Model Predictive Control
Checkpoints 3 & 4
Date: 11 October 2020

1 Checkpoint 3

1.1 Question 1

We can generalize the cost function by adding the penalty on the control:

$$V(x(k), p) = \sum_{i=1}^{N_p} |x(k+i,p) - x_d|^2 + \alpha p^2 \tag{1}$$

where α is a predefined non-negative value. After the steps as described in the course (the x_d is in $\psi_0(x(k))$):

$$V(x(k), p) = [\psi_1^T \psi_1]p^2 + [2\psi_1^T \psi_0(x(k))]p + \psi_0^2(x(k)) + \alpha p^2 \tag{2}$$

And then:

$$V(x(k), p) = [\psi_1^T \psi_1 + \alpha]p^2 + [2\psi_1^T \psi_0(x(k))]p + \psi_0^2(x(k)) \tag{3}$$

The solution will be:

$$p^{unc}(x(k)) = -\frac{\psi_1^T \psi_0(x(k))}{\psi_1^T \psi_1 + \alpha} \tag{4}$$

1.2 Question 2

Some explanations on the codes:

- The array `alphaValues` contains the values of α we need to loop across.
- The main modifications are on the solution of the optimization problem and the calculation of $\psi_0(x(k))$.

Simualtion results:

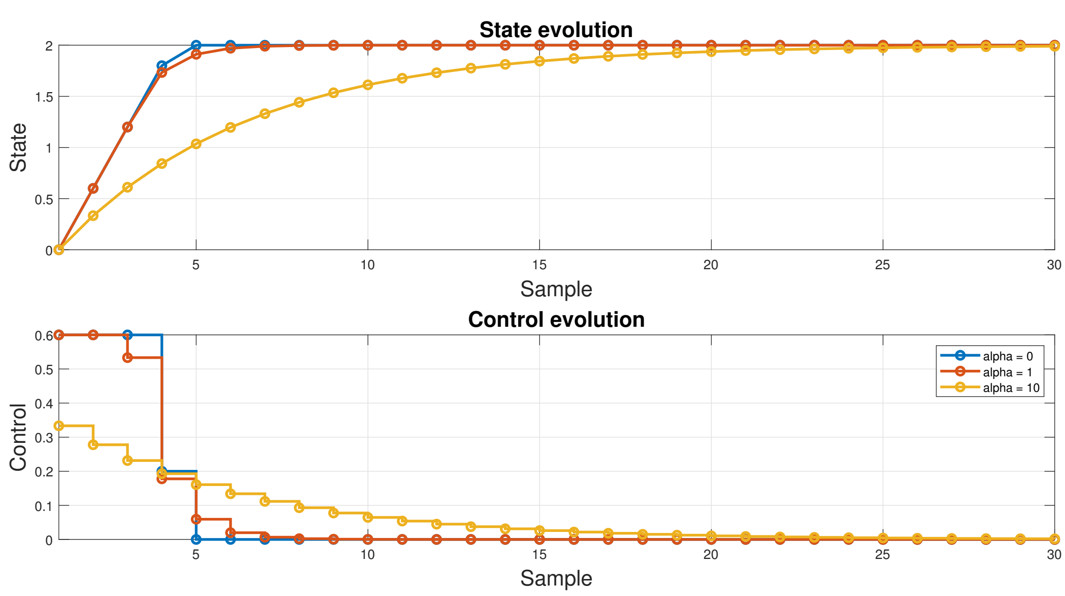

Figure 1: Simulation results for the new MPC design.

1.3 Question 3

Some comments and explanations on the results:

- Here we see a trade-off between settling time and control. The quadratic cost that we minimize now concerns both the error and the control signal. For a larger α, we put more weight and thus more attention to minimizing the control, while smaller α means the error gets more attention.

- The results show clearly that for a large α, such as 10, the control signal has a smaller magnitude. This interpretation can also be seen from the 4th equation, as α is in the denominator. So we can tune the cost function this way to save some energy/money while letting the system converge more slowly. This example shows the convenience of MPC in terms of expressing performance index.

2 Checkpoint 4

2.1 Question 1

As the `kron(A,B)` function has been programmed in MATLAB, we can use it. The result is of course the same.

2.2 Question 2

Some explanations on the codes:

- The Γ matrix `Gam` is first initiated as blank.
- The student used a variable named `lineJ` which corresponds to each line of `Gam`. At each iteration, the value of A, B, and $-I$ are put in the correct places thanks to appropriate shifting (the term `j*(nx + nu)`).
- At the end of each iteration, `lineJ` is added to the end of `Gam` and is then reset for the next iteration.

2.3 Question 3

Results of many trials:

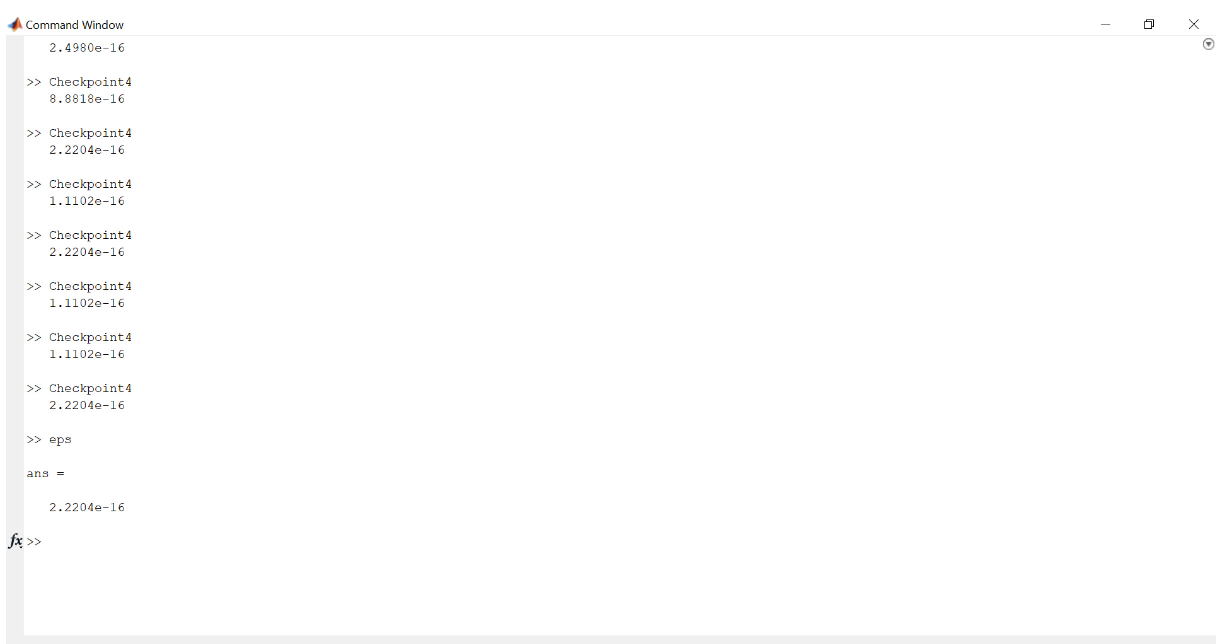

Figure 2: Validation of the LTI dynamics function.

We can see that the value of `residual` is always very small. We can compare it with `eps`, the smallest value in MATLAB that can be added to 1 i.e. $1 + \texttt{eps} = 1.0000$ and $1 + \texttt{eps}/2 = 1$. The value of `eps` is the last one. We see that `residual` is comparable to `eps` in terms of magnitude, so it is basically zero in MATLAB. That means the function has been validated.