SAPIENZA
UNIVERSITÀ DI ROMA

SAPIENZA UNIVERSITY OF ROME

Faculty of Information Engineering, Informatics and Statistics

Department of Computer, Control and Management Engineering (DIAG)

ROBOTICS2 PROJECT
PROF: ALESSANDRO DE LUCA
TUTOR: PIETRO PUSTINA

Data-driven identification of a one-link robot with flexible joint

Lorenzo Cirillo 1895955
Claudio Schiavella 1884561

July 26, 2023

Contents

1 Introduction

A manipulator robot can be described mathematically through its dynamic model. The latter illustrates the relationships between the generalized forces applied to the robot and the resulting motion of its mechanical structure. Various approaches exist to derive the model, such as D'Alembert's or Hamilton's principles. Specifically from the Hamiltonian approach are derived the Euler-Lagrange equations, used for modeling and control. This type of approach is energetic. It allows us to derive in symbolic form the equations of motion, enabling and encouraging the analysis of the dynamic properties of the robot and the possible controls to be applied to it.
In such a context, it is very easy to make a connection with neural networks. Indeed,

the latter have demonstrated excellent capabilities in modeling systems over time, achieving very good performance and quality results. However, a generic neural network has difficulty generalizing to contexts in which conservation laws are present. Such a problem in the field of robotics is not good.
For this reason, [1] introduced a new type of neural network, called Lagrangian Neural Network (LNN), able to learn conservation laws
Our project uses both a baseline neural network and a LNN to learn the Lagrangian and consequently the dynamic model of single-link robots, with a rigid or elastic joint. The aim is to verify the functioning of the two methods, analyzing results and problems.
Consequently, the report will be structured as follows:

- Section 2 introduces a parametric approach to the Euler-Lagrange equations based on Lagrangian functions, as proposed in [1].
- Section 3 presents the dynamic models of the robots used in the simulations.
- Section 4 presents the architectures of the neural networks used in the simulations.
- Section 5 presents the settings and results of the simulations in various application contexts.

2 Parametric Lagrangian

The Lagrangian approach is based on a set of generalized coordinates $q \in \mathbb{R}^N$, where n is the number of degrees of freedom of the robot. The Lagrangianis defined as difference between the kinetic energy $T(q, \dot{q})$ and the potential energy $U(q)$ of the system, i.e.

$$L(q, \dot{q}) = T(q, \dot{q}) - U(q).$$

From this point we derive the Euler-Lagrange equations, assuming for our context the absence of dissipative forces

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = 0 \qquad \text{for} \quad i = 1, ..., N. \tag{1}$$

Expanding (1) for robotic systems, we obtain the well-known equation for manipulators

$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = 0.$$

As suggested by the authors of [1], the next step would be to find an analytical expression of the Lagrangian function and then expand the Euler-Lagrange equations to identify the dynamic model. However, this analytical form is not always known or easily obtained. Consequently, an alternative way is needed, which is proposed in [1]. Initially, it is necessary to rewrite (1) in vector form

$$\frac{d}{dt}\nabla_{\dot{q}}L = \nabla_q L.$$

Next, we expand the time derivative across the gradient using the chain rule

$$(\nabla_{\dot{q}}\nabla_{\dot{q}}^T L)\ddot{q} + (\nabla_q \nabla_{\dot{q}}^T L)\dot{q} = \nabla_q L.$$

At this point, with a simple matrix inversion, we obtain the expression of the acceleration vector

$$\ddot{q} = (\nabla_{\dot{q}}\nabla_{\dot{q}}^T L)^{-1}[\nabla_q L - (\nabla_q \nabla_{\dot{q}}^T L)\dot{q}]. \tag{2}$$

The (2) specializes in the (3) when the system is a rigid robot.

$$\ddot{q} = M(q)^{-1}[u - (c(q,\dot{q}) + g(q))]. \tag{3}$$

In conclusion, we can see how (2) contains the Lagrangian to the right of the equal and its derivatives. By doing so, thanks to the automatic differentiation, learning the Lagrangian with a neural network we can calculate the acceleration. What is obtained can be fed into an integrator to find the entire dynamics of the system.

3 Robot

In this section, dynamic models of the rigid and elastic 1R robots are derived for both the Euler Lagrange equations and the method proposed in [1], as well as expressing these with state vectors so that they can be processed by integrators.

3.1 Robot rigid model

The model of a rigid robot is derived by assuming that the manipulator works in a vertical plane and that the y axis of the frame is agreeing with the direction of gravity.
The kinetic energy is

$$T = \frac{1}{2}(I + md^2)\dot{q}^2,$$

and consequently, the inertia matrix is

$$M(q) = I + md^2. \tag{4}$$

The Coriolis and centrifugal terms are computed in a classical way

$$C_1(q) = \frac{1}{2}(\frac{dM_1(q)}{dq} + (\frac{dM_1(q)}{dq})^T - \frac{dM(q)}{dq_1}) = 0$$

$$c(q,\dot{q}) = c_1(q,\dot{q}) = \dot{q}^T C_1(q)\dot{q} = 0$$

We are not surprised by the zero value of $c(q,\dot{q})$. In fact, by having one single link, the centrifugal terms that contribute to the joint when it is moving are null.

As a consequence, by having $c(q, \dot{q})$ depends only on the single link of our robot, the result is canonical.

The potential energy is

$$g = \begin{bmatrix} 0 \\ g_0 \\ 0 \end{bmatrix}, \tag{5}$$

$$U_g = -mg_0 dcos(q), \tag{6}$$

$$g(q) = (\frac{dU_g}{dq})^T = mg_0 d \sin(q), \tag{7}$$

Consequently, the model is

$$(I + md^2)\ddot{q} + mg_0 d \sin(q) = 0. \tag{8}$$

Finally, the acceleration obtained solving (8) is

$$\ddot{q} = -\frac{1}{(I + md^2)} mg_0 d \sin(q). \tag{9}$$

3.1.1 Parametric Lagrangian of rigid model

Having computed the dynamic model from an analytical approach to the Lagrangian function, we can immediately make a comparison by observing the result that the application of the analytical Lagrangian function introduced in Section 2

$$L(q, \dot{q}) = T(q, \dot{q}) - U(q) = \frac{1}{2}(I + md^2)\dot{q}^2 + mg_0 dcos(q). \tag{10}$$

From this Lagrangian expression, we derive all the elements needed to apply (2)

$$\nabla_{\dot{q}}^T L = \frac{\partial L}{\partial \dot{q}} = (I + md^2)\dot{q}$$

$$\nabla_q L = -mg_0 d \sin(q)$$

$$\nabla_{\dot{q}} \nabla_{\dot{q}}^T L = \frac{\partial^2 L}{\partial \dot{q}^2} = (I + md^2)$$

$$\nabla_q \nabla_{\dot{q}}^T L = \frac{\partial^2 L}{\partial q \partial \dot{q}} = 0$$

Thus obtaining a result that is shown to be equivalent to that found in equation (9), as expected

$$\ddot{q} = (\nabla_{\dot{q}} \nabla_{\dot{q}}^T L)^{-1} [\nabla_q L - (\nabla_q \nabla_{\dot{q}}^T L)\dot{q}] = -\frac{1}{(I + md^2)} mg_0 d \sin(q).$$

3.1.2 Rigid state space

Equation 8 can be expressed in state-space form by defining the 2×1 state vector

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}.$$

Accordingly, the dynamics of the system can be expressed as

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{1}{I+md^2} m g_0 d \sin(q) \end{bmatrix}.$$

3.2 Robot elastic model

Similar to the rigid case, we derive the dynamic model of the same robot, but with elasticity at the joint. In addition to the considerations made for the rigid robot, based also on what is expressed in [3] and [4], we also assume that:

- The coordinates frames of the motors are placed at their center of mass, as we understand the latter to be uniformly rotating bodies. The coordinate axes can thus be said to be principal axes, resulting in a diagonal inertia matrix.
- The kinetic energy of motors is mainly given by their rotation, so they will have a purely rotary motion.
- The motors are considered after the reduction ($\theta = \theta_m / n_r$).

The kinetic energy of the link (T) and motor(T_m) are respectively:

$$T = \frac{1}{2}(I + md^2)\dot{q}^2 \qquad T_m = \frac{1}{2}, I_m \dot{\theta}^2.$$

The inertia matrices for both cases are thus obtained

$$M(q) = I + md^2, \qquad B_m = I_m.$$

As in the rigid case, it is immediate to see that the term of the Coriolis and Centrifugal forces turns out to be zero, due to the constant nature of the inertia matrix.

$$c(q, \dot{q}) = 0$$

Finally, we derive the potential energies of the link and motor

$$U_m = 0, \qquad U_g = -m g_0 d \cos(q), \qquad U_e = \frac{1}{2} k (q - \theta)^2 \dot{\theta},$$

$$U = U_m + U_g + U_e = -m g_0 d \cos(q) + \frac{1}{2} k (q - \theta)^2,$$

$$g(q) = (\frac{\partial U}{\partial q})^T = m g_0 d \sin(q) + k(q - \theta), \qquad (\frac{\partial U}{\partial \theta})^T = k(\theta - q)\dot{\theta}$$

We thus obtain the dynamic model and the corresponding acceleration expressions

$$\begin{cases} (I + md^2)\ddot{q} + mg_0 d\,\sin(q) + k(q - \theta) = 0 \\ I_m\ddot{\theta} + k(\theta - q) = 0 \end{cases}$$

$$\begin{cases} \ddot{q} = -\frac{1}{I+md^2}(mg_0 d\,\sin(q) + k(q - \theta)) \\ \ddot{q} = \frac{1}{I_m}k(q - \theta) \end{cases} \tag{11}$$

3.2.1 Parametric Lagrangian of elastic model

As we did in the rigid case, we can also see in the elastic context the result of applying the parametric Lagrangian function

$$L(q, \dot{q}) = T(q, \dot{q}) - U(q) = \frac{1}{2}(I + md^2)\dot{q}^2 + \frac{1}{2}I_m\dot{\theta}^2 + mg_0 d\,\cos(q) - \frac{1}{2}\,k\,(q - \theta)^2.$$

From this Lagrangian expression, we derive all the elements needed to apply equation (2)

$$\nabla_{\dot{q}}^T L = \begin{bmatrix} \frac{\partial L}{\partial \dot{q}} \\ \frac{\partial L}{\partial \dot{\theta}} \end{bmatrix} = \begin{bmatrix} (I + md^2)\dot{q} \\ I_m\dot{\theta} \end{bmatrix},$$

$$\nabla_{\dot{q}}^T L = \begin{bmatrix} \frac{\partial L}{\partial q} \\ \frac{\partial L}{\partial \theta} \end{bmatrix} = \begin{bmatrix} -mg_0 d\,\sin(q) - k(q - \theta) \\ k(q - \theta) \end{bmatrix},$$

$$\nabla_{\dot{q}}\nabla_{\dot{q}}^T L = \begin{bmatrix} \frac{\partial L^2}{\partial \dot{q}^2} & \frac{\partial L^2}{\partial \dot{\theta}\partial \dot{q}} \\ \frac{\partial L^2}{\partial \dot{q}\partial \dot{\theta}} & \frac{\partial L^2}{\partial \dot{\theta}^2} \end{bmatrix} = \begin{bmatrix} I + md^2 & 0 \\ 0 & I_m \end{bmatrix},$$

$$\nabla_{q}\nabla_{\dot{q}}^T L = \begin{bmatrix} \frac{\partial L^2}{\partial \dot{q}^2} & \frac{\partial L^2}{\partial \theta\partial \dot{q}} \\ \frac{\partial L^2}{\partial q\partial \dot{\theta}} & \frac{\partial L^2}{\partial \dot{\theta}^2} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

thus obtaining

$$\begin{bmatrix} \ddot{q} \\ \ddot{\theta} \end{bmatrix} = (\nabla_{\dot{q}}\nabla_{\dot{q}}^T L)^{-1}[\nabla_q L - (\nabla_q \nabla_{\dot{q}}^T L)\dot{q}] = \begin{bmatrix} -\frac{1}{I+md^2}(mg_0 d\,\sin(q) + k(q - \theta) \\ \frac{1}{I_m}k(q - \theta) \end{bmatrix}. \tag{12}$$

3.2.2 Elastic state space

Equation shown in 11 can be expressed in state-space form by defining the 4×1 state vector

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} q \\ \theta \\ \dot{q} \\ \dot{\theta} \end{bmatrix}.$$

Accordingly, the dynamics of the system can be expressed as

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ -\frac{1}{I+md^2}(mg_0 d\,\sin(q) + k(q-\theta)) \\ \frac{1}{I_m}k(q-\theta) \end{bmatrix}$$

4 Neural Network Architectures

This section describes the two architectures that will be the subject of our simulations, one classical and one LNN. In particular, the approach that each network has toward the problem of robot dynamics estimation will be analyzed in depth.

4.1 Baseline Neural Network

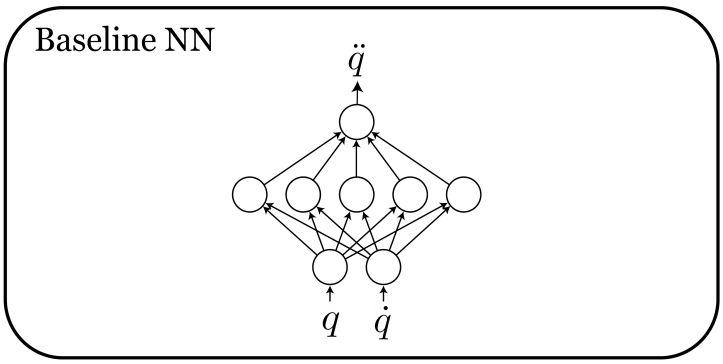


Figure 1: Baseline NN architecture

The first architecture we consider is a classical feedforward neural network, consisting only of fully connected layers.
This network tries to learn the acceleration $\ddot{q}$ directly from the configuration q and the velocity $\dot{q}$, but without considering the Lagrangian and the constraints imposed by it via (2).
We use this network as a baseline against which to compare the LNN approach to the problem.
As for the various activation functions and other network parameters such as the number of layers and the number of neurons, we use the same as those of the LNN, to strengthen the comparison with the baseline.

4.2 Lagrangian Neural Network

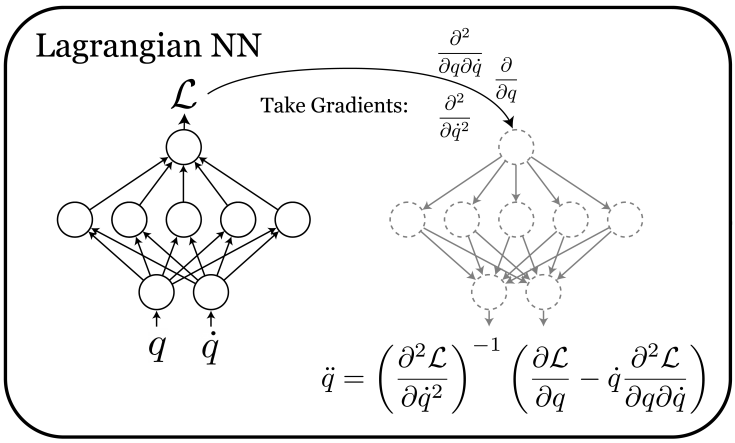


Figure 2: LNN architecture

At least structurally, the LNN turns out to be very similar, if not the same, as the structure of the baseline NN. However, the approach by which prediction is arrived at is completely overturned. In fact, in this case, the goal of the LNN is to predict, or rather model, the parametric Lagrangian function, in a functional programming context. This approach is completely different from that applied by the baseline NN, which, aims immediately to predict the value of the acceleration given the position and velocity inputs. In the case of the LNN we can see that once the parametric Lagrangian function is obtained, the gradients are applied as shown in Section 2. Only after this process, the acceleration is obtained and compared with its ground truth. We can immediately conclude that this process is much closer to the nature of the robot than the baseline NN approach. The acceleration comes from a Lagrangian function, even if a black box of a parametric nature. By doing so, the prediction will benefit all the features, or at least the learned features, that a Lagrangian approach has.

5 Simulations

In this section, we present the simulations carried out to observe the behavior of the two neural network architectures presented in Section 4 on the robot models obtained in Section 3 and compare their results. The first type of simulation is based on observing the behavior of the networks in replicating the robot's dynamics. The second type, on the other hand, presents the same task but with the presence of noise to highlight how robust these architectures are in non-optimal contexts. Both types of tests will compare the true trajectory with the one calculated from the baseline NN and LNN, also showing the error between the ground truth and predictions and highlighting the Root Mean Square Error (RMSE) of the two cases

$$\text{RMSE}(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N}}$$

where y is the ground truth, $\hat{y}$ are the predictions and N is the size of the data.

The above approach will be observed on various types of trajectories and for the elastic case, the evolution of the motor configuration will of course also be taken into account. There will be also a focus on how the networks manage to conserve the total energy of the system. The time steps in the graphs are intended to be in tenths of seconds, as proposed in [1].

5.1 Dataset generation

Starting from 300 initial conditions, based on the equations obtained from Section 3, trajectories were derived over a time interval of 100-time steps with which to compose the dataset. The latter were obtained by the application of the differential equation solver odeint, on the previously exposed data. The entire available dataset is divided into train data and test data, as in the standard neural network training approach. Specifically, train data is 70% and validation data is 30% of the total. Following training, simulations are performed on trajectories generated ad-hoc for the test.

5.2 Rigid case

The choice of the values to assign to the robot parameters is very important since, if they are inconsistent, the obtained results can be flawed and not truthful. Rigid robot parameters are:

- Link length (l).
- Link base radius (r).
- Link CoM distance from the base (d).
- Link mass (m).
- Link inertia (I).

Let us assume that the only link of the rigid robot we are analyzing:

- It is a tall aluminum cylinder.
- Its mass is uniformly distributed.
- Its density is $D_{all} = 2700 \frac{kg}{m^3}$.
- Its inertia is mainly given by its length l.

At this point, we can derive its mass and inertia from the following formulas

$$m = D_{all}V, \qquad V = lr^2\pi, \qquad I = \frac{1}{2}ml^2. \tag{13}$$

After these considerations, we can define the parameters of the rigid robot that we will use in the simulations

Parameter	**Measure**
Length	$0.30\,[m]$
Base radius	$0.01\,[m]$
CoM distance	$0.15\,[m]$
Mass	$0.25\,[kg]$
Inertia	$0.01125\,[kg\,m^2]$

Table 1: Rigid robot parameters

5.2.1 Network setup

As anticipated in Section 4 both networks were set with the same structural parameters. In particular, the following values were chosen: 4 layers, 500 neurons, and Softplus activation function.

In addition, the train of both networks was set for several epochs equal to 10000 and a Mean Absolute Error (MAE) loss, as implemented in [1].

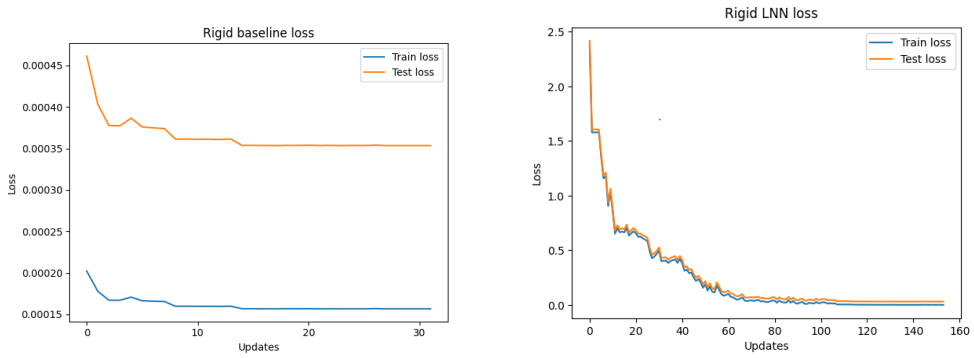
$$\text{MAE}(y, \hat{y}) = \frac{\sum_{i=0}^{N-1} |y_i - \hat{y}_i|}{N}$$

In the previous equation y is the ground truth, $\hat{y}$ are the predictions and N is the size of the past batch of data.

It should be mentioned that the loss has a regularisation mechanism, adding an L2 penalty term to the MSE value to avoid overfitting. The optimizer used is Adam, which applied an adaptive learning rate to increase the effectiveness of the training. Additionally, the update of weights occurred only in particular contexts, such as reaching a threshold of epochs or improving loss. This approach, also recommended in [1] aims to optimize training time.

5.2.2 Results

First, we report the graphics of the test and train losses for the baseline NN and the LNN in 3:



(a) Baseline NN (b) LNN

Figure 3: Test and train losses

As we can see in figure 3, both the test and the train losses decrease for the baseline and the LNN. Anyway, we can see that the LNN has more consistent learning than the baseline NN because of the much more pronounced decrease in loss.
However, to confirm our intuitions, it is necessary to proceed with testing. After the training phase, we experimented with trained models of both networks on about ten different trajectories. We report the most significant results next.

Simulation 1: Trajectory with $q(0) = 1.33\,rad$**,** $\dot{q}(0) = 0.2\,\frac{rad}{s}$ In the first simulation, we compare the behavior of the two networks for the following desired trajectory

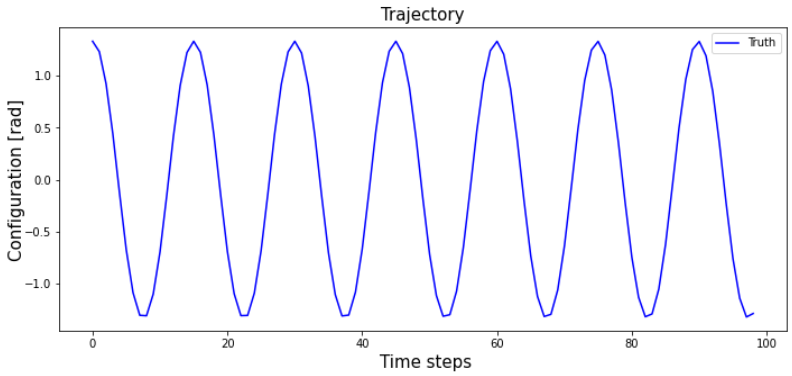
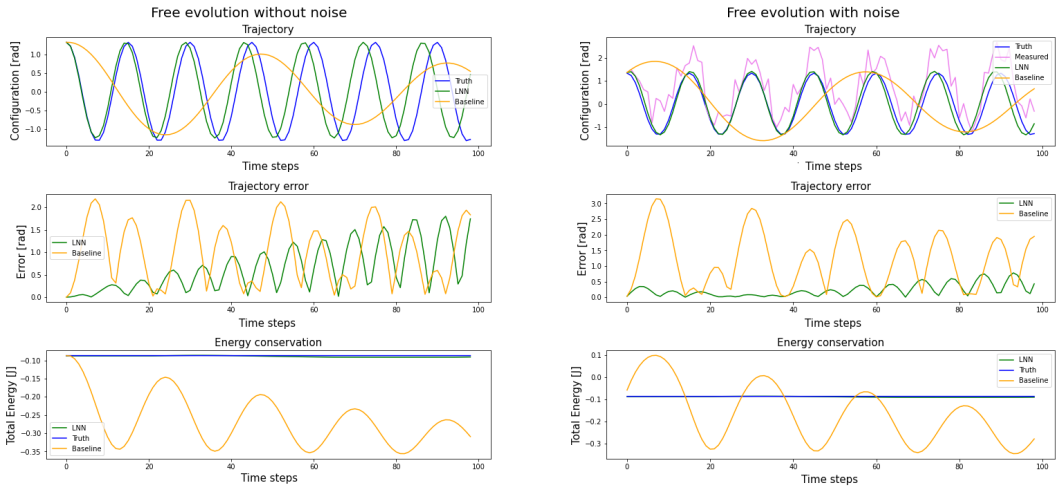


Figure 4: *Simulation 1.* Plot of the desired trajectory for initial conditions $q(0) = 1.33\,rad$, $\dot{q}(0) = 0.2\,\frac{rad}{s}$



(a) Without noise

(b) With noise

Figure 5: *Simulation 1.* Graphs of the trajectory, error, and energy conservation for the NN baseline and LNN. The LNN approach demonstrates better results than the NN baseline.

As we can see from figure 5a, very good behavior of the LNN in both study cases is evident. Starting with the noiseless context, it can be seen that, especially at the initial moments of the trajectory, the LNN returns values that are almost coincident with the true trajectory, however, gaining an advance as time passes. The baseline NN, on the other hand, while guessing at least the sinusoidal trend of the trajectory, is certainly much less correct. Based on what was presented in Section 1, it turns out to be of interest to observe the conservation of total energy. Here we can see how the LNN achieves a very good result, completely overlapping the ground truth and, consequently, indicating full conservation of total energy. This behavior is not present in the baseline NN, as expected.

Network	**No noise**	**Noise**
Baseline NN	$1.2141\,[rad]$	$1.4715\,[rad]$
LNN	$0.8175\,[rad]$	$0.2980\,[rad]$

Table 2: *Simulation 1.* LNN and baseline NN RMSE on the whole desired trajectory for initial conditions $q(0) = 1.33\,rad$, $\dot{q}(0) = 0.2\,\frac{rad}{s}$

So far we have been able to see how, under ideal conditions, we can confirm what we have deduced theoretically. However, to verify the actual robustness of the proposed architectures, we can see in figure 5b how the latter react when the data from which they are to predict the trajectory are subjected to uniform noise, which pollutes the measurement with random samples between $\pm\frac{\pi}{2}$. We can see that the trend we had in the case without noise is also maintained in the case with noise. The LNN succeeds very well in predicting the trajectory, even at the time when the data is plagued by noise. In this case, moreover, the lag behavior as time passes seems to

be less consistent, which is also confirmed by the results in table 2 that show a lower RMSE of the LNN in the noisy case than in the clean case. The goodness of the LNN results is also confirmed in the context of energy conservation. For the baseline NN, the noisy case introduces additional difficulties for this approach, which tends to worsen its results.

Simulation 2: Trajectory with $q(0) = 0\,rad$, $\dot{q}(0) = 4\,\frac{rad}{s}$ We now observe how the two architectures can react at a level of initial speed that is certainly much higher than the previous

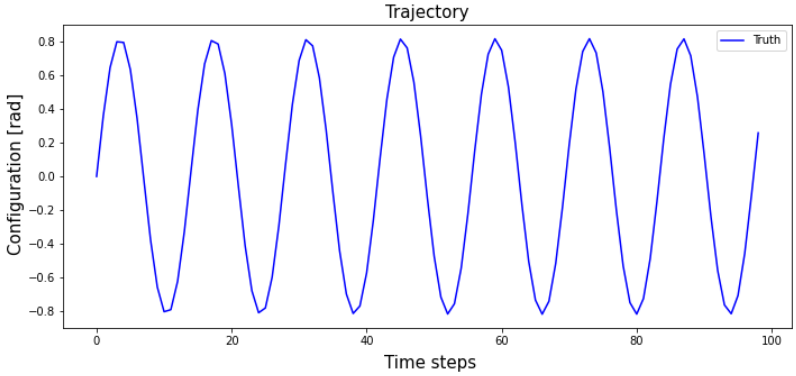
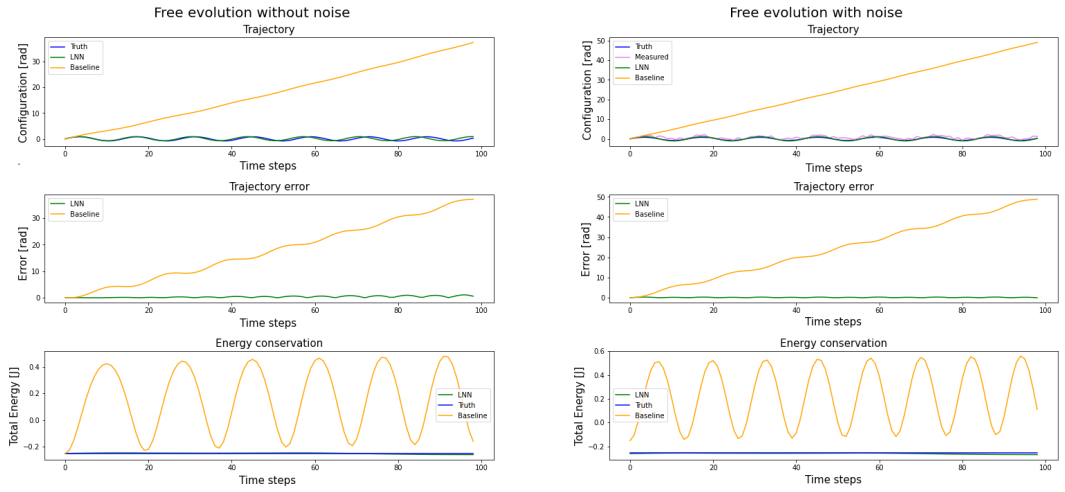


Figure 6: *Simulation 2.* Plot of the desired trajectory for initial conditions $q(0) = 0\,rad$, $\dot{q}(0) = 4\,\frac{rad}{s}$



(a) Without noise

(b) With noise

Figure 7: *Simulation 2.* Graphs of the trajectory, error, and energy conservation for the NN baseline and LNN. For these initial conditions, the baseline NN demonstrates anomalous behavior, while the LNN continues to demonstrate very good properties in application contexts.

The first thing that is highlighted by all the graphs in figure 7 is how in both scenarios of the simulation, without noise and with, the NN baseline acquires a strongly

abnormal behavior, returning a trajectory that has very little good, if any, to it. On the other hand, looking at the LNN results, the difference in performance is obvious and remarkable. Above all, it is interesting to note that LNN again maintains very good total energy conservation behavior.

Network	**No noise**	**Noise**
Baseline NN	$20.8874\,[rad]$	$28.0418\,[rad]$
LNN	$0.4576\,[rad]$	$0.2227\,[rad]$

Table 3: *Simulation 2.* LNN and baseline NN RMSE on the whole desired trajectory for initial conditions $q(0) = 0\,rad$, $\dot{q}(0) = 4\,\frac{rad}{s}$

Also from the RMSE results in table 3, the LNN performs very accurately in both the normal and noisy contexts. Compared with the results of *Simulation 1* we see that for LNN the RMSE index is lower in both study cases. Most likely this is to be associated with the fact that the advance of the trajectory predicted by the LNN occurs later in the execution time.

Simulation 3: Trajectory with $q(0) = 2.66\,rad$**,** $\dot{q}(0) = 0.5\,\frac{rad}{s}$ To conclude our simulations on the rigid case, we decide to try a trajectory in which the robot starts from a much more pronounced position than the previous ones, positioning it almost mirroring its position at $0rad$

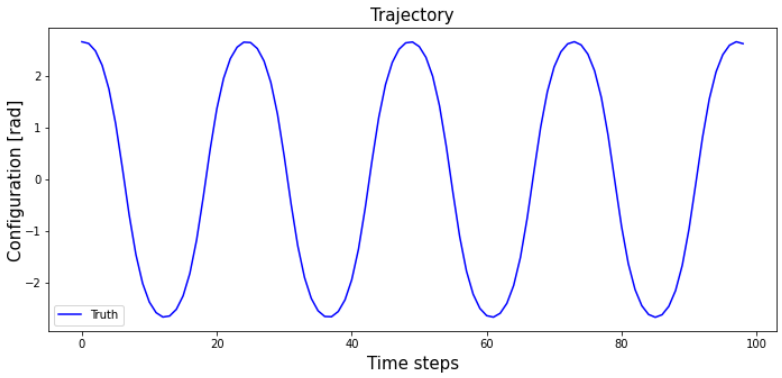
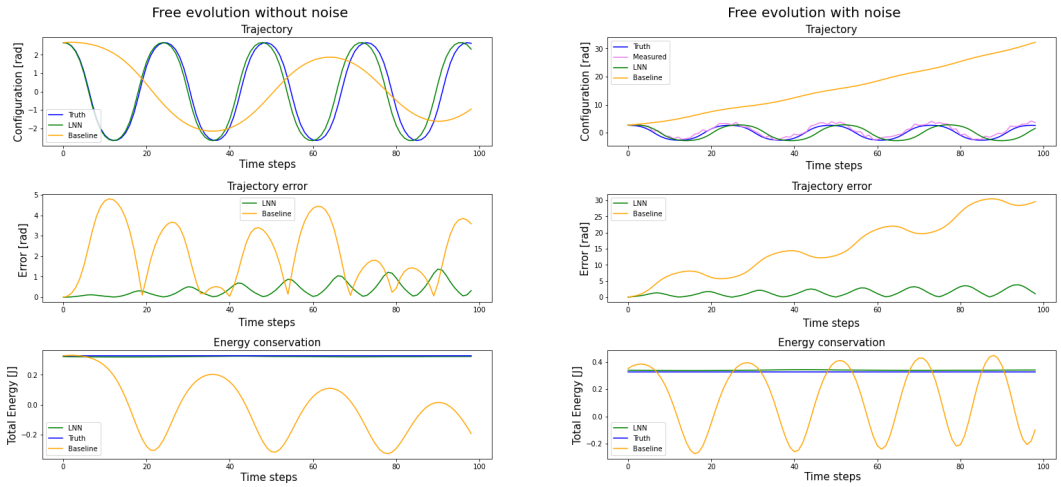


Figure 8: *Simulation 3.* Plot of the desired trajectory for initial conditions $q(0) = 2.66\,rad$, $\dot{q}(0) = 0.5\,\frac{rad}{s}$



(a) Without noise

(b) With noise

Figure 9: *Simulation 3.* Graphs of the trajectory, error, and energy conservation for the NN baseline and LNN. In this case, the introduction of noise leads to difficulties for both architectures..

What we have seen in previous simulations is repeated now. Especially for the noiseless case, we get very similar results to *Simulation 1*, with the NN baseline trajectory not good, and the LNN trajectory starting to lag after a few time steps

Network	**No noise**	**Noise**
Baseline NN	$2.5636\,[rad]$	$18.1152\,[rad]$
LNN	$0.5096\,[rad]$	$1.7531\,[rad]$

Table 4: *Simulation 3.* LNN and baseline NN RMSE on the whole desired trajectory for initial conditions $q(0) = 2.66\,rad$, $\dot{q}(0) = 0.5\,\frac{rad}{s}$

In this last simulation, however, both architectures suffer more from the introduction of noise. With the disturbance of the measurement from which to predict trajectories, the NN baseline again exhibits abnormal behavior as in *Simulation 2*. The LNN is also relatively more distressed than in previous cases. It can be seen from the graph in Figure 9, how from the very beginning of the trajectory, the prediction of the LNN turns out to lag behind the ground truth, with this effect increasing with time. This fact is also confirmed by the values in table 4. We also note that the total energy of the LNN, compared to the previous simulations, undergoes a slight deviation from the actual total energy. However, overall the LNN again maintains better behavior than the baseline NN.

5.3 Elastic case

In the elastic context, the link parameters of the 1R robot remain unchanged, but the motor and the elastic characteristics should be added to the discussion. Thus,

the following additional parameters should be considered for the elastic robot:

- Motor height (h_m).
- Motor base radius (r_m).
- Motor mass (m_m).
- Reduction ratio (n_r).
- Motor inertia (I_m).
- Stiffness constant (I_m).

As for the motor, we will make the following assumptions:

- It is a low steel cylinder.
- Its mass is uniformly distributed.
- Its density is $D_{steel} = 7850 \frac{kg}{m^3}$.
- Its inertia is mainly given by its base radius r_m.

By specializing the formulas in (13) for this case study, we can derive the parameters of the elastic robot that we will use in subsequent simulations, remembering that we consider the motors after reduction, as expressed in Section 3

Parameter	**Measure**
Height	$0.04\,[m]$
Base radius	$0.02\,[m]$
Mass	$0.4\,[kg]$
Reduction ratio	160
Inertia	$0.0128\,[kg\,m^2]$
Stiffness constant	$50\,[Nm/rad]$

Table 5: Elastic robot parameters

The value of the stiffness constant was decided after observing some values in [2] and then slightly adapted to the code implemented from [1].

5.3.1 Network setup

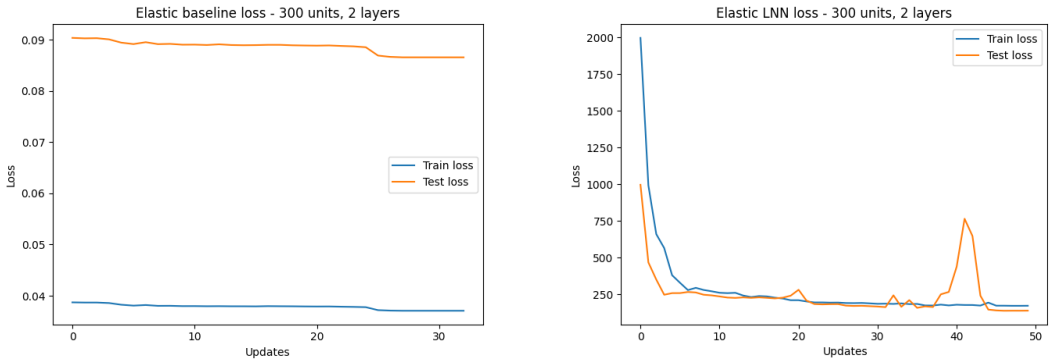
The structural parameters of the networks, in this case, are very similar to those reported in 5.2. However, the case study on which the networks will be tested is a new application, at least for the approach shown in [1]. For this reason, the architectures that will be tested will have three different structures, to see how the response of the different complexity of the architectures to the proposed task varies. Specifically, 3 different configurations are proposed

Configuration	Units	Layers
A	300	2
B	500	4
C	700	5

Table 6: NN parameters for elastic robot simulations

All configurations keep Softplus as the trigger function, following what is indicated in [1]. Again, the characteristics of network training are the same as those presented in Section 5.2.1

5.3.2 Results - Configuration A



(a) Baseline NN

(b) LNN

Figure 10: Test and train losses

Analyzing what the data from the training phase shows us in Figure 10, we can immediately see problems in the learning phase. The baseline NN loss turns out to drop very slowly, while the LNN loss starts from a high value and, just before the end of the training, has a peak in the test loss. After the training phase, which did not give at all encouraging results, we proceed to test the models on various trajectories, showing the most significant results.

Simulation 1: Trajectory with $q(0) = 2.66\,rad$**,** $\dot{q}(0) = 0.5\,\frac{rad}{s}$**,** $\theta(0) = 3.06\,rad$**,** $\dot{\theta}(0) = 1\,\frac{rad}{s}$ In the first elastic case simulation, we analyze the behavior of the two networks when the robot is almost standing vertical

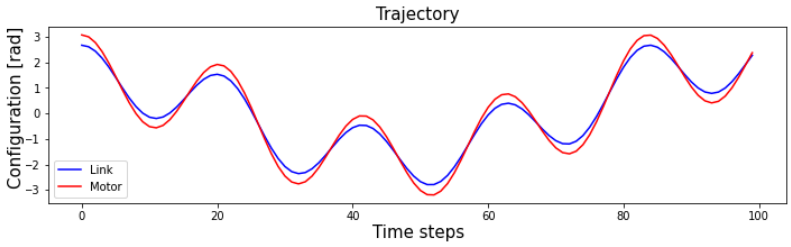
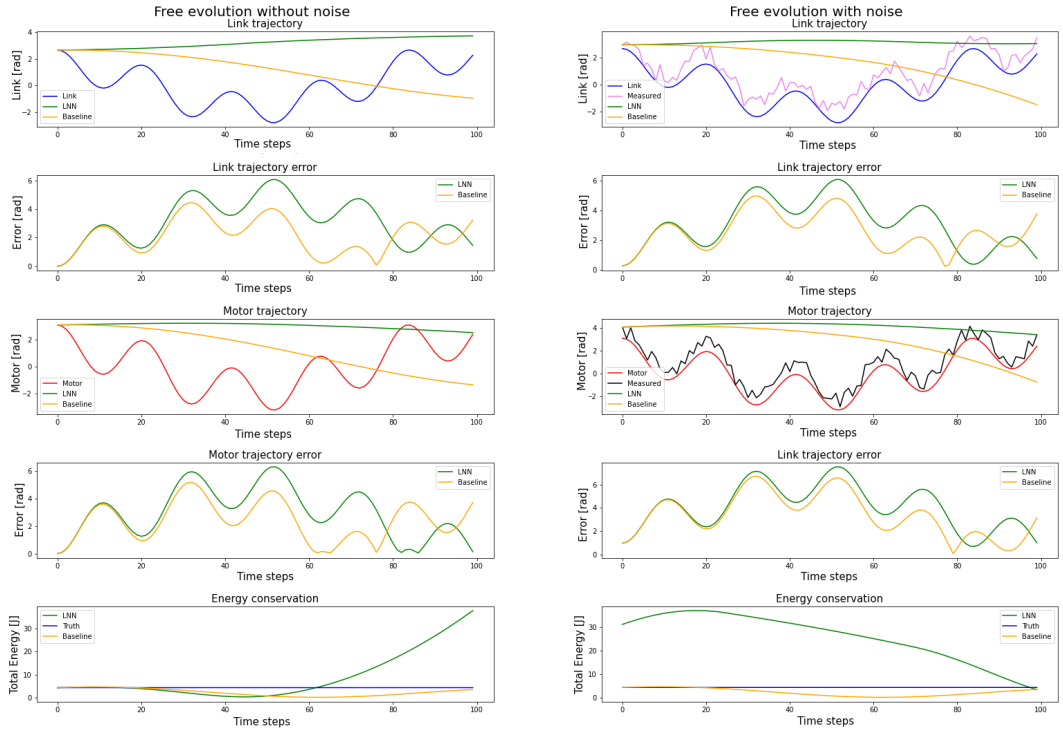


Figure 11: *Simulation 1.* The plot of the desired link and motor trajectory for initial conditions $q(0) = 2.66\,rad$, $\dot{q}(0) = 0.5\,\frac{rad}{s}$, $\theta(0) = 3.06\,rad$, $\dot{\theta}(0) = 1\,\frac{rad}{s}$



(a) Without noise (b) With noise

Figure 12: *Simulation 1.* Graphs of the trajectory, error, and energy conservation for the NN baseline and LNN. In this first simulation, both networks present enormous difficulties in correctly predicting the desired trajectory, both link, and motor.

As well shown by the graphs in figure 12, the first approach to the elastic robot with configuration A does not lead to good results at all. Neither network can correctly predict the evolution of the link and motor trajectories. This appends for both noise and free noise contexts. Moreover, we note how, unlike the rigid case, the LNN in configuration A fails to conserve total energy, reporting strongly abnormal values.

Link		
Network	**No noise**	**Noise**
Baseline NN	2.4337 $[rad]$	2.8176 $[rad]$
LNN	3.5242 $[rad]$	3.4831 $[rad]$

Table 7: *Simulation 1.* LNN and baseline NN RMSE on the whole link desired trajectory for initial conditions $q(0) = 2.66\,rad$, $\dot{q}(0) = 0.5\,\frac{rad}{s}, \theta(0) = 3.06\,rad$, $\dot{\theta}(0) = 1\,\frac{rad}{s}$

Motor		
Network	**No noise**	**Noise**
Baseline NN	2.7626 $[rad]$	3.8201 $[rad]$
LNN	3.4796 $[rad]$	4.4901 $[rad]$

Table 8: *Simulation 1.* LNN and baseline NN RMSE on the whole motor desired trajectory for initial conditions $q(0) = 2.66\,rad$, $\dot{q}(0) = 0.5\,\frac{rad}{s}, \theta(0) = 3.06\,rad$, $\dot{\theta}(0) = 1\,\frac{rad}{s}$

Looking at the RMSE results shown in the tables 7 and 8, paradoxically in such wrong results, the NN baseline achieves values that are better in comparisons with those obtained by LNN.

Simulation 2: Trajectory with $q(0) = 1.33\,rad$**,** $\dot{q}(0) = 1\,\frac{rad}{s}, \theta(0) = 1.51\,rad$**,** $\dot{\theta}(0) = 0.5\,\frac{rad}{s}$ In this second simulation, we try lowering the starting position of the robot

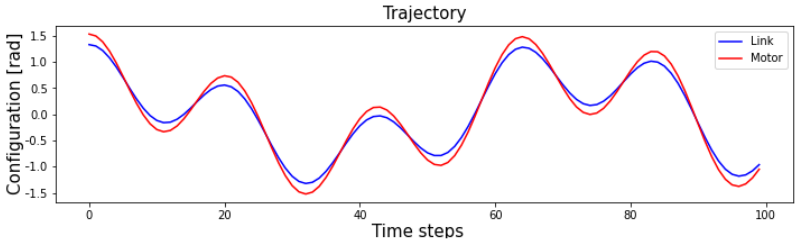
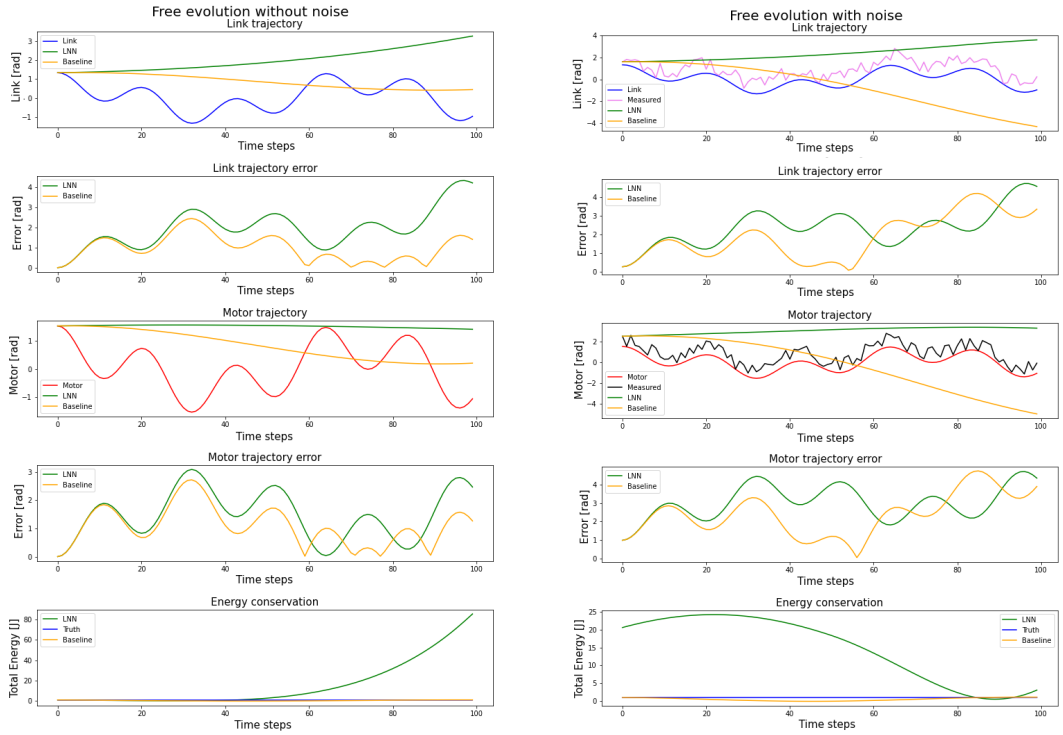


Figure 13: *Simulation 2.* The plot of the desired link and motor trajectory for initial conditions $q(0) = 1.33\,rad$, $\dot{q}(0) = 1\,\frac{rad}{s}, \theta(0) = 1.51\,rad$, $\dot{\theta}(0) = 0.5\,\frac{rad}{s}$



(a) Without noise

(b) With noise

Figure 14: *Simulation 2.* Graphs of the trajectory, error, and energy conservation for the NN baseline and LNN. The second simulation on a different type of trajectory brings no improvement in the overall context, suggesting that the complexity of the networks is perhaps too low to generalize the problem.

The second simulation on a different type of trajectory brings no improvement in the overall context. Indeed, from the point of view of energy conservation, LNN returns even worse results. With the introduction of noise, the quality of the results does not change.

	Link	
Network	**No noise**	**Noise**
Baseline NN	$1.1707\,[rad]$	$2.2049\,[rad]$
LNN	$2.1816\,[rad]$	$2.5608\,[rad]$

Table 9: *Simulation 2.* LNN and baseline NN RMSE on the whole link desired trajectory for initial conditions $q(0) = 1.33\,rad$, $\dot{q}(0) = 1\,\frac{rad}{s}$, $\theta(0) = 1.51\,rad$, $\dot{\theta}(0) = 0.5\,\frac{rad}{s}$

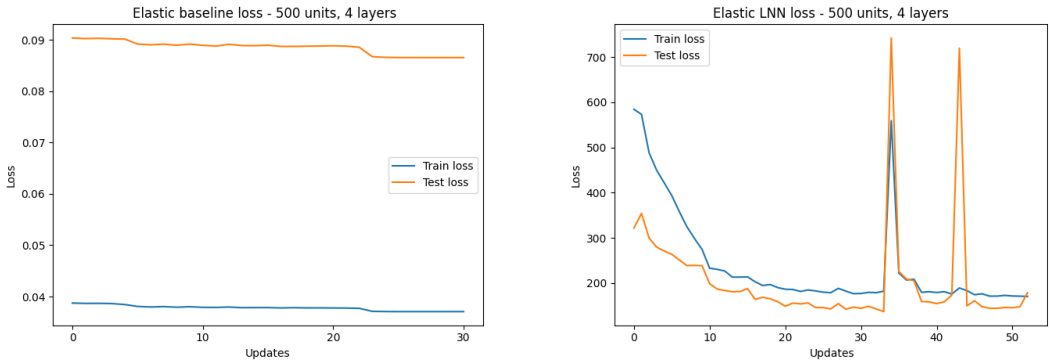
Motor		
Network	**No noise**	**Noise**
Baseline NN	$1.2734\,[rad]$	$2.6318\,[rad]$
LNN	$1.7103\,[rad]$	$3.1560\,[rad]$

Table 10: *Simulation 2.* LNN and baseline NN RMSE on the whole motor desired trajectory for initial conditions $q(0) = 1.33\,rad$, $\dot{q}(0) = 1\,\frac{rad}{s}$, $\theta(0) = 1.51\,rad$, $\dot{\theta}(0) = 0.5\,\frac{rad}{s}$

From the RMSE tables, the values are slightly better than in the previous simulation, but still not sufficient. Moreover, again from the tables 9 and 10, we can see how the introduction of noise discretely increases the error.

These first two simulations on the elastic case show the difficulty of the two types of networks, but analyzing the results obtained suggests, most likely, that the complexity of the networks is perhaps too low to generalize the problem, as from the various indices available, we can see that both the baseline NN and LNN fail to capture important features from the data.

5.3.3 Results - Configuration B



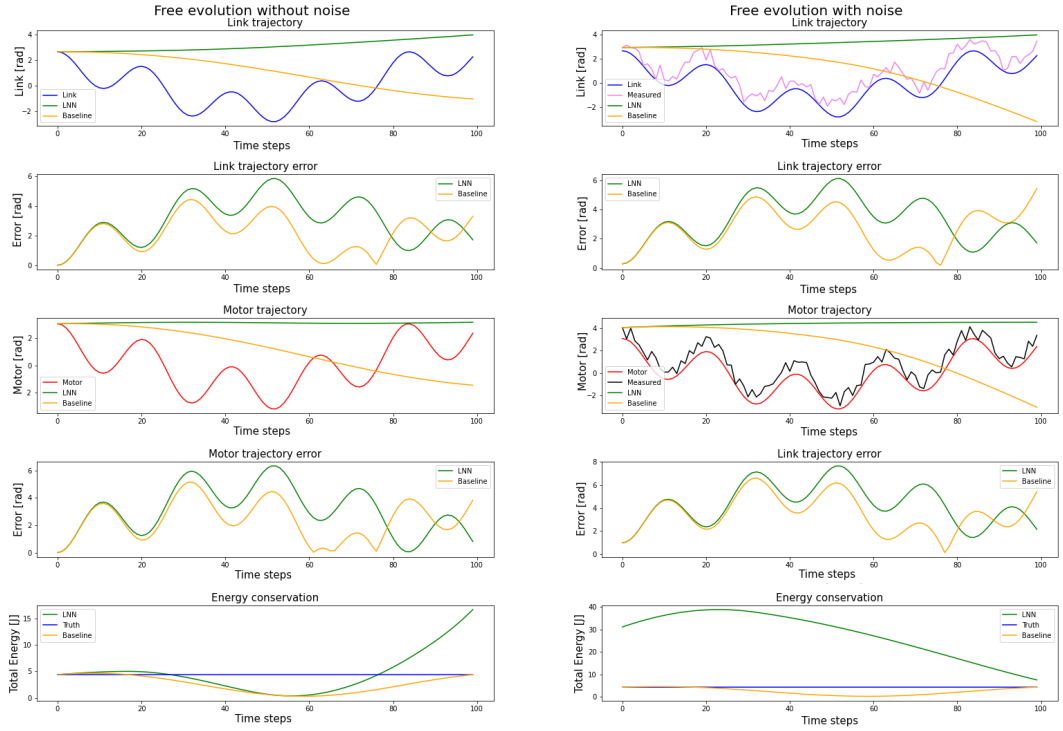
(a) Baseline NN

(b) LNN

Figure 15: Test and train losses

The increase in network complexity brings the first changes already during the network training phase, as we can see in figure 15. Although the baseline NN maintains very similar learning compared to the configuration of the previous case, with a loss that hardly ever decreases, the LNN undergoes important changes. Both the initial and final losses are better than the previous case, although they still maintain a high value. Another sign that should not be underestimated is the presence of LNN loss peaks toward the end of the training session. To carefully evaluate the changes that have occurred, we set the simulations to the same trajectories as in the previous case.

Simulation 3: Trajectory with $q(0) = 2.66\,rad$, $\dot{q}(0) = 0.5\,\frac{rad}{s}$, $\theta(0) = 3.06\,rad$, $\dot{\theta}(0) = 1\,\frac{rad}{s}$ The trajectory is the same as shown in figure 11.



(a) Without noise

(b) With noise

Figure 16: *Simulation 3.* Graphs of the trajectory, error, and energy conservation for the NN baseline and LNN. Despite the increased complexity of the networks, the changes that have taken place are still too weak to speak of good results.

Looking at the graphs of 16, the general situation is almost unchanged compared to the previous configuration, although the LNN seems to have had a change in its performance compared to the NN baseline, which remains more or less the same. The error intensity of the LNN is slightly lower, for both the link and motor trajectories. This trend seems to be confirmed even when noise is introduced, naturally without any noticeable results.

Link		
Network	**No noise**	**Noise**
Baseline NN	$2.4253\,[rad]$	$2.9910\,[rad]$
LNN	$3.4193\,[rad]$	$3.6460\,[rad]$

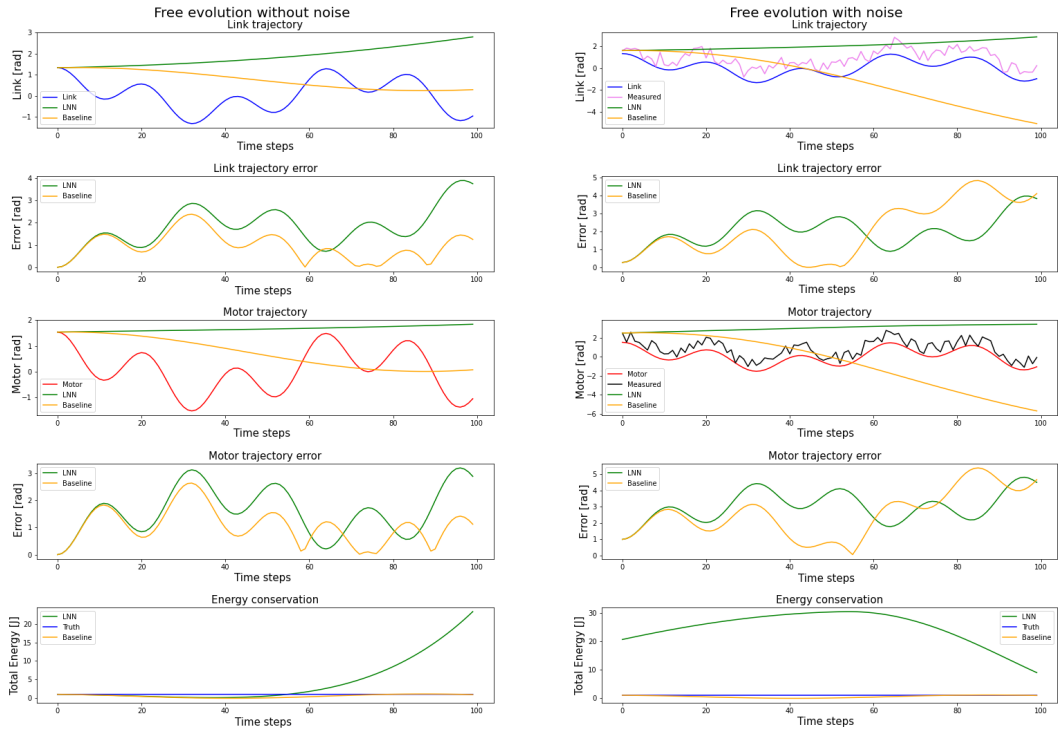
Table 11: *Simulation 3.* LNN and baseline NN RMSE on the whole link desired trajectory for initial conditions $q(0) = 2.66\,rad$, $\dot{q}(0) = 0.5\,\frac{rad}{s}$, $\theta(0) = 3.06\,rad$, $\dot{\theta}(0) = 1\,\frac{rad}{s}$

Motor		
Network	**No noise**	**Noise**
Baseline NN	2.7547 $[rad]$	3.8176 $[rad]$
LNN	3.4556 $[rad]$	4.3988 $[rad]$

Table 12: *Simulation 3.* LNN and baseline NN RMSE on the whole motor desired trajectory for initial conditions $q(0) = 2.66\,rad$, $\dot{q}(0) = 0.5\,\frac{rad}{s}, \theta(0) = 3.06\,rad$, $\dot{\theta}(0) = 1\,\frac{rad}{s}$

Even from the RMSE values, LNN seems to bring very slight signs of change. Comparing these results with those of the previous configuration table, a slight improvement in the statistics can be seen. However, once again, the introduction of noise worsens these values, showing the low robustness of this architecture. The NN baseline, on the other hand, keeps them relatively unchanged.

Simulation 4: Trajectory with $q(0) = 1.33\,rad$**,** $\dot{q}(0) = 1\,\frac{rad}{s}, \theta(0) = 1.51\,rad$**,** $\dot{\theta}(0) = 0.5\,\frac{rad}{s}$ Also, in this case, the trajectory is the same as shown in figure 13.



(a) Without noise

(b) With noise

Figure 17: *Simulation 4.* Graphs of the trajectory, error, and energy conservation for the NN baseline and LNN. Again, we see a very similar situation to the simulation before this one the results show a trend towards improvement, even if minimal.

So far, from the results shown in 17, we can begin to confirm the insights gained from the previous simulation. The results are of course not good, but the error

values are dropping slightly as the complexity of the networks increases. It should still be noted that LNN still fails to generalize the conservation of total energy.

Link		
Network	**No noise**	**Noise**
Baseline NN	$1.1247\,[rad]$	$2.5380\,[rad]$
LNN	$2.0212\,[rad]$	$2.1989\,[rad]$

Table 13: *Simulation 4.* LNN and baseline NN RMSE on the whole link desired trajectory for initial conditions $q(0) = 1.33\,rad$, $\dot{q}(0) = 1\,\frac{rad}{s}$, $\theta(0) = 1.51\,rad$, $\dot{\theta}(0) = 0.5\,\frac{rad}{s}$

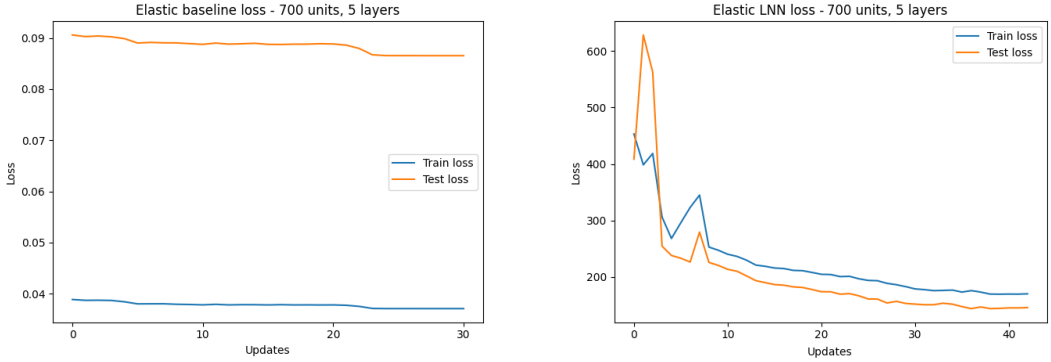
Motor		
Network	**No noise**	**Noise**
Baseline NN	$1.2359\,[rad]$	$2.9295\,[rad]$
LNN	$1.6357\,[rad]$	$3.1532\,[rad]$

Table 14: *Simulation 4.* LNN and baseline NN RMSE on the whole motor desired trajectory for initial conditions $q(0) = 1.33\,rad$, $\dot{q}(0) = 1\,\frac{rad}{s}$, $\theta(0) = 1.51\,rad$, $\dot{\theta}(0) = 0.5\,\frac{rad}{s}$

What was presented in the previous paragraph is underlined by the values of 13 and 14. As far as LNN is concerned, however, it should be noted that these changes are on the order of tens of units for the RMSEE values, which is still too small an improvement. On the other hand, the NN baseline oscillates at values very similar to previous applications.

At the end of this fourth simulation, it could be seen that the increase in complexity has somewhat modified the performance of the networks, positively, especially the LNN, which seems to have benefited the most from the changes. However, these improvements are still very low, so in the last two simulations, we propose we will try to increase the width and depth of the networks even more.

5.3.4 Results - Configuration C

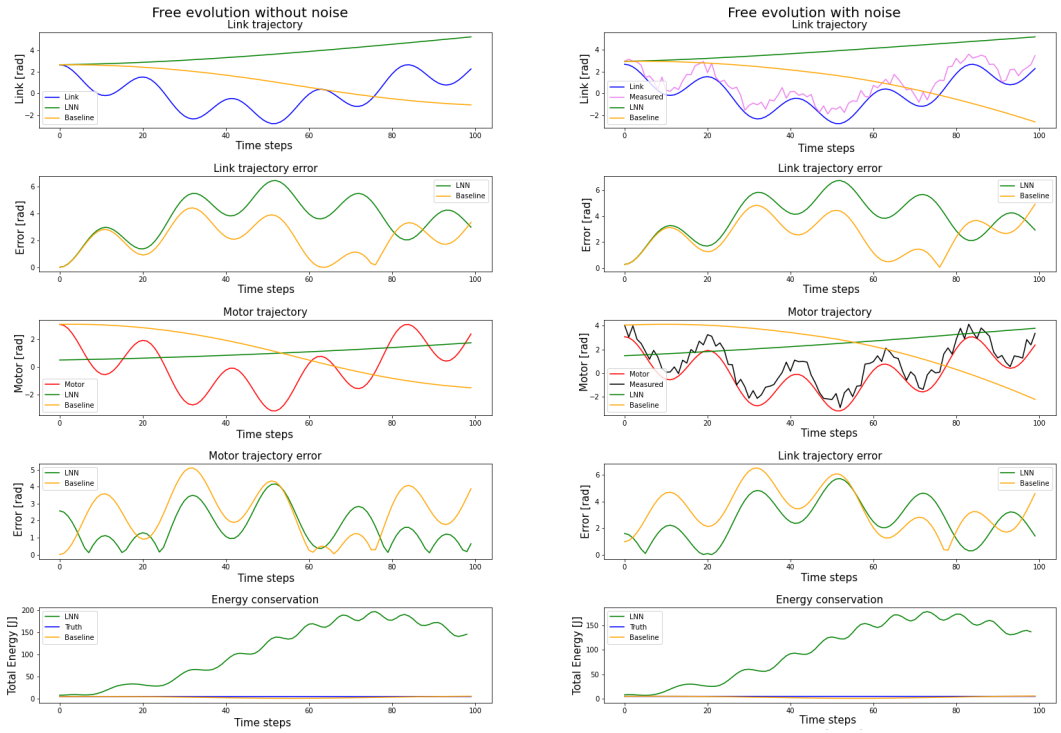


(a) Baseline NN (b) LNN

Figure 18: Test and train losses

The further step of increasing the complexity of the networks brings back interesting results in the LNN train. In fact, from the figure 18, following some initial oscillations of the loss, which we can define as canonical in a neural network training context, for the first time since the beginning of the simulations on the elastic case, the loss falls gradually and without peaks, although maintaining a high final value. Overall, this is certainly a positive indicator in the context of the NN training. The NN baseline, on the other hand, as it had begun to show in the previous setup, maintains its loss trend without any great drops. One thing to note is that, for the LNN in particular, the training time has increased dramatically with the new configuration imposed on the networks. Also in these simulations, to maintain a term of comparison, we will test the trained networks on the trajectories already presented.

Simulation 5: Trajectory with $q(0) = 2.66\,rad$**,** $\dot{q}(0) = 0.5\,\frac{rad}{s}$**,** $\theta(0) = 3.06\,rad$**,** $\dot{\theta}(0) = 1\,\frac{rad}{s}$ The trajectory for this simulation is also similar to that in figure 11.



(a) Without noise (b) With noise

Figure 19: *Simulation 5.* Graphs of the trajectory, error, and energy conservation for the NN baseline and LNN. The slight effects of increased network complexity continue, in particular for the first time the total energy of the LNN does not diverge.

The analysis of the training data only partially confirms the intuition of the improved situation. Once again, neither architecture was able to correctly predict the trajectory of the link and motor. However, for the first time since the elastic case simulations began, the total energy of the LNN does not diverge, remaining limited, albeit at an abnormal value compared to ground truth. The situation just described is also confirmed in the case of the introduction of noise.

	Link	
Network	**No noise**	**Noise**
Baseline NN	$2.4173\,[rad]$	$2.8553\,[rad]$
LNN	$3.0199\,[rad]$	$3.1879\,[rad]$

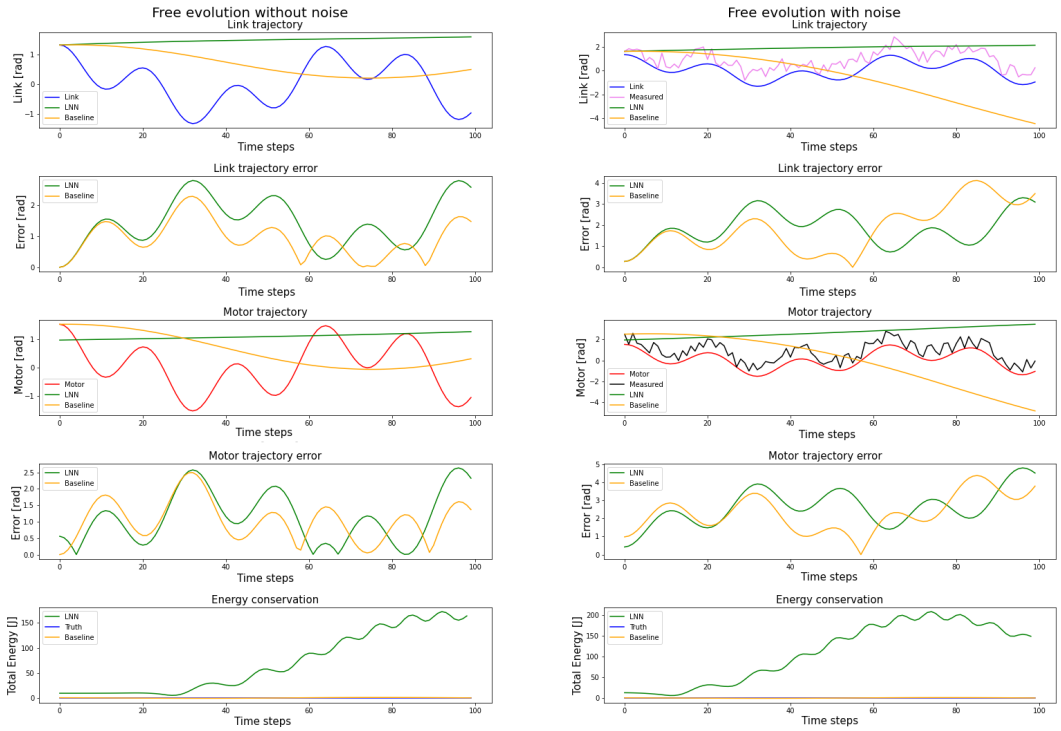
Table 15: *Simulation 5.* LNN and baseline NN RMSE on the whole link desired trajectory for initial conditions $q(0) = 2.66\,rad$, $\dot{q}(0) = 0.5\,\frac{rad}{s}$, $\theta(0) = 3.06\,rad$, $\dot{\theta}(0) = 1\,\frac{rad}{s}$

Motor		
Network	**No noise**	**Noise**
Baseline NN	$2.7524\,[rad]$	$3.6792\,[rad]$
LNN	$1.9237\,[rad]$	$3.0289\,[rad]$

Table 16: *Simulation 5.* LNN and baseline NN RMSE on the whole motor desired trajectory for initial conditions $q(0) = 2.66\,rad$, $\dot{q}(0) = 0.5\,\frac{rad}{s}, \theta(0) = 3.06\,rad$, $\dot{\theta}(0) = 1\,\frac{rad}{s}$

As far as the RMSE values are concerned, the slow decrease of this type of index by LNN continues. It once again decreases by a few tenths of a value. In particular, however, we can at this point definitively confirm the very low robustness of the LNN model for the elastic case in the face of signals with noise. We note above all how in 16 the corresponding RMSE value increases considerably in correspondence with noise. Moreover, as we expected, the NN baseline remains, in general, in the range of values of the previous simulations.

Simulation 6: Trajectory with $q(0) = 1.33\,rad$**,** $\dot{q}(0) = 1\,\frac{rad}{s}, \theta(0) = 1.51\,rad$**,** $\dot{\theta}(0) = 0.5\,\frac{rad}{s}$ Finally, this trajectory is also the same as in the figure 13.



(a) Without noise

(b) With noise

Figure 20: *Simulation 6.* Graphs of the trajectory, error, and energy conservation for the NN baseline and LNN. The trend of improvements remains very similar to the previous case. Energy conservation by LNN also changes.

In this other type of trajectory, we see results in agreement with what has been achieved so far with this configuration of networks. Unfortunately, it must be emphasized that for the umpteenth time in the elastic case, both the LNN and the baseline NN fail to correctly predict the trajectory of the link and the motor, even partially. The major change, as in the previous trajectory, seems to be in the conservation of total energy by the LNN, which seems to be settling without diverging.

Link		
Network	**No noise**	**Noise**
Baseline NN	$1.1221\,[rad]$	$2.1565\,[rad]$
LNN	$1.6345\,[rad]$	$2.0139\,[rad]$

Table 17: *Simulation 6.* LNN and baseline NN RMSE on the whole link desired trajectory for initial conditions $q(0) = 1.33\,rad$, $\dot{q}(0) = 1\,\frac{rad}{s}$, $\theta(0) = 1.51\,rad$, $\dot{\theta}(0) = 0.5\,\frac{rad}{s}$

Motor		
Network	**No noise**	**Noise**
Baseline NN	$1.2009\,[rad]$	$2.5066\,[rad]$
LNN	$1.3689\,[rad]$	$2.8148\,[rad]$

Table 18: *Simulation 6.* LNN and baseline NN RMSE on the whole motor desired trajectory for initial conditions $q(0) = 1.33\,rad$, $\dot{q}(0) = 1\,\frac{rad}{s}$, $\theta(0) = 1.51\,rad$, $\dot{\theta}(0) = 0.5\,\frac{rad}{s}$

The trend presented is also confirmed in the RMSEE values of the tables 17 and 18. Once again, the improvement in these indices for LNN is in the order of tens of units. This improvement is lost when noise is introduced, although this deterioration is relatively lower than the same value from configurations A and B. From these considerations, we can definitively confirm the weakness against the noise of the LNN applied in the elastic case. Finally, the baseline NN maintains its values in the range of the previous cases.

At the end of the simulations of the elastic case, we can see that the intuition of gradually increasing the complexity of the network has paid off somewhat in terms of performance, although the latter remains not at all good in all tests. For the LNN in particular, this increase in complexity induces a very slow decrease in error, suggesting that perhaps a very drastic increase in the complexity of the network would finally lead to satisfactory and sufficient results. However, given the training time of configuration C, our most complex network, this drastic increase in the depth and breadth of the network would have required high computational resources, requiring suitable tools to support very heavy training.

6 Conclusions

At the end of this project, we can observe the following concepts. First of all, at least in the rigid case, it is well demonstrated how the identification of the robot's dynamic model is better using the LNN approach, both in terms of trajectory error and energy conservation. Unfortunately, the good results obtained in the rigid case were not confirmed in the elastic case. However, important and interesting information can be extracted from the simulations of this case. It can be seen from the data obtained that as the complexity of the network increases, the outputs of the Lagrangian seem to change slightly for the better. As we can see, the case for configuration C of the LNN is the only one with non-divergent total energy. This leads us to think that by increasing the complexity of the network, in particular its depth and width, even better results could be achieved. Along with this, one could try to increase the number of initial conditions on which the network is trained. However, these two assumptions for improving the network's performance would presuppose high computational resources, so these insights could only be tested in suitable laboratory settings. A final insight that may arise from our study is that the LNN approach proposed in [1] bases its experiments on totally unitary configurations, a concept that we rejected in preference to a configuration as close to reality as possible.

References

[1] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

[2] Andrea Giusti, Jorn Malzahn, Nikos Tsagarakis, and Matthias Althoff. On the combined inverse-dynamics/passivity-based control of elastic-joint robots. *IEEE Transactions on Robotics*, PP:1–11, 08 2018.

[3] M. W. Spong. Modeling and Control of Elastic Joint Robots. *Journal of Dynamic Systems, Measurement, and Control*, 109(4):310–318, 12 1987.

[4] P. Tomei. A simple pd controller for robots with elastic joints. *IEEE Transactions on Automatic Control*, 36(10):1208–1213, 1991.