $n\_neurons = cst$

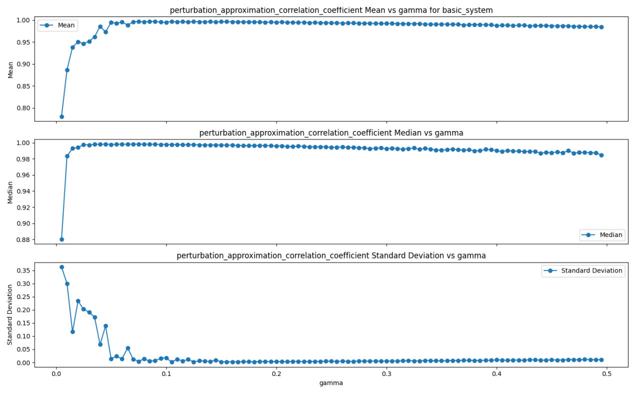


(a) Standard devation for all neurons curve for each $\gamma = cst$

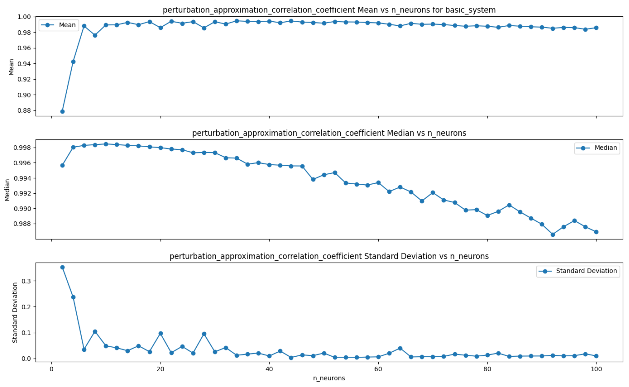


(b) Standard deviation for all γ curve for each $n\_neurons = cst$

Figure 2.5: Standard deviations for the basic system



(a) Correlation coefficient for all neurons curve for each $\gamma = cst$



(b) Correlation coefficient for all γ curve for each $n\_neurons = cst$

Figure 2.6: Correlation coefficient for the basic system

- What is the Neural Network's contribution to the robustness of the system ?
- Could it work with a static gain STWC ?
- Can it works in open-loop ?
- The dynamic nature of the weights is really worth it compare to an off-loop Neural Network ?

Open Loop approximation

By hypothesis, the output of the Neural Network is used in the control law. As it is a Lyapunov design, the bondaries of the approximation regarding the the real perturbation are assured by the stability of the system that is himself assured by the Lyapunov function that used the dynamics of the Neural Network wieghts.

To resume, there is no reason that the Neural Network approximation is good in open-loop.

Static gain approximation

The proof is based on the fact that the Neural Network works woth an ASTWC. It would be intersting to create a new Neural Network that works with a static gain and tune them to unable the control to counter the perturbation and see if the Neural Network add some robustness to the system.

But as we said before, we have to find new dynamic weights definition.

Neural Network contribution

It is prooved in Mohammad article that the ouput presents less chattering with the Neural Network. We can see that the gains are also smaller du to the anticipation of the perturbation dynamics.

But, the analysis shows that the approximation's quality is linked to the stability of the system. If the system isn't reaching the sliding surface quickly, the Neural Network will not be able to approximate the perturbation.

Conclusion

Bibliography

[1] Aarkan. Sobel filter kernel of large size. https://stackoverflow.com/questions/9567882/sobel-filter-kernel-of-large-size, 2012. Accessed: 2024-08-21.

[2] Philippe Babin, Philippe Giguère, and François Pomerleau. Analysis of robust functions for registration algorithms. In *2019 International Conference on Robotics and Automation (ICRA)*, Palais des congres de Montreal, Montreal, Canada, May 20-24 2019. IEEE. Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019.

[3] Marine Pétriaux Jean-Emmanuel Deschaud Fabio Elnecave Xavier, Guillaume Burger and François Goulette. Multi-imu proprioceptive state estimator for humanoid robots. https://arxiv.org/abs/2307.14125, 2023.

[4] Péter Fankhauser, Michael Bloesch, and Marco Hutter. Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robotics and Automation Letters*, 3(4), October 2018.

[5] Ignacio Vizzo, Benedikt Mersch, Tiziano Guadagnino, et al. Kiss-icp: In defense of point-to-point icp simple, accurate, and robust registration if done the right way. *IEEE Robotics and Automation Letters*, 8(2):113–120, February 2023.

[6] Wikipedia contributors. Algorithme de tracé de segment de bresenham. https://fr.wikipedia.org/wiki/Algorithme_de_trac%C3%A9_de_segment_de_Bresenham, 2024. Accessed: 2024-08-21.

Annexes