UNIVERSITY OF SALERNO

DEPARTMENT OF COMPUTER ENGINEERING

Report Data Driven

Project 2

Synthesis of control policies from demonstrations for fully driverless trains

Francesco Avallone - 0622701488 WP2 - WP6

Martina Lamberti - 0622701476 WP3 - WP4

Lorenzo Pagliara - 0622701576 WP1 - WP5

{f.avallone20, m.lamberti61, l.pagliara5 }@studenti.unisa.it

Academic Year 2021 - 2022

Contents

1 Formulation of the control problem from the high-level description

The control problem developed involves the use of data from an expert, from which the agent has to take advantage to improve its performance in order to ensure certain properties (e.g. avoiding sudden accelerations, in close proximity to a station ensuring a "gentle" braking). Therefore, even though you don't have a reference mathematical model, the goal is to make the behavior of the system similar to the expert's behavior by referring to the data provided in input to the problem. At each step, the optimal control action is chosen to minimize a certain cost index that captures the discrepancy between the closed loop behavior of the agent and the one of the expert.

2 Design choices

For the realization of this control system, a probabilistic design approach is followed, in particular the *control from the demonstration*. This choice is motivated by the fact that, compared to a deterministic approach, it allows to take into account the uncertainty present in any system due to noisy sensors, actuators and environment. Furthermore, the presence of a demonstrative dataset, i.e. data describing the behavior of an expert, makes it logical to choose to synthesize a control policy that is as similar as possible to that of the demonstrator, so that it can be used by the agent to solve the same task as the one being demonstrated.

3 Relation between the code and the methodology

The relation between the code implemented and the methodology is described accurately in the Markdown documentation in the code. You can access to it directly in the GitHub repository clicking the following link:

GitHub repository

4 Numerical results and discussion of the performance

4.1 Representation of the trajectory of the train

From the figure 1, it is possible to notice that at the initial position the train is running, thus, from the data given in input, you cannot appreciate its whole journey. In addition, there is a lot of variance around the mean value of the trajectory which is relevant to the agent's performance.

It is necessary to place the emphasis on the interpolation process, needed to identify new points of a discrete set from the dataset: we use two different interpolation methods (*scipy.interpolate.interp1d* that interpolates a 1-D function and *CubicSpline* that gives an interpolating polynomial that is smoother and has smaller error than some other interpolating polynomials). In the same figure, there are the resulting trajectories which are practically overlapped to the human eye. In order to carry out a more rigorous analysis, we have created a script that captures the differences between the two interpolations with parity of position and it turns out that the discrepancy is on the third decimal digit.

4.2 Simulation

From the simulation 2 it is possible to observe how the behavior of the expert closed loop system is very oscillating and it does not succeed to easily reach the final state, using the target policy (it reaches only the state 1200).

As a result, it is obvious to expect that the agent's closed loop behavior, achieved by using a policy that minimizes the discrepancy with the expert's closed loop behavior, is equally fluctuating (figure 3).Both the simulation are obtained by setting as termination condition a state previous to the last, otherwise it would not stop. The reason for this oscillatory behavior, for both the expert and the agent, is due to the parameters of mean and variance of the pdfs $f(x_k|u_k, x_{k-1})$ e $g(x_k|u_k, x_{k-1})$, modelled as Gaussians. With the following table we wanted to demonstrate that in a given state index (e.g. state= 98) it is more likely to go back instead of going forward. The table represents what are the states that we can reach starting from the current state (column "Index"), what is the range of meters encoded with the same index state (column "Range meters") and how many control inputs allow to reach it (column "Number of occurrences").

In particular,in the table below, we can observe that, if the system is in the state index 98, evaluating the $f(x_k|u_k, x_{k-1})$ for each control input, we can observe that the probability to reach a next state is equal to 0. This is caused by the choice of the discretization step which brings together states in the same status index, as a result of low resolution. It follows, of course, that the probability of attaining the next state increases with a low resolution because of the function *round*, used to approximate the state to the state index, takes more advantage from approximation with a greater *d_step*.

Index	Range meters	Number occurrences
97	[1454.882, 1462.500]	43
98	[1462.681, 1472.838]	57

Table 1: Table obtained considering as current state index = 98.

4.3 Computational effort

We have seen that the computational effort increases with the decreasing of the discretization step or, likewise, the increasing in the number of samples.

The plot in figure 4 has been obtained by several simulations of the closed loop system with different number of samples and calculating whenever the execution time.

The code for the mandatory WPs is taken from the given supporting code [5] and Lab 4 [2].

5 Possible future developments for improvements

Possibile improvements can be achieved by:

- performing a more accurate probabilistic modeling of the system, such as refining the model parameters, by using data from real process;
- using more sources involves more experts from which is possible to take the best attitude of each one, in every time step. In conclusion, the overall behavior may be better than the behavior of individual experts.
- using the target policy to define time-varying constraints for the MPC control input. In this way, the controller has some indirect knownledge about the velocity that the train has to apply in a particular region of the path.

A Optional WP

A.1 WP4 - optional: receding horizon

If we want to apply a receding version of the algorithm ($tHor > 0$), we have to take into account the future γs necessary to choose a good agent policy that minimizes the $\mathcal{D}_{KL}(f_{0:N} \,||\, g_{0:N})$ in the employed time window.

A.1.1 Design choices

What differentiates the greedy version from the one with the receding horizon is, in the code, the variable $stateAtTime$ that indicates the next states related to the current one, on which we calculate $\gamma_k(X_k)$, so that the policy adopted at the current timestep minimizes the $\mathcal{D}_{KL}(f_{0:N} \,||\, g_{0:N})$.

A.1.2 Comments and observations on the receding horizon version

From the results, ascertained that the two behaviors exhibit strong oscillations, it is not possible to have meaningful diagrams in order to appreciate the improvements thanks to the time horizon.

A.2 WP5 - optional: generalization with non-Gaussian probability functions

To ensure that the control from demonstration algorithm implemented was as general as possible, we have implemented a version that allows the use of a generic probability distribution. With that goal, we have implemented a function that allows to manage the binning of a trajectory of states and control inputs, in order to obtain the conditional pmf $f(x_k|x_{k-1}, u_k)$. Furthermore, a function has been implemented capable of calculating the DKL according to its more generic definition, that is the integral.

A.2.1 Design choices

Due to the lack of datasets containing realistic trajectories of the system, it has not been possible to obtain a probabilistic description of the system by using the data binning. For this reason, we have decided to compare the closed-loop response of the system described by a Gaussian distribution by using numerical and analytical techniques.

A.2.2 Numerical results and discussion of the performance

Comparing the simulations of the closed-loop behavior of the system obtained respectively by using numerical and analytical techniques, it is possible to observe that the trajectory achieved, in particular for the position, has a slightly less oscillatory behavior in the case of using numerical techniques (figure 5 and figure 6). However, the control from demonstration algorithm that makes use of numerical techniques requires greater computational and temporal resources.

The code in this work package takes inspiration from Lab 1 [4].

A.3 WP6 - optional: MPC solution

A.3.1 Design choices

In order to implement a further algorithm and benchmark the results, we developed a MPC version by using do-mpc python toolbox. As well known, it is necessary to have a mathematical model: for this reason, we have considered to use the open loop system, $f(x_k|u_k, x_{k-1})$, to generate a dataset associating to each couple (u_k, x_{k-1}) the future state x_{k+1}. Starting from this dataset, a regression algorithm has been developed in order to have a mathematical model that approximates the $f(x_k|u_k, x_{k-1})$ behavior.

In addition, we decided to put some noise in the control input in order to evaluate the performance of the closed loop system in presence of noise. Moreover, if we apply a control input not provided by MPC, we wanted to see the reaction capabilities of the MPC controller.

A.3.2 Numerical results and discussion of the performance

As time horizon decreases, further time is needed to reach the target as you can see in figures 7 and 8, where a time horizon of 7 and 5 was used respectively. It is also possible to observe that the closed loop response of the system by employing MPC is much smoother than the one obtained by using the control from demonstration. This is due to the fact that MPC makes use of information from a deterministic mathematical model.

The code in the WP6 takes inspiration from the Gekko example [3] and the project structure of do-mpc [1].

B Images

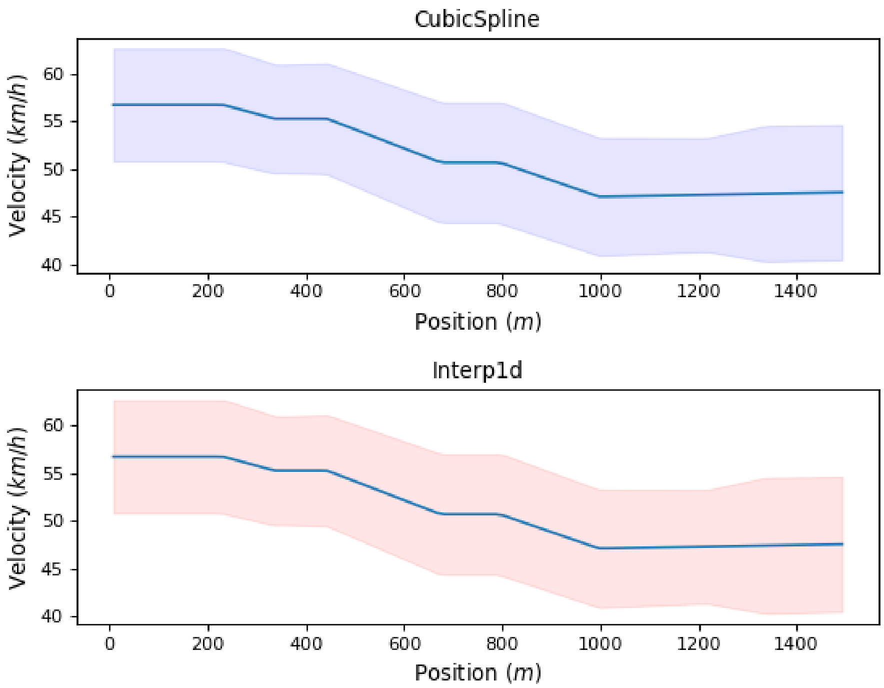

Figure 1: Visualization of the trajectory of the train and interpolation process with two different methods.

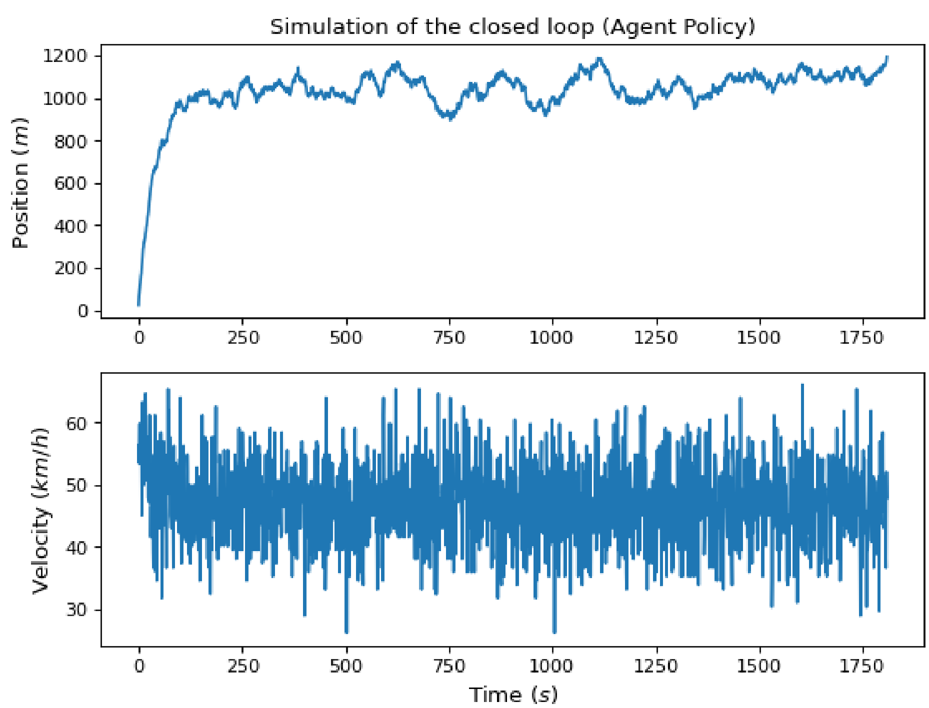

Figure 2: Behavior of the expert closed loop system.

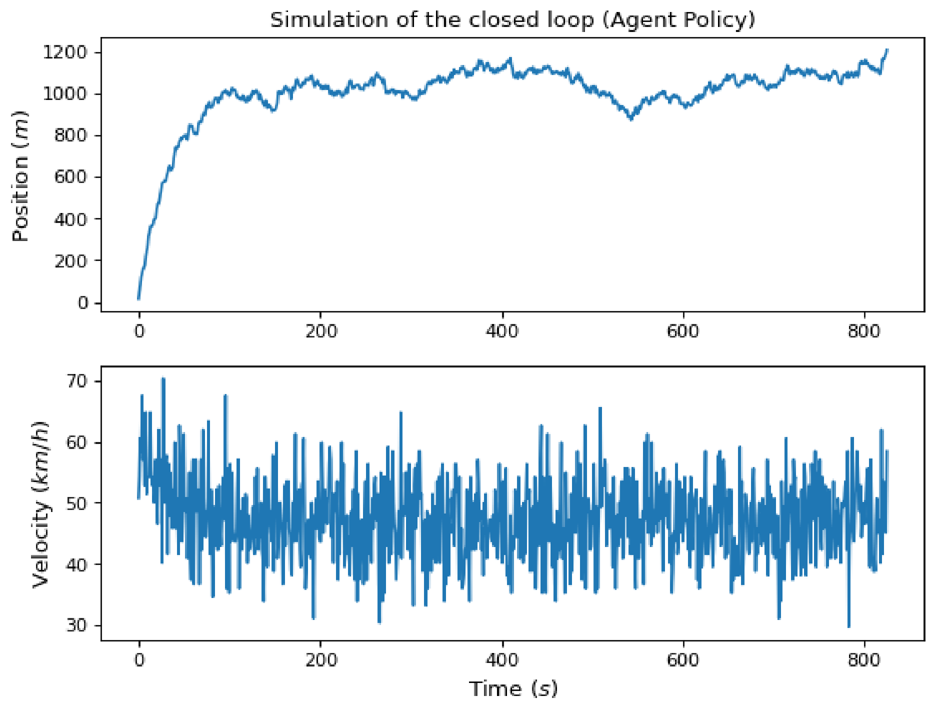

Figure 3: Behavior of the agent closed loop system.

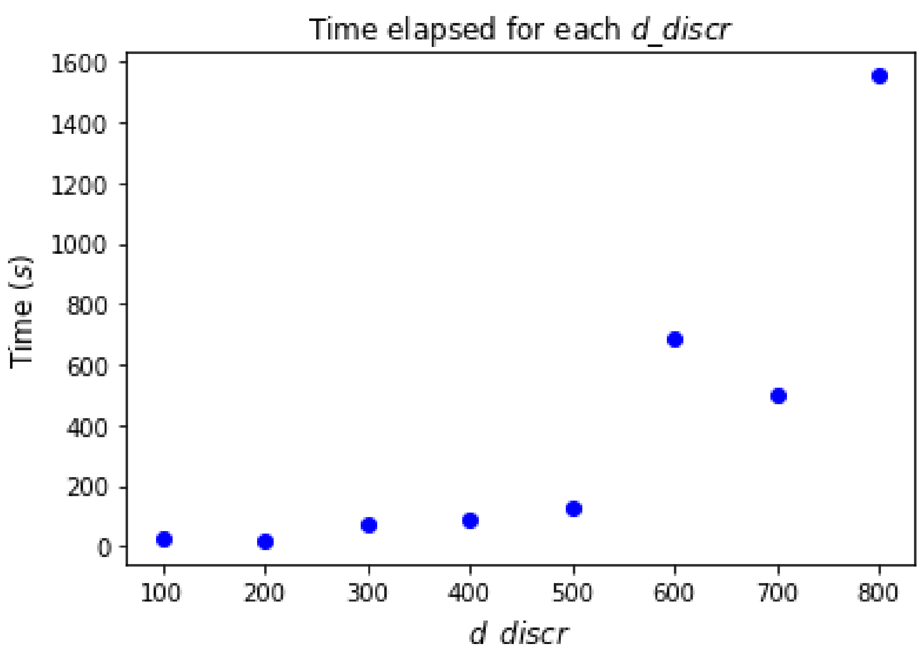

Figure 4: Computational time as the number of samples increases

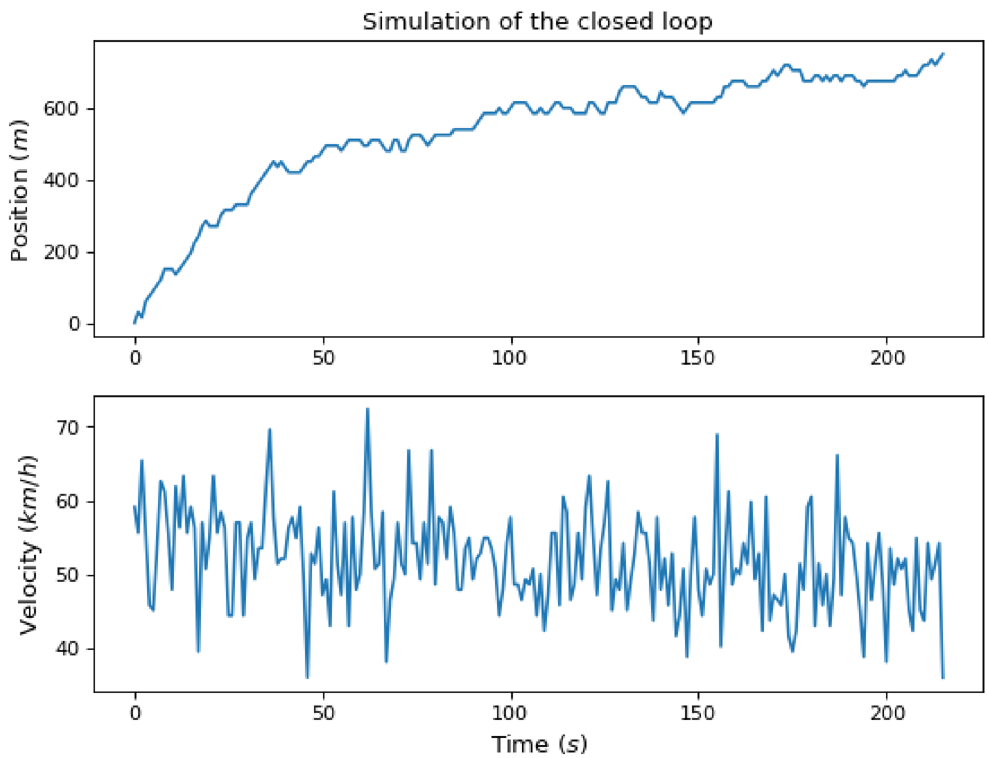

Figure 5: Closed loop obtained through numerical techniques.

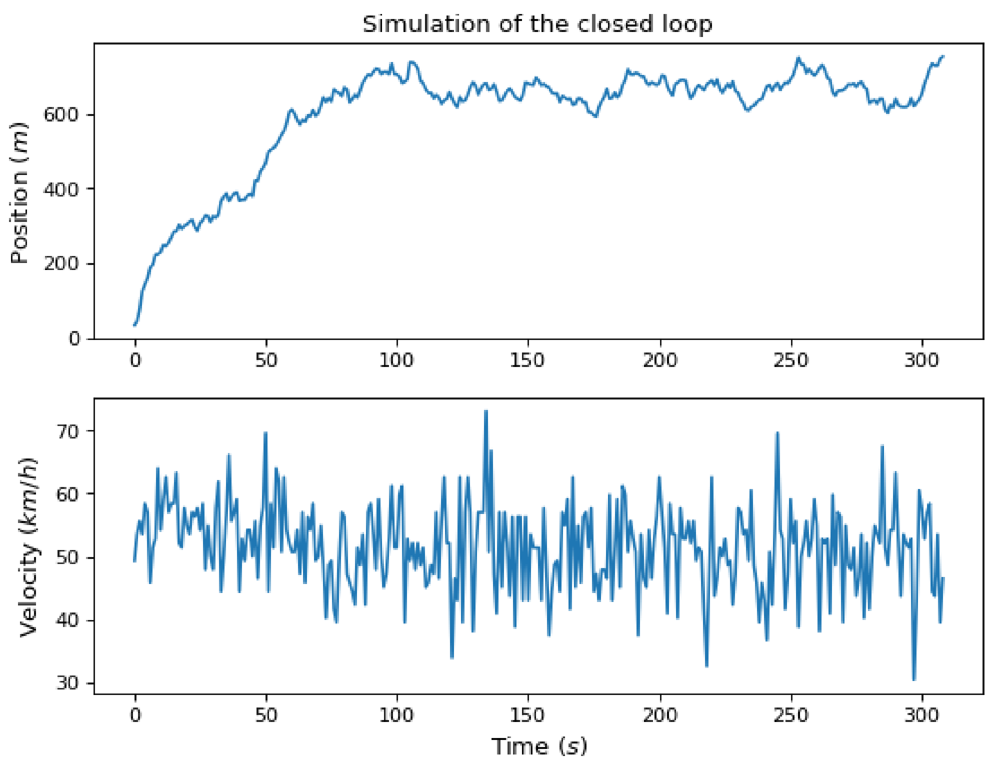

Figure 6: Closed loop obtained through analytical techniques.

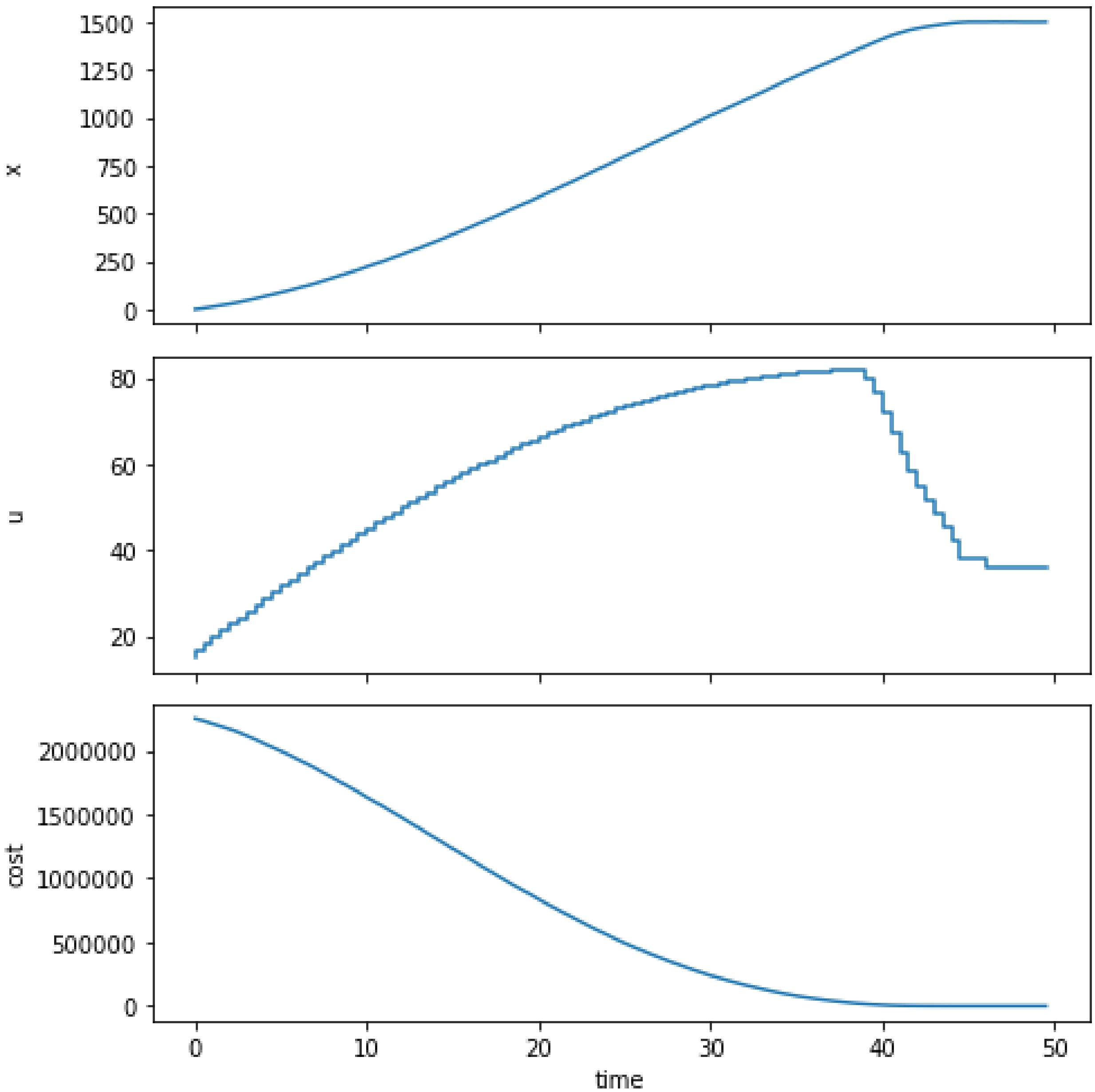

Figure 7: In the upper figure, there is the sequence of the states (positions assumed by the train) as time changes. The middle one is characterized by the sequence of the control inputs as time changes. The lower figure represents the cost function as a function of time.

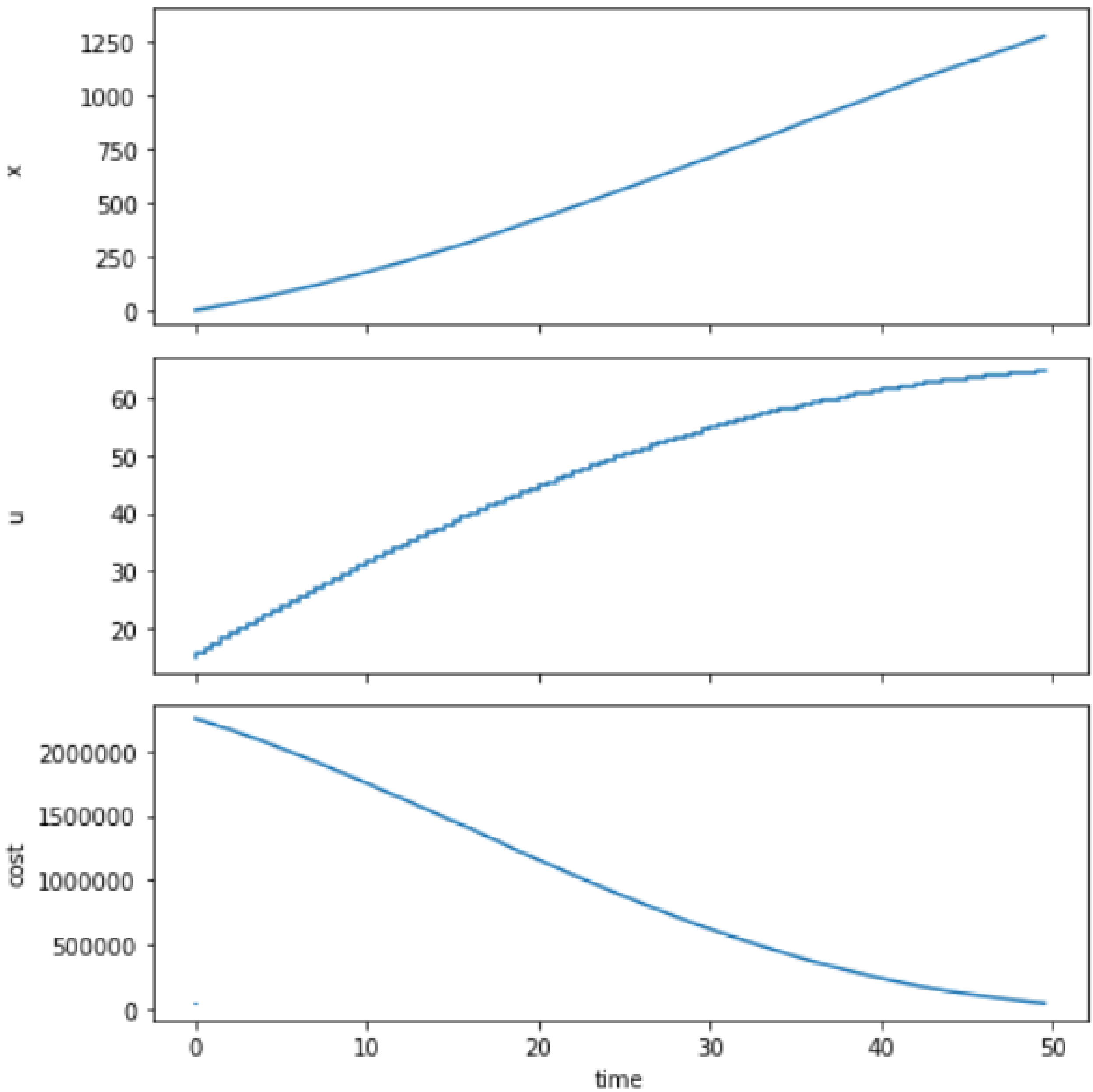

Figure 8: Results with time horizon = 5. As you can see, at time = 50 s the position reached is 1250 m.

References

[1] Lucia - Codrean - Schoppmeyer - Engell. *Rapid development of modular and sustainable nonlinear model predictive control solutions.* URL: https://www.do-mpc.com/en/latest/project_structure.html.

[2] Giovanni Russo. *Control_from_demons.ipynb*. URL: https://elearning.unisa.it/pluginfile.php/318828/mod_resource/content/0/Control_from_demons.ipynb.

[3] Giovanni Russo. *GEKKO_example.ipynb*. URL: https://elearning.unisa.it/mod/folder/view.php?id=20730.

[4] Giovanni Russo. *PMFs_from_data.ipynb*. URL: https://elearning.unisa.it/pluginfile.php/311159/mod_resource/content/0/PMFs%20from%20data-Lab%20project_COMPLETE_version.ipynb.

[5] Giovanni Russo. *SupportingScript.ipynb*. URL: https://elearning.unisa.it/pluginfile.php/322708/mod_resource/content/0/Project%202%20-%20supporting%20script.ipynb.