[1]

Optimal control via Deep Neural Networks

A.Hanggoro, Y. Zahidi, A.Ismail

***Abstract*— Model predictive control is an advanced method to control the dynamics of a system while satisfying a certain set of constraints. The method itself is gaining more and more popularity in all sorts of industries ranging from chemical plants and oil refineries where they have been used since the 80s, to power plants, and more generally power systems.**

Deep neural networks are neural networks with more than one layer. They are considered to be universal function approximators, which would make them perfectly suited for data driven control applications.

Although they have been around since since the 70's, lack of data has made it impossible in practice to showcase their real abilities until the last two decades where massive automation and connectivity have made it possible to generate a massive amount of data.

In this paper we show how it is possible to simulate an mpc optimized control offline on an HVAC system connected to a grid, a battery and a solar panel (the system is introduced in the introduction) and train a deep neural network (6 layers, 50 units each layer) in order to achieve MPC without explicit control computation or instruction, but rather exclusively the data generated by the offline simulation.

***Index Terms*— Model predictive control, Optimal control, Deep Neural Networks, Optimization, Deep Learning, HVAC, Power Electronics, Power Systems, Power Control, fuzzy logic.**

I. Introduction

The cost of computing device is decreasing. This opens new possibilities to use modern control system in a wide range of applications. The modern control methods such as Model Predictive Control (MPC) has gained popularity because of its ability to control the thing under certain conditions. Parallel to the MPC Control, there is another popular control approach based on Neural network method such as Deep neural network.

In this Project we would like to implement both of the control method (MPC and Deep Learning) to control the room temperature. The system that we would like to control can be described as follows:

There is a room that the temperature needs to be controlled so that it stay at a specific range of temperature. The Heating, ventilation, and air conditioning (HVAC) system is responsible as an actuator to regulate the system through heating and cooling. The energy which is needed by HVAC to do the control action is supplied from grid power, battery, and solar panel.

The challenge that we face in this project is whether if we can control the room temperature under specific constraints, but also maximize the energy that we can sell to the grid by calculating the amount of energy generated by solar panel in combine with the optimal control action.

II. Proposed system solution

The MPC-Controller has a special ability in comparison with a classical control method like PID-Controller. Because the MPC-Controller is based on optimization, it can control the system within specified constraints such as physical constraints and give the optimal output.

This control method is suitable to use for this application. where the system output such as the control signal to regulate the power used to energize the HVAC. The known or measured variable such as the states, room temperature, and the disturbances are used as the control input. Within several constraints like the input constraints or the mixed constraints, we calculated the optimal control output, which is the desired power needed for the HVAC which also at the end is the optimal solution for the goal which is maximizing the energy sold to the grid.

Another approach that we use is to use deep learning (DL) method. This method is basically a program that predicts the specific output based on what it learn. The MPC method and DL method work closely. We use the optimal output of the MPC to feed the DL controller so that it can learn offline (offline means it do the learning not in operating condition). The way we teach the DL controller is first we run as many loops with an MPC controller with different initial condition (around 10.000 data points) and then we teach the DL controller with 8000 data points and let another 2000 as a test. The input that is needed to teach the DL controller is with room temperature ,the battery condition, and the disturbances.

III. Model Predictive Control

A. MPC Introduction

MPC control is used to control multivariable control. It uses an explicit model and handle constraints. The Model predictive control based on iterative, finite-horizon optimization of the plant. It takes the states at time t and calculate next n steps called horizon to achieve the best optimal control output. This step will be executed again for the next iteration.

To use the MPC we have to formulate the MPC problem. the MPC problem can be formulated as within following

[1].

information:

1. The cost function
2. The constraints
3. The system dynamic

B. Implementation

In this project we have several information that we need to formulate our MPC problem.

$$\text{minimize } (x,u) \quad \sum_{k=0}^{N-1} (P_{grid}^{k} + \gamma(E_{bat}^{k} - E_{bat}^{ref})^2) \tag{1a}$$

subject to

$$\chi_{lb} \leq \chi_k \leq \chi_{ub} \tag{1b}$$

$$U_{lb} \leq U_k \leq U_b \tag{1c}$$

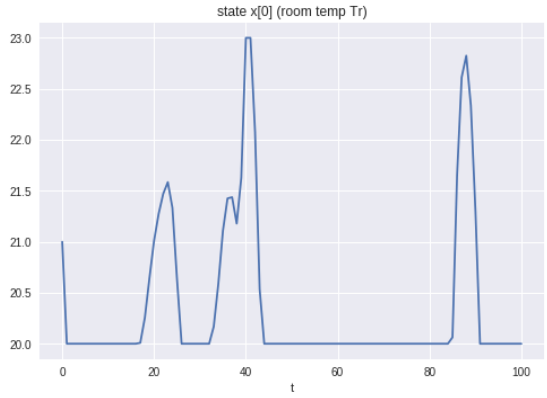
$$m_{lb} \leq Du_k + Gd_k \leq m_{ub} \tag{1d}$$

$$\chi_{k+1} = A\chi_k + Bu_k + Ed_k \tag{1d}$$

This is the MPC problem description for this project. the cost function (1a) that we have to minimize describe about the energy usage from the grid and battery which is mean we would like to minimise the energy usage from the grid. The states x which consist of two variables (room Temperature and battery energy) have several constraints which is should be around 20 Degree to 23 Degree for the temperature and the energy battery must be between 0 and 200000. Another thing we have a mixed constraints D and G which describe the physical constraints.

The Matrix A, B, E for the states is given as well as for matrices D and G. We now have to parameterize the gamma and the battery reference. To solve the optimization problem, we use the tools named casadi. Casadi is a python library that is useful to solve optimization problems.

With the initial input of following:
Room temperature = 21 Degree
Battery energy = 150000
Battery reference = 50000
Gamma γ = 0.0523
We plot the result as following:



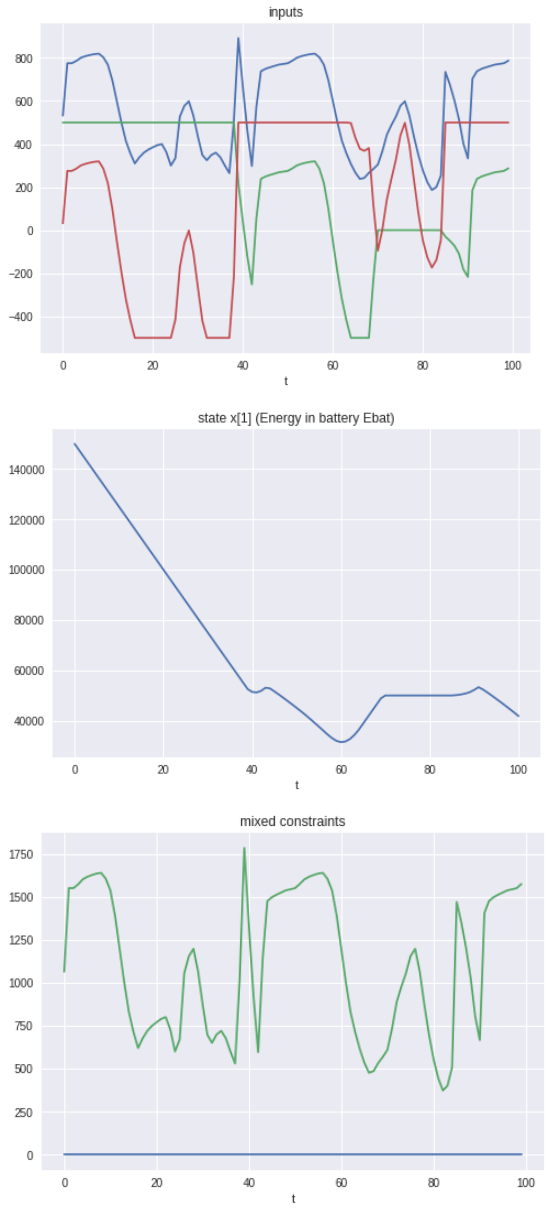


Fig 1. The results of the MPC solution for this project

The program produce an optimal output control that is used to feed the deep learning controller. The next step is to run as many loop with different initial input to produce a different kind of condition (approx 10.000 data point) so that the deep learning controler have sufficient amount of data to learn. For learning case, we choose following parameter:

Battery reference = 50000
Gamma γ = 0.043

IV. Deep learning for optimal control

A. Introduction

Taking into consideration the general scheme of an optimally controlled system, just as it has been shown in the previous section, we realize there are multiple ways control can benefit from implementing a deep neural network.

Although there is multiple ways a Deep Neural Network can help us achieve the optimal control of our system's dynamics, there's mainly 3 ways it can do so :

1. Predict the dynamics of the system, which can be really useful if the system is highly non-linear, or subject to important disturbances that we wish to predict.
   In that specific case, it can be used with an MPC controller or any controller for that matter.
2. Predict the input that an MPC controller would generate for any measured state and disturbance. This would mean that the neural network would be able to literally replace the MPC controller and interact directly with the system in order to optimally control its inputs.
   The training of the DNN would be achieved offline, on the data generated by a simulated or real-time MPC controller.
   However, the implementation of the DL based controller can be achieved online.
3. The third possible way is a mix of the two previous ways, where the trained DL based controller would interact with a second DNN that is trying to approximate the dynamics of the system (including the disturbances), solving thereby the need to have a mathematical description of the model.

B. Our Approach : The Deep Learning based controller

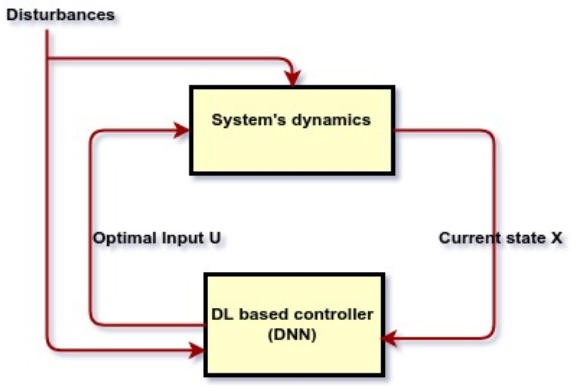
Our approach is the second one, where a feed-forward Deep Neural Network is trained on data that is generated during a simulation of the MPC controlled system.

The choice of a simple feed-forward neural network that takes as inputs the initial state of a horizon window and the N consecutive disturbances of that same horizon window is justified because our system equation can be recursively developed to prove that the optimal input of the system doesn't depend on the previous states, but rather only the initial state of the horizon and the N future disturbances.

In other words, if the MPC controller doesn't require knowledge of the previous states to generate the optimal output, the Neural Network only has to train on the initial state of a horizon and the N future disturbances with the optimal inputs generated by the MPC controller during offline simulation as labels (we are dealing with supervised learning here)

To achieve the training and the testing of the so-called DL based controller, 3 steps have been taken :

1. Simulation of MPC with different allowed combinations of the initial states (2 states) : the results are stored a csv data file containing for each datapoint what will serve as input (initial state and N future disturbances) to the DNN and the labels (optimal outputs computed by the MPC controller simulated offline)
2. Training on training data and evaluation on test data of the feedforward Deep Neural Network to assess which architecture displays the best accuracy-training time compromise (we chose 6 deep layers with 50 units each). The trained model is then saved as a .h5 file for later use as a DL based controller.
3. Simulation of the DL controlled system (simply using the system equation) by connecting the simulated system with the DL controller : current state and N future disturbances are fed to the DNN controller and the DL based controller makes a prediction of what it 'thinks' the optimal output is. The optimal output is fed back to the system as th below figure shows.

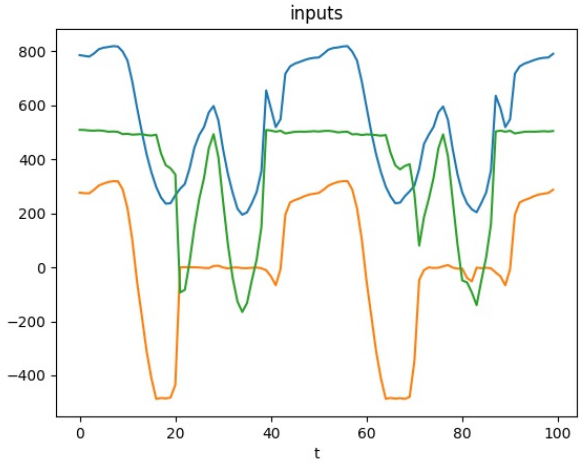


C. Results

1. Quantitative evaluation on test data

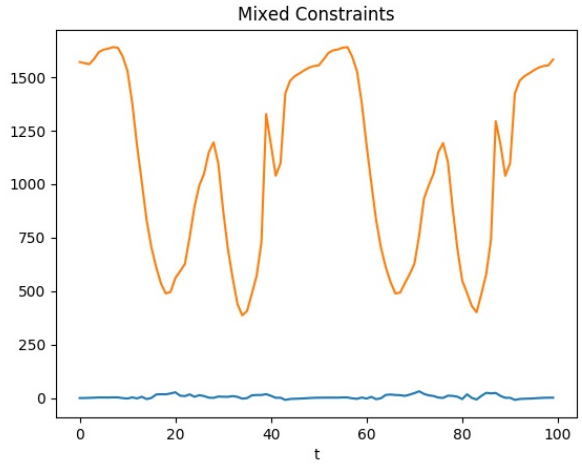
The evaluation of our trained model's accuracy is first assessed through the results of the training results (we stop when a mean average error of about 7 is reached) then through the model.evaluate method provided by the framework Keras :
-Accuracy : 99.35 %
- Mean Absolute Error (MAE) : 7.31
- Mean Squared Error : 288.9

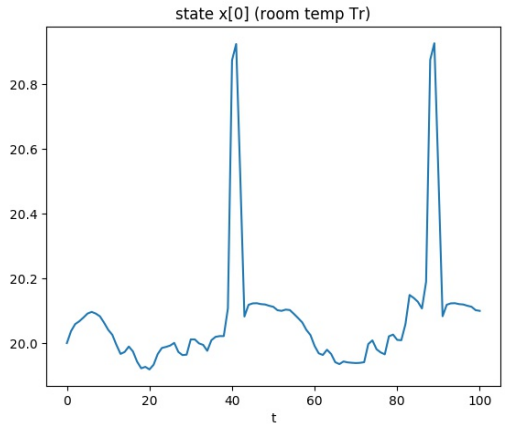
Although the final purpose is to simulate the DL based controller (the trained DNN so to speak) working in conjunction with a function that simulates the dynamics of the system through the system equation x+ = A.x + B.u + E.d and therefore the testing doesn't necessarily mean the DL based controller is working properly, the testing results are however a very good indicator that the DNN has fitted the optimal control function without overfitting the data (without memorizing the data).
We make sure that by verifying that the MAE during testing is of the same order of magnitude as the MAE during the training phase.

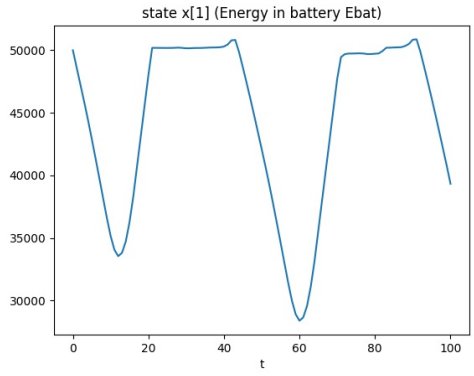


2. Qualitative evaluation by simulation

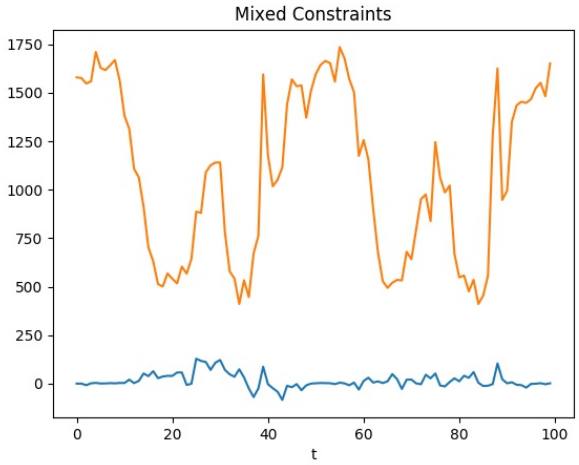
Here are the results for a simulation with disturbances that are the exact same as those that the DNN controller worked on, as well as the results for disturbances where gaussian noise has been added on that original disturbance the DNN trained on (3 different noise levels : 2, 10 and 20)

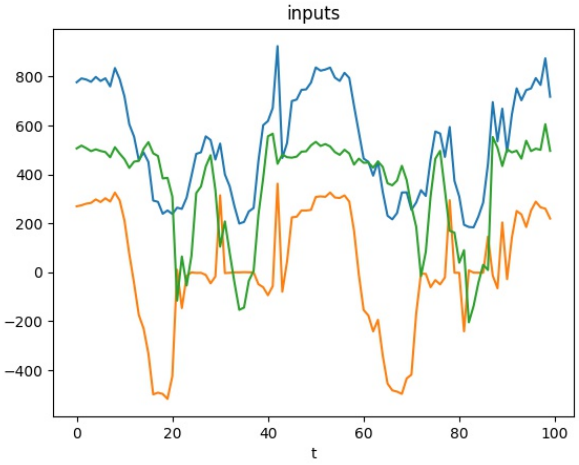


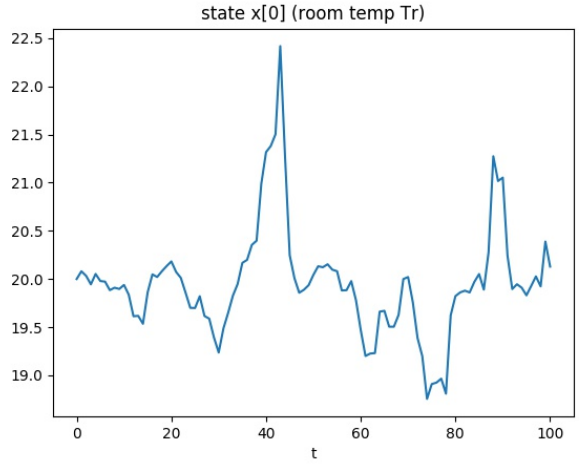


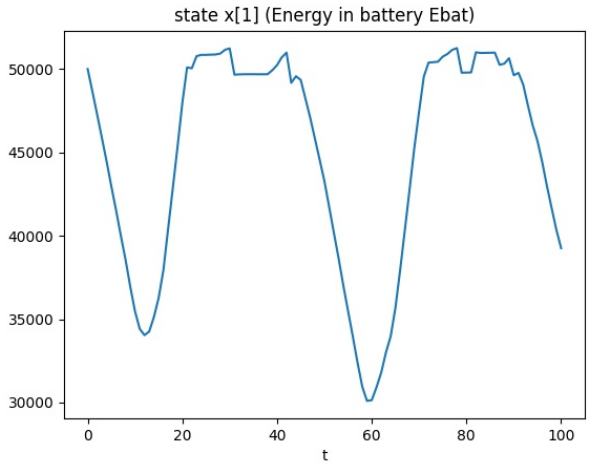


Now the results with noise gaussian noise level = 2 added to the disturbances :

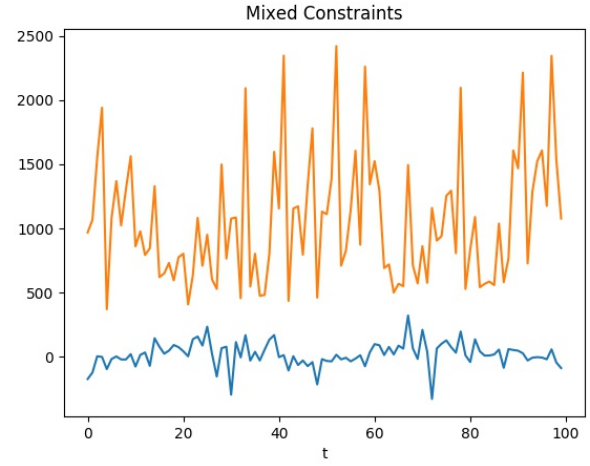


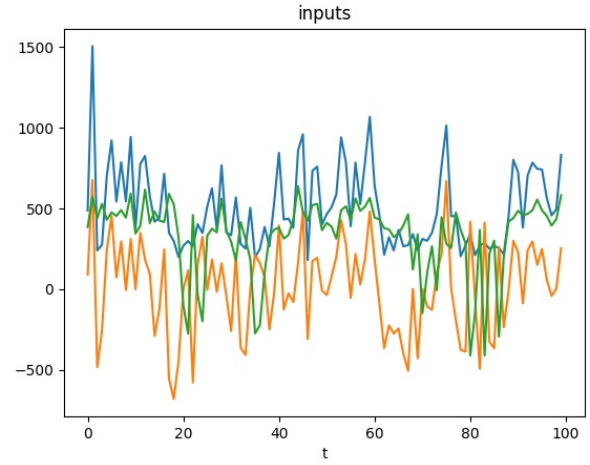


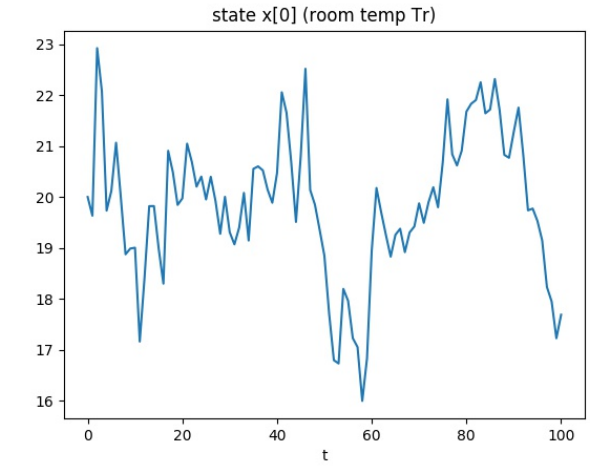


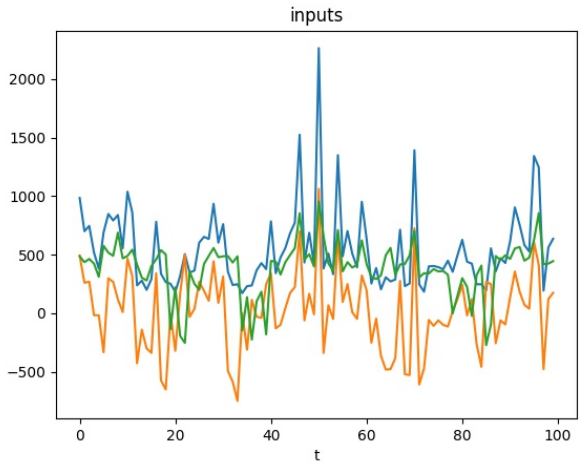


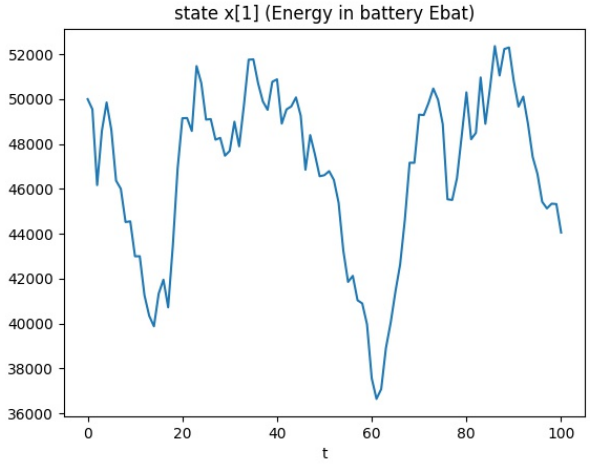
With level 10 :

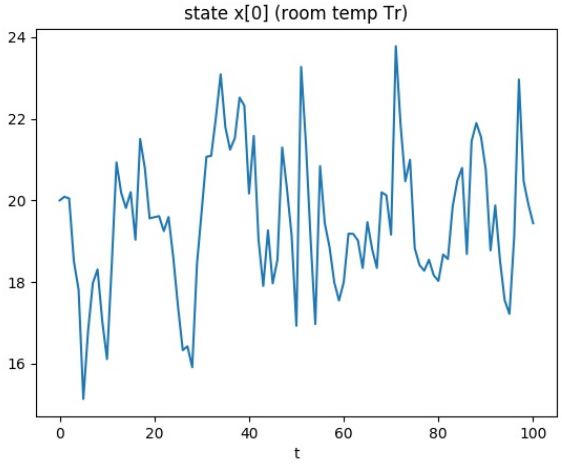




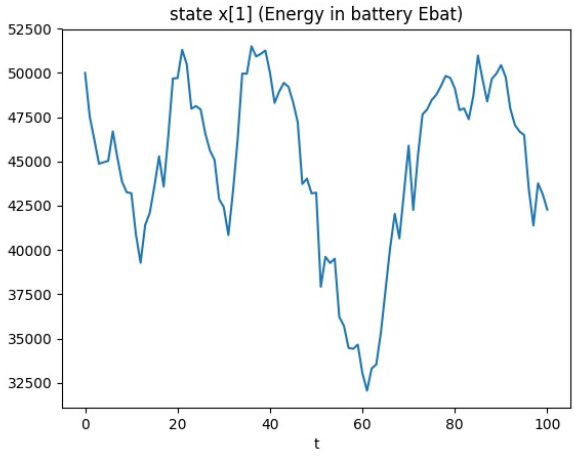








And level 20 :



The results clearly show that the action taken by the DL controller is definitely not optimal since bound conditions are often violated for different important parameters : the temperature input, for example, goes below 20 from time to time and the mixed conditions below 0 as well.
Worth noticing as well, is the fact the conditions are not less respected with higher magnitude as the disturbance level is increased.

A few reasons can explain the questionable behavior of our DL controller :

- The DNN has only been trained until a Mean Average Error of 7 was reached which is probably not enough to get an output that always respects the boundary conditions.
- The DNN was tested on disturbances where we added noise, but was however trained on disturbances without gaussian noise.

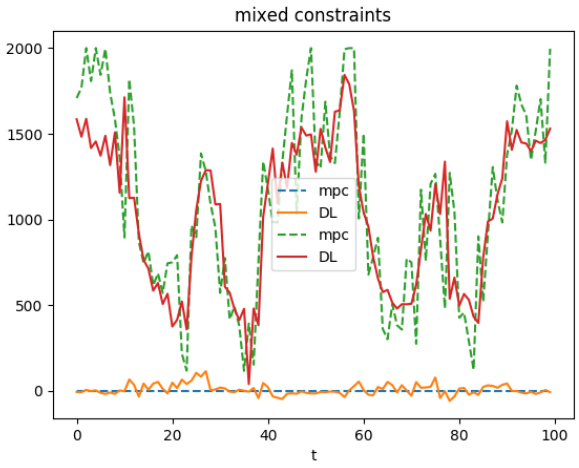
- The gaussian noise added on top of the disturbances cannot be predicted by the DNN even though the original disturbance, that has a clear pattern, can. the DL controller doesn't yield optimal results since some of the bounds we set are violated more and more as the disturbance level is increased : the inputs, for example, go below the threshold value of 20 and the second mixed constraint is not always respected since it goes below zero from time to time.

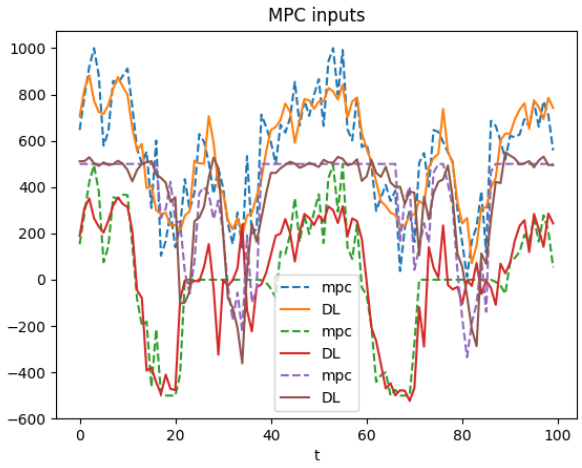
D. Possible improvements

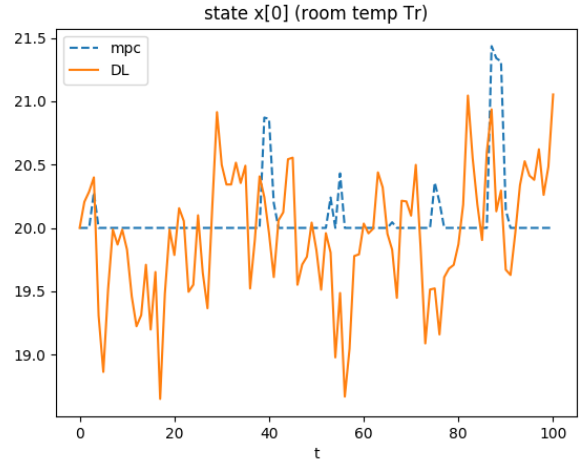
Let us, now that we've determined the possible reasons for our bad performance, try to implement a new version of the DL controller where the DNN has been this time trained on disturbances with varying levels of noise added to the disturbances and compare the results yielded by the DL controller to and MPC controller simulation.

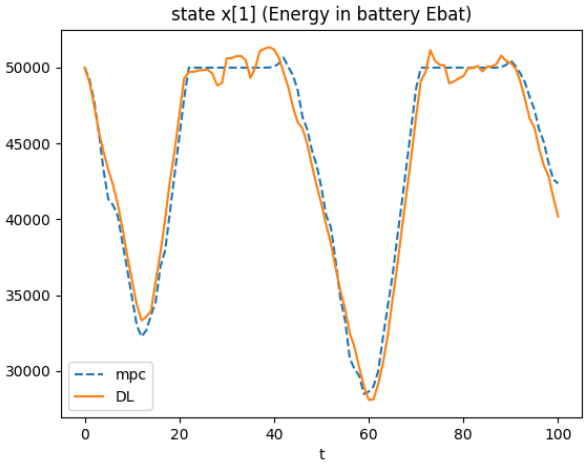
MPC controller action is dashed for comparison on the same plot to the DL controller's action.

Both simulations are executed for X_init = [20 50e04], N=5 and S = 100 (and a level of disturbance of 10) :









V. Conclusion

Training a Feed Forward 5 layers deep (50 units per layer) Neural Network on the data generated by an MPC controller taking action on disturbances with gaussian noise added to it has allowed us to refine our model and make it more performant as the 3 last plots clearly show.

However, it seems that the model still needs refining and probably much more training in order to reach a Mean Absolute Error low enough so that the boundary conditions imposed on the input and the states as well as the mixed conditions are always respected.

This latter goal is however not necessarily achievable, especially if noise level is too high (thus very unpredictable by the DNN and sometimes not even solvable by the MPC controller).

References

1. https://keras.io/
2. The tutorial and the course from ISIS
3. Real-time optimal control via Deep Neural Networks: study on landing problems, Carlos Sánchez-Sánchez and Dario Izzo. *European Space Agency, ESTEC, Noordwijk, The Netherlands.*
4. Model Predictive Control: Theory and Design J. B. Rawlings and D. Q. Mayne July 9, 2012