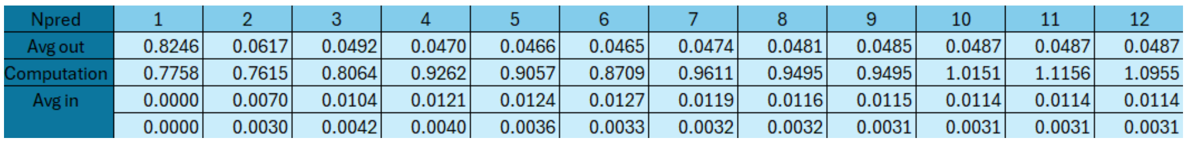
Laboratory 2

PÎNDICHI Elena and JAROUSSEAU Arthur

October 14, 2024

1 Changing values

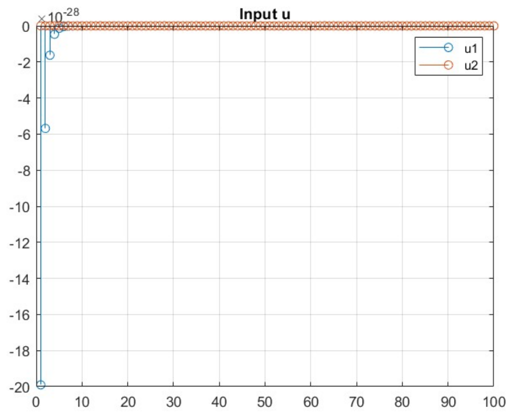
1.1 Prediction horizon N_p



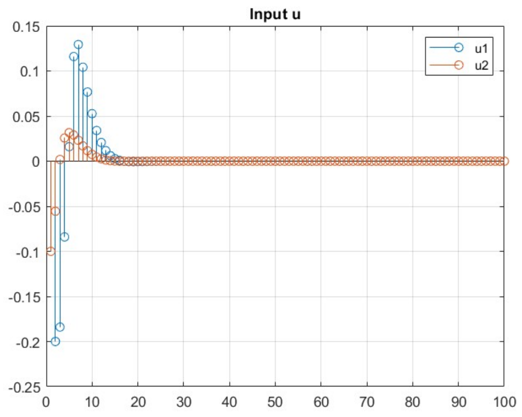
Npred	1	2	3	4	5	6	7	8	9	10	11	12
Avg out	0.8246	0.0617	0.0492	0.0470	0.0466	0.0465	0.0474	0.0481	0.0485	0.0487	0.0487	0.0487
Computation	0.7758	0.7615	0.8064	0.9262	0.9057	0.8709	0.9611	0.9495	0.9495	1.0151	1.1156	1.0955
Avg in	0.0000	0.0070	0.0104	0.0121	0.0124	0.0127	0.0119	0.0116	0.0115	0.0114	0.0114	0.0114
	0.0000	0.0030	0.0042	0.0040	0.0036	0.0033	0.0032	0.0032	0.0031	0.0031	0.0031	0.0031

Figure 1: Npred values

Changing the prediction horizon can lead to important behavioural changes that can be observed in the system's performances and computation time.



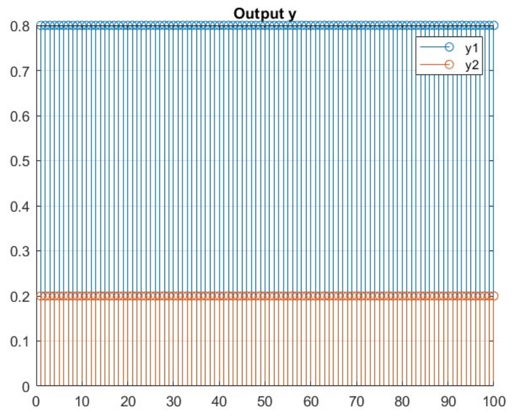
(a) N = 1



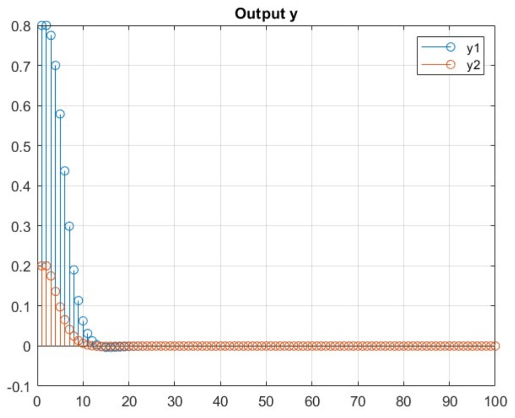
(b) N = 12

Figure 2: Inputs

Reducing the N_{pred} leads to a decrease in the computation time. This happens because the optimization problem solved at each time step predicts the system's future behaviour over a horizon. The smaller the prediction horizon, the smaller the number of future steps that must be calculated. This reduces the size and complexity of the optimization problem. Increasing the prediction horizon we can observe that the computation time is increasing as well, because it needs more time to process the future steps that it has to make due to the fact that it sees much more into the future.

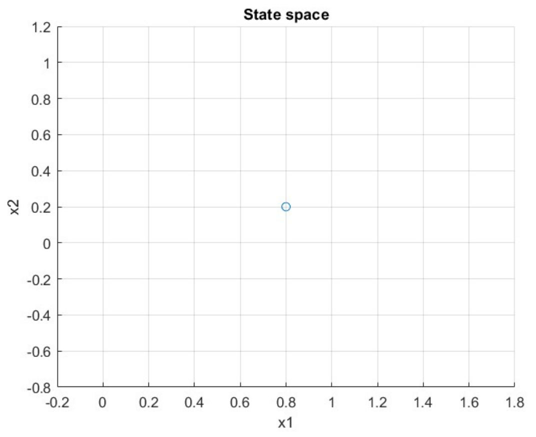


(a) N = 1

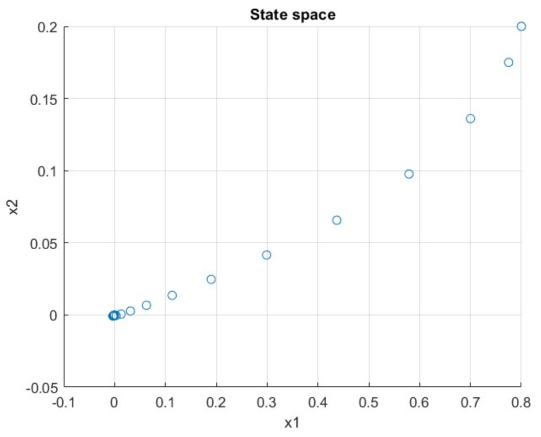


(b) N = 12

Figure 3: Outputs



(a) N = 1



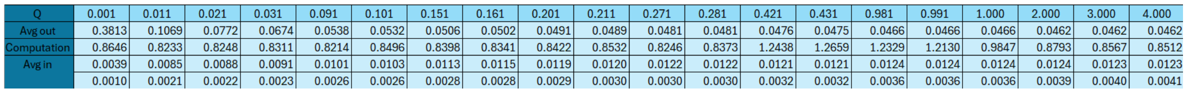
(b) N = 12

Figure 4: State space

Reducing N_{pred} typically increases error of the system's output. This is because MPC uses the prediction horizon to anticipate future disturbances and adjust the control input accordingly. In our case, we see that the average output seems to be decreasing with a lower N_{pred}.

As N_{pred} decreases, the controller has less information about the future states, leading to more aggressive control actions in the shortest term. This can be observed through an increase in the average control input as the horizon is reduced. A larger horizon allows the controller to "smooth" its inputs over time, reducing the total control effort.

1.2 Q matrix

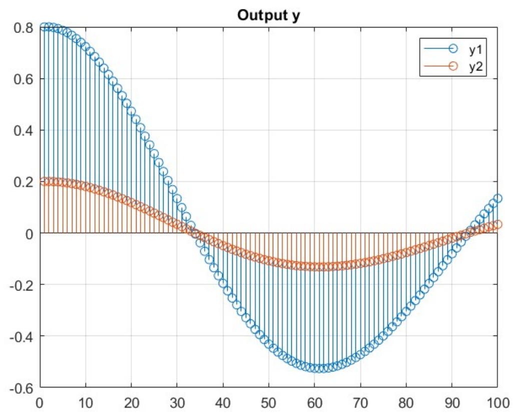


Q	0.001	0.011	0.021	0.031	0.091	0.101	0.151	0.161	0.201	0.211
Avg out	0.3813	0.1069	0.0772	0.0674	0.0538	0.0532	0.0506	0.0502	0.0491	0.0489
Computation	0.8646	0.8233	0.8248	0.8311	0.8214	0.8496	0.8398	0.8341	0.8422	0.8532
Avg in	0.0039	0.0085	0.0088	0.0091	0.0101	0.0103	0.0113	0.0115	0.0119	0.0120
	0.0010	0.0021	0.0022	0.0023	0.0026	0.0026	0.0028	0.0028	0.0029	0.0030

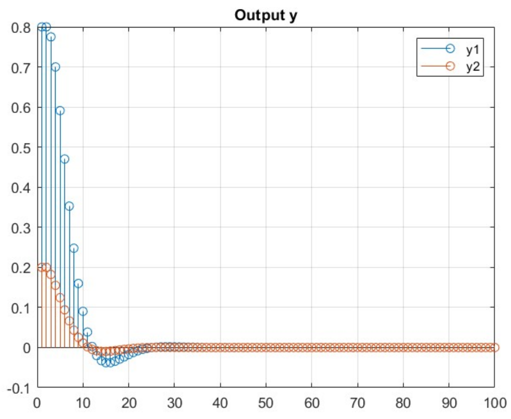
Q	0.271	0.281	0.421	0.431	0.981	0.991	1.000	2.000	3.000	4.000
Avg out	0.0481	0.0481	0.0476	0.0475	0.0466	0.0466	0.0466	0.0462	0.0462	0.0462
Computation	0.8246	0.8373	1.2438	1.2659	1.2329	1.2130	0.9847	0.8793	0.8567	0.8512
Avg in	0.0122	0.0122	0.0121	0.0121	0.0124	0.0124	0.0124	0.0124	0.0123	0.0123
	0.0030	0.0030	0.0032	0.0032	0.0036	0.0036	0.0036	0.0039	0.0040	0.0041

Figure 5: Q values

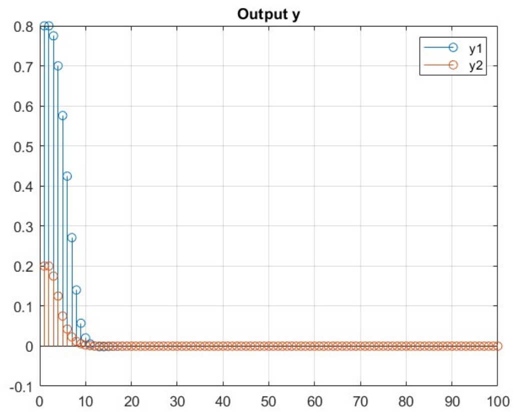
The higher the output weight is, the more the controller prioritizes minimizing the output error, improving tracking performance. But we can observe a significant difference in the average input, meaning that we will have a better performance while increasing the Q matrix, but worse input, as seen in Table: 5.



(a) Q = 0.001



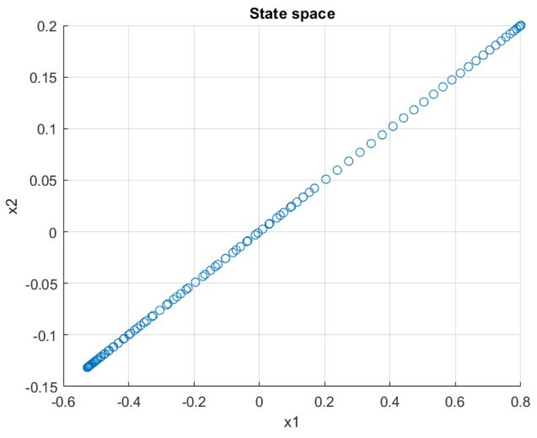
(b) Q = 0.421



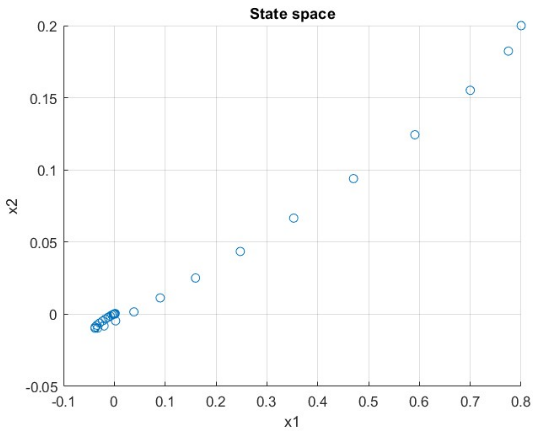
(c) Q = 4

Figure 6: Outputs

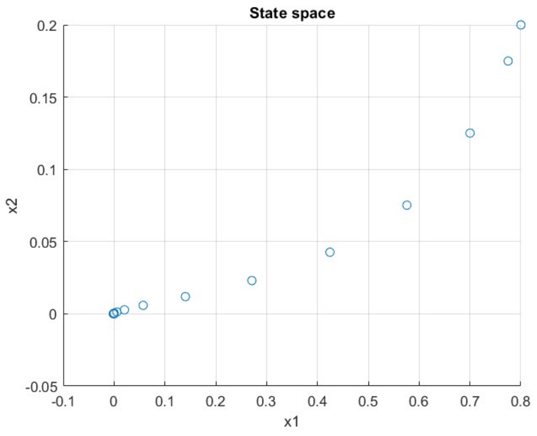
Looking at the outputs, we can observe that the lowest the Q, the lowest the performances as discussed earlier. The output does not stabilize in the simulation time, and as we increase it, we can see that it provides better performances. As we increase the Q matrix, we also observed that at some point, the differences will be almost insignificant because of the system that we are working on.



(a) Q = 0.001



(b) Q = 0.421

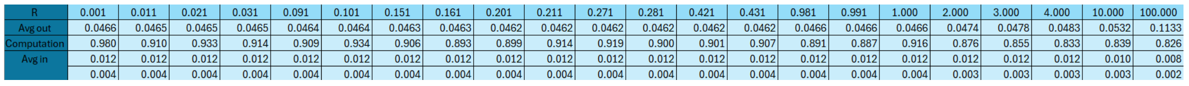


(c) Q = 4

Figure 7: State space

Through Fig. 7 we can see that, as expected, Q is the cost matrix for the states and hence it affects the state. The output is affected due to the fact that, in our problem, the output are represented by the 2 first states, and we can see directly the impact of the Q matrix. In the state space, they converge more naturally while working with a higher Q, but a higher Q will affect the inputs.

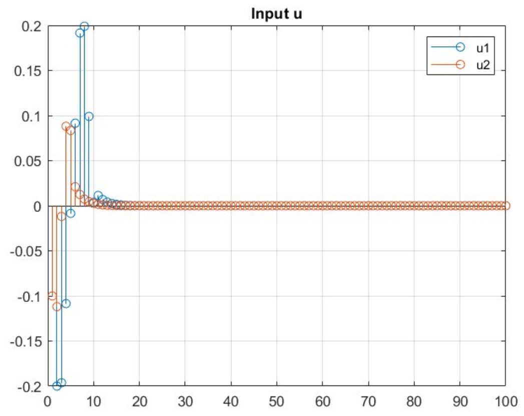
1.3 R matrix



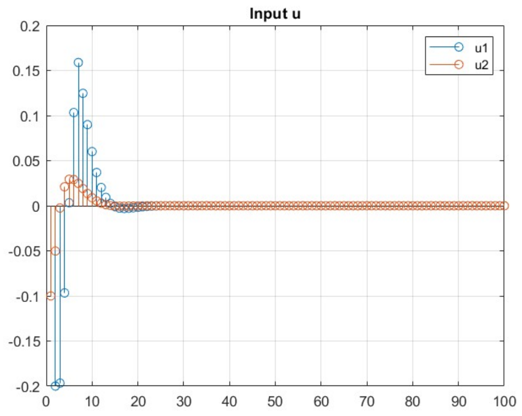
R	0.001	0.011	0.021	0.031	0.091	0.101	0.151	0.161	0.201	0.211	0.271	0.281	0.421	0.431	0.981	0.991	1.000	2.000	3.000	4.000	10.000	100.000
Avg out	0.0466	0.0465	0.0465	0.0465	0.0464	0.0464	0.0463	0.0463	0.0462	0.0462	0.0462	0.0462	0.0462	0.0462	0.0466	0.0466	0.0466	0.0474	0.0478	0.0483	0.0532	0.1133
Computation	0.980	0.910	0.933	0.914	0.909	0.934	0.906	0.893	0.899	0.914	0.919	0.900	0.901	0.907	0.891	0.887	0.916	0.876	0.855	0.833	0.839	0.826
Avg in	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.010	0.008
	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.004	0.003	0.003	0.003	0.003	0.002

Figure 8: R values

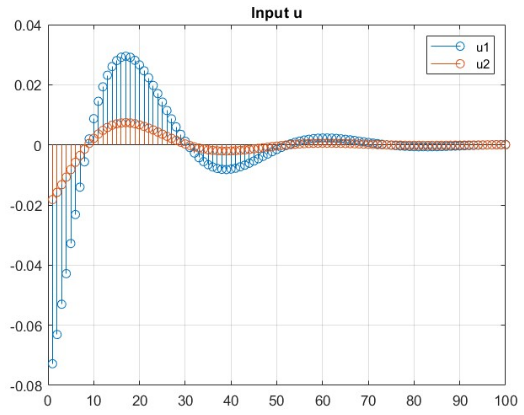
Increasing the input weight reduces aggressive control actions, leading to smoother control inputs. So, we can observe in the Table: 8 that the bigger the R matrix is, the better the inputs are, but the worse the performances become. It does not affect the computation time, but it is difficult to choose wether is better to increase its value or not, depending on the problem we are working on.



(a) R = 0.001



(b) R = 0.421

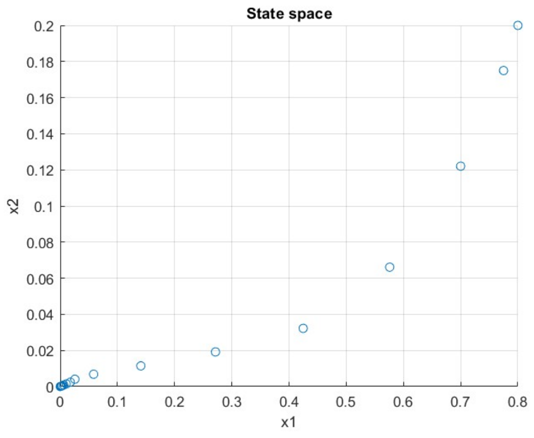


(c) R = 100

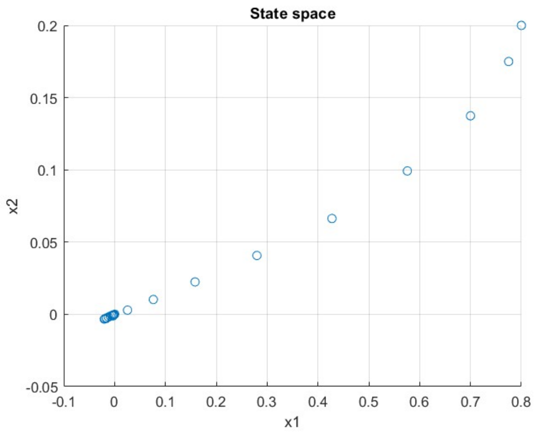
Figure 9: Inputs

As expected, the R matrix will affect the inputs, and we can observe in Fig: 9 that the higher it becomes, the smoother the command will be.

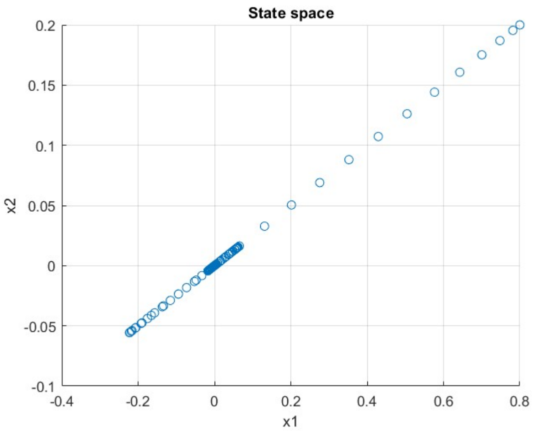
We can also observe that it affects the states, which means that it affects the performance. As it gets higher, we can see that the states exceed the reaching point and they return. When we reduce the R matrix, we let the command to act more aggressively and get to the stabilization point much faster, but when we increase the input cost matrix, we limit the command but we lose the performances.



(a) R = 0.001



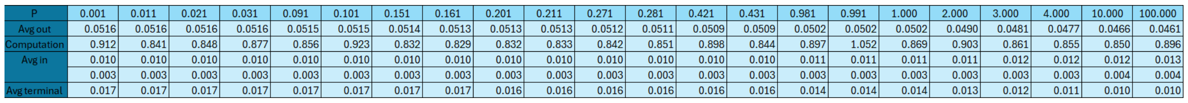
(b) R = 0.421



(c) R = 100

Figure 10: State space

1.4 P matrix

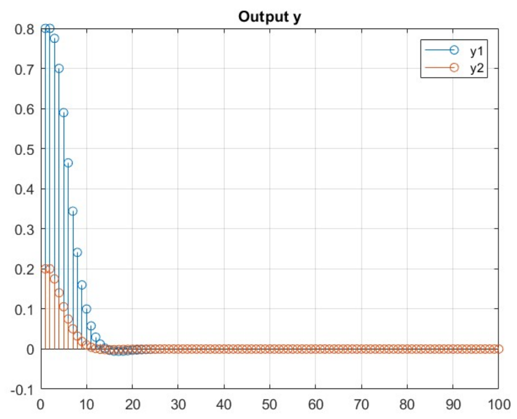


P	0.001	0.011	0.021	0.031	0.091	0.101	0.151	0.161	0.201	0.211	0.271	0.281	0.421	0.431	0.981	0.991	1.000	2.000	3.000	4.000	10.000	100.000
Avg out	0.0516	0.0516	0.0516	0.0516	0.0515	0.0515	0.0514	0.0513	0.0513	0.0513	0.0512	0.0511	0.0509	0.0509	0.0502	0.0502	0.0502	0.0490	0.0481	0.0477	0.0466	0.0461
Computation	0.912	0.841	0.848	0.877	0.856	0.923	0.832	0.829	0.832	0.833	0.842	0.851	0.898	0.844	0.897	1.052	0.869	0.903	0.861	0.855	0.850	0.896
Avg in	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.011	0.011	0.011	0.011	0.012	0.012	0.012	0.013
	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.004	0.004
Avg terminal	0.017	0.017	0.017	0.017	0.017	0.017	0.017	0.017	0.016	0.016	0.016	0.016	0.016	0.016	0.014	0.014	0.014	0.013	0.012	0.011	0.010	0.010

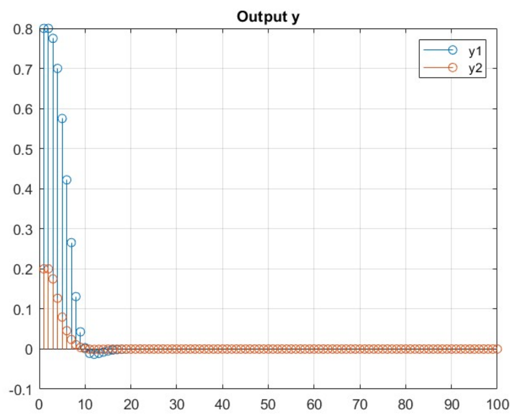
Figure 11: P values

In our study, we varied the terminal weight P, which directly influences how MPC controller prioritizes reaching a specific final state at the end of the prediction horizon. By increasing P, we observed that the system's final state converged more accurately to the desired target. This adjustment forced the controller to place greater focus on long-term goals, leading to more precise tracking and better stability. As P increased, the control actions became more focused on minimizing the error at the end of the horizon, ensuring that the system reached its objective with higher accuracy. This demonstrates the importance of P in guiding the system.

However, increasing the terminal weight can also present challenges in terms of system reliability, particularly in the presence of external disturbances. A higher P forces



(a) P = 0.001

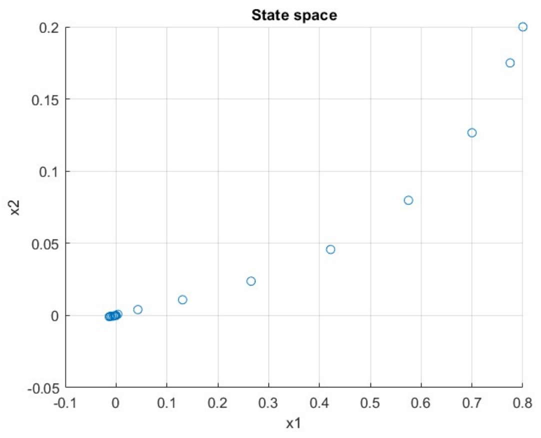


(b) P = 100

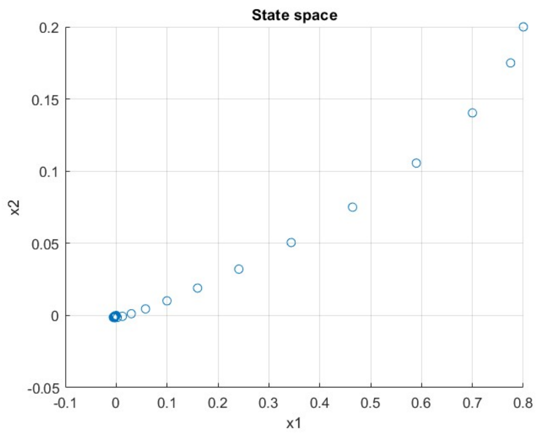
Figure 12: Outputs

the controller to prioritize reaching the exact final state, which can make the system less flexible in adapting to unexpected changes or disturbances.

To resume, it is preferable to use a long horizon with no strong penalty on the terminal state, as this allows the controller to anticipate future system behaviors and plan smoother control actions. This approach enhances stability and robustness against disturbances.



(a) P = 0.001



(b) P = 100

Figure 13: State space

2 Exercises

2.1 Reducing the N_{pred}

As presented in the previous section, if we decrease the prediction horizon, we will receive an answer in the shortest time because there are not many steps that we have to predict for the future. It decreases the size and complexity of the optimization problem but we will receive higher errors of the system's output and input because it does not get enough time to anticipate future disturbances and adjust the parameters accordingly.

The input will also be more aggressively because it was not provided with enough information about the future states.

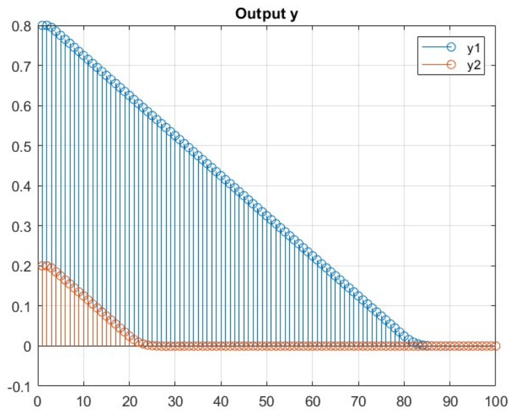
Therefore, it is not recommended to work with a small prediction horizon because the system will not receive enough information to act upon it.

2.2 Prediction horizon and terminal weight

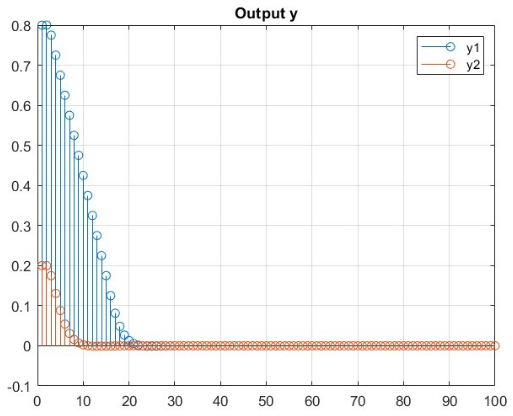
We have already mentioned in the previous section the effect of the terminal weight and from a stability point of view, we recommend *a long horizon with no particular penalty on terminal state*, because, in order to receive better performances, we need to increase the prediction horizon so that both the response and the command would have time to react and to change accordingly. The terminal weight affects only the terminal state and the faster it gets after iterating over the prediction horizon to the reference point. With this decision, we will enhance the stability and robustness against disturbances.

2.3 Adding constraints on the state variation

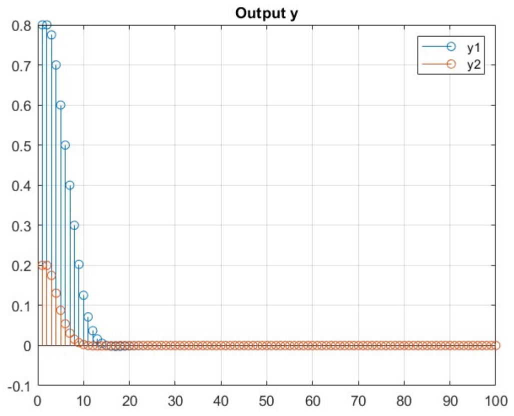
In our study, we varied the constraints on the system's state to observe how the MPC controller would handle them. Specifically, we tested both strict and moderate constraints to see if the controller could maintain its performance while staying within the allowed limits. In our case, the MPC was able to respect all the constraints during the simulation, even under more restrictive conditions. This shows that, under the given setup and system dynamics, the controller could find feasible solutions that met both the constraints and the control objectives.



(a) 0.01



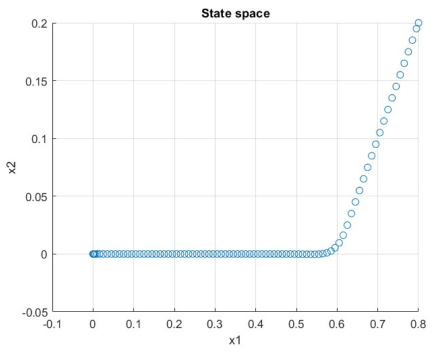
(b) 0.1



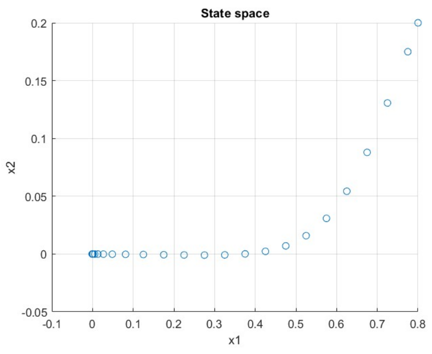
(c) 1

Figure 14: Output

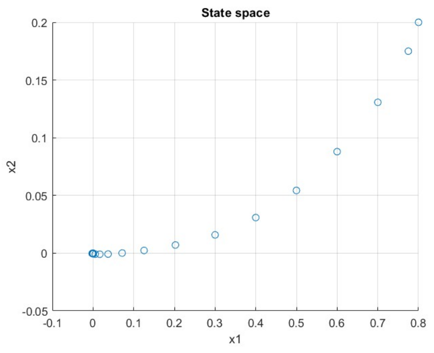
However, this isn't always guaranteed. In more complex systems or with very tight constraints, the MPC might struggle to find a solution that always satisfies every condition. When constraints become too restrictive or if external factors like disturbances come into play, the controller might violate the limits, leading to performance degradation or instability. Therefore, while the controller performed well in our case, it's important to recognize that constraint satisfaction can be highly dependent on the system setup and the level of flexibility allowed in the constraints.



(a) 0.01

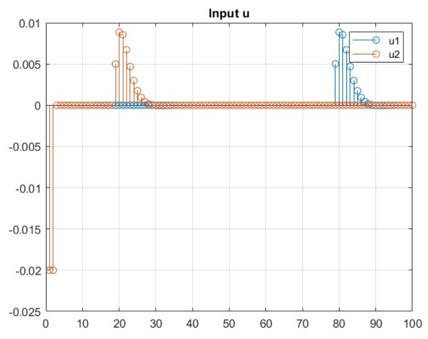


(b) 0.1

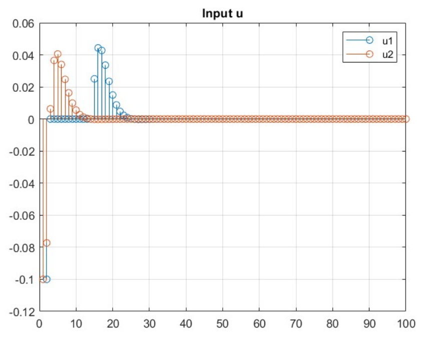


(c) 1

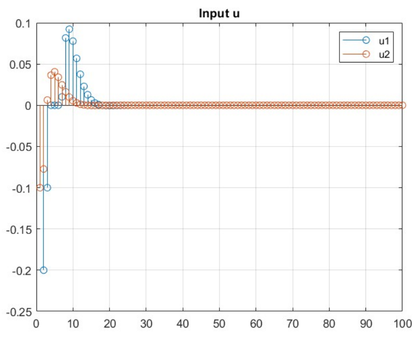
Figure 15: State space



(a) 0.01



(b) 0.1



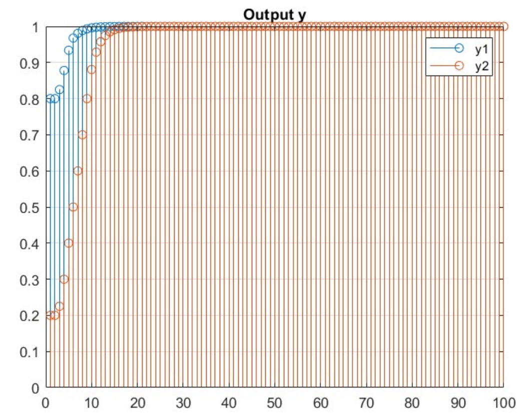
(c) 1

Figure 16: Inputs

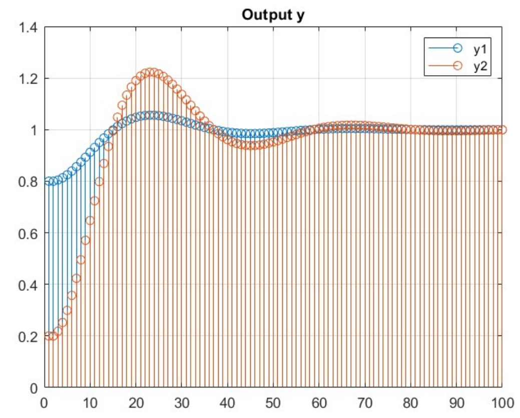
2.4 Reference tracking problem

When we chose to penalize the cost function for the input variations, we are deciding either to let the command act as aggressively as it could or to penalize it and to receive a smoother command. The reference that we implied is a step function, and for this particular reference, we did not need to change the cost matrix for the states.

We can observe from Fig. 14 that applying a smaller R, will make the output follow the reference much faster, but the input is too aggressive and it does not react to possible disturbances applied over the system.

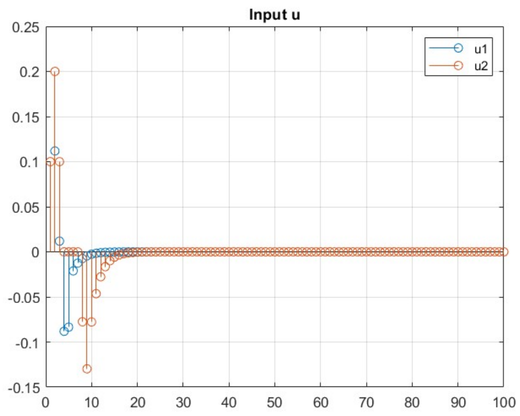


(a) 0.01

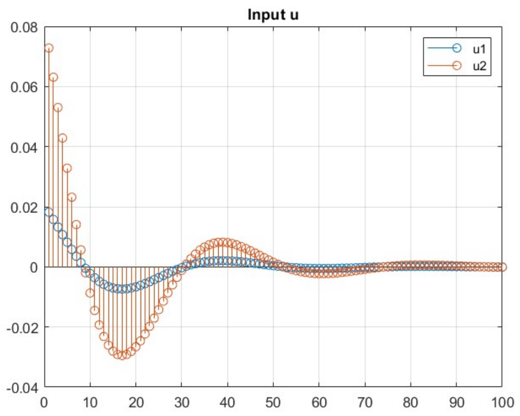


(b) 100

Figure 17: Output

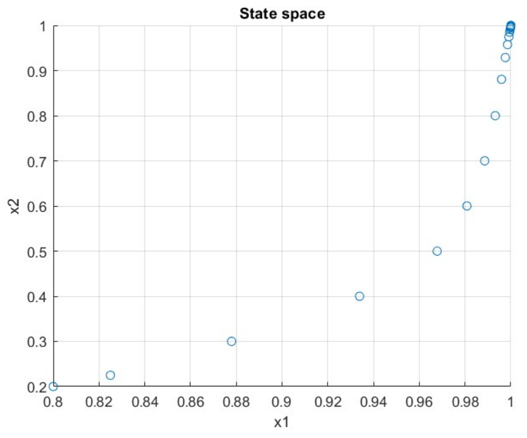


(a) 0.01

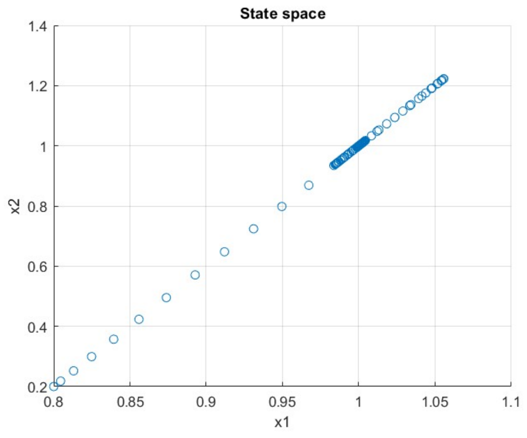


(b) 100

Figure 18: Inputs



(a) 0.01



(b) 100

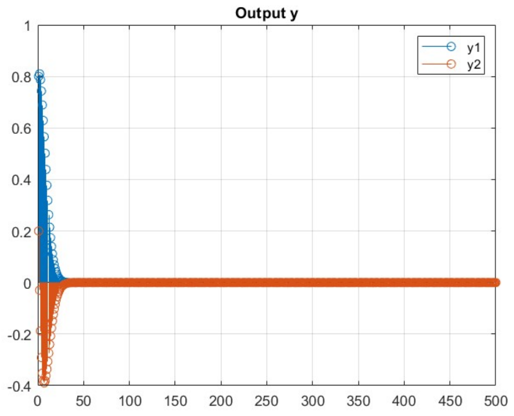
Figure 19: State space

The more we increase the R matrix, the better the input will become and the smoother it will get, but we will loose all performances of the system's output and states as explained in the previous section and as we can see in the figures from above.

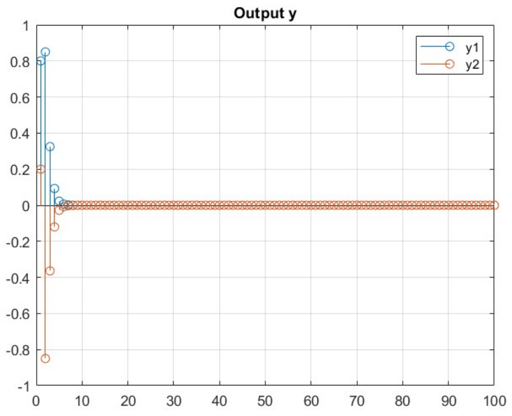
2.5 Previous lab session

We will choose to change the sampling time for this system:

$$A = \begin{bmatrix} 0 & 1 \\ 2 & -3 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{1}$$

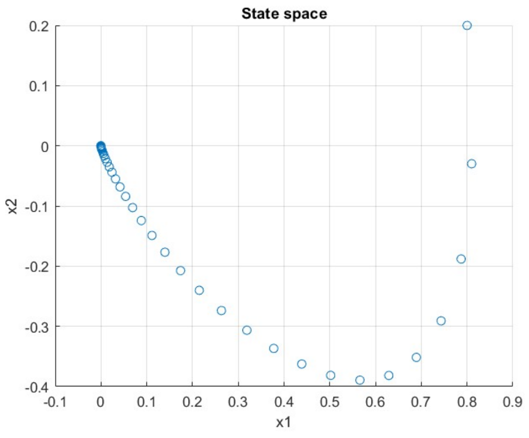


(a) $T_e = 0.1$

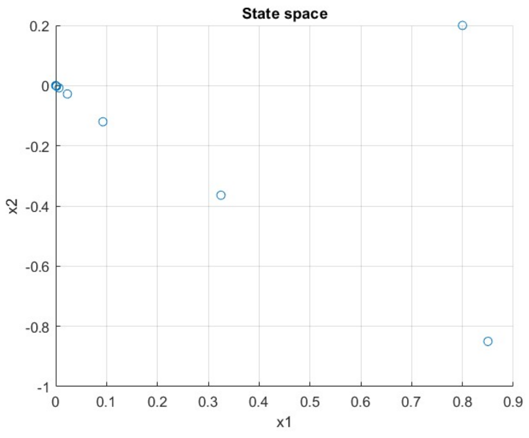


(b) $T_e = 0.5$

Figure 20: Output

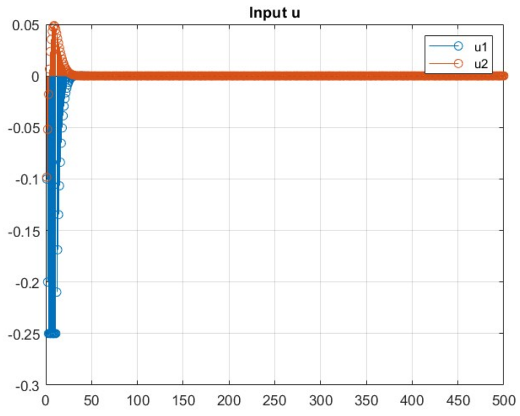


(a) $T_e = 0.1$

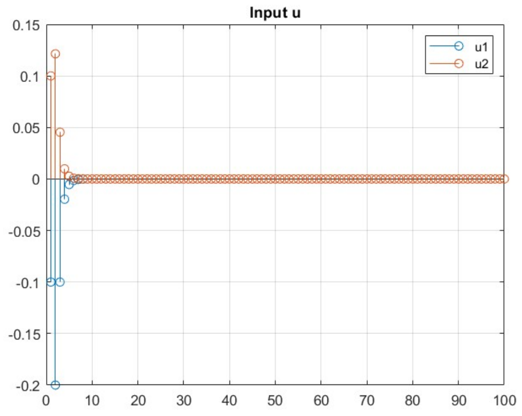


(b) $T_e = 0.5$

Figure 21: State space



(a) $T_e = 0.1$



(b) $T_e = 0.5$

Figure 22: Input

We can see that whenever we use a smaller sampling period, we would receive a more stable and robust system, with a fast response and a instant command. As we increase the sampling time period, we can observe in the states space, that the activity is not normal, because the process does not receive enough time to anticipate and understand all the states and the decision that it has to make and to imply.