OLCAR- Exercise 1 – ILQC

Answers to question related to programming exercise

Handout: 17.03.2015
Due: 15.04.2013

Yi Hao Ng, Raphael Stadler
(Team Number 10)

Problem 1

Design LQR controller with one goal state (2.)

Question 1: How does the controller behave with varying positions of the Task.goal_x?

For too far away goal points the LQR algorithm doesn't succeed to find optimal state and control trajectories. In this situation a path is constructed which not even passes the desired goal point, the quadrotor "crashes". The rotor speeds are proportional to the distance to the goal point. This can be observed in the trajectory in the way, that quadrotor tries to get close to the goal point as fast as possible and then adjusts itself to the exact position more slowly.

Question 2: What is the cause of the varying performance (model, cost function,...)?

The performance of the quadrotor is measured via a performance index J (total cost). The smaller this index is, the better the performance of the quadrotor is. The performance index is affected by the system dynamics/model ($\dot{x} = A \cdot x + B \cdot u$, where A and B are given by the model) and also the weighting matrices Q and R which are used to penalize deviations of the states and control inputs. Depending on the structure of A, B, Q, R the performance of the quadrotor varies.

Question 3: How do changes in Task.cost.Q_lqr and Task.cost.R_lqr affect the behavior?

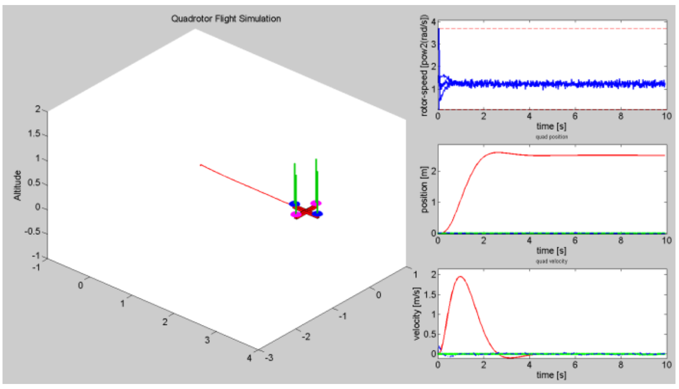
`Task.cost.Q_lqr` (further on labeled Q) penalizes the distance between the object and the goal position, while `Task.cost.R_lqr` (further on labeled R) penalizes the control effort it takes to reach the goal position. Overall it can be stated, that the behavior depends on the ratio between the magnitudes of Q and R ($m = |Q|/|R|$).

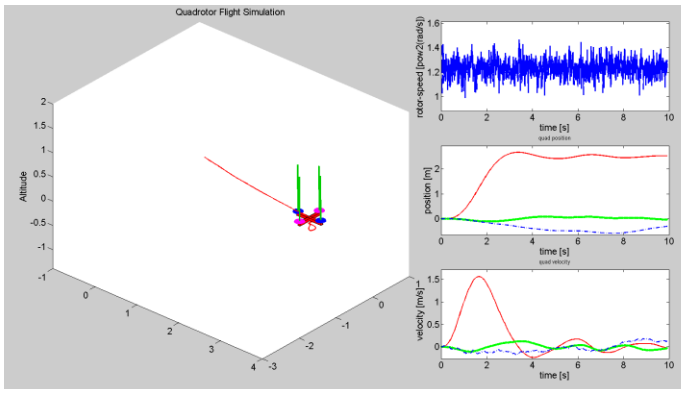
If $m \gg 1$: The quadrotor aims to reach the goal position as fast as possible, since the state deviations are penalized heavily by the huge Q weighting matrix. On the other hand it is not of any importance how big the rotor speeds are and it is possible that the maximum rotor speed can be big, which is only slightly penalized by the small weighting matrix R.

If $m \ll 1$: The controller keeps the maximum rotor speed as small as possible, since the control effort is penalizes heavily by the huge R weighting matrix. On the other hand it is not a problem that the quadrotor is not close to the goal position during the control period, since this is penalized a lot lesser.

Example: Reaching goal state (2.5,0,0) from (0,0,0)

$m \gg 1$ $m \ll 1$





Design LQR controller with a via-point (3.)

Question 4: How does a via-point change behavior of the LQR controller?
In principle the inclusion of the via-point is treated by the algorithm as two sub-problems which are themselves equivalent to the original problem in the following manner: First, the algorithm solves for the trajectories to reach the via-point until a certain time `Task.vp_time`, then it takes this via-point as a new starting point and solves for the final goal point which has to be reached in the remaining time (`Task.goal_time – Task.vp_time`) in the same way as if there weren't any via-point before.
For both sub-problems the algorithm uses the same weighting matrices Q, R, while only the corresponding goal positions differ (which affects the control input directly via θ).

Question 5: Why is the system capable of reaching further away Task.goal_x states?
For LQR it is not possible to reach goal points which are too far away. This comes from the fact, that the rotor/quadrotor speed is proportional to the distance to the goal point. Having a too large distance leads to a rotor speeds which do not allow stable control anymore - especially the speeds right after the quadrotor starts get too high.
As already mentioned in the answer of question 4, the inclusion of a via-point can be considered as solving two sub-problems. If the goal point itself is too far away from the starting point, but the via-point is "reachable" and the goal point is "reachable" from the via-point, the two sub-problems can be solved, while the original problem was not solvable.

Question 6: Defining via points Task.vp and time Task.vp_time seem to have a positive influence on the system behavior and increase stability. What are the disadvantages?
As described above, since the system treats the via-point as a second sub-problem, the inclusion of the via-point gives more information about the desired trajectory. But, an inappropriate via-point might lead to a bad performance of the system or it is even possible that the via-point is not reachable while the goal point would be. The selection of the via-point in this way influences the performance of the system directly.

Problem 2

Design the ILQC controller (1.)

Question 7: How do the trajectories of the LQR and ILQC controller differ? How do the costs compare?
As initial trajectories the ILQC algorithm takes the trajectories which are computed by LQR. These trajectories are then used to compute the ILQC cost for the first time. Then, in an iterative manner, the algorithm tries to optimize this cost.
The ILQC cost function consists of pure quadratic terms of the control and the state trajectories and also some additional linear terms of the control and state trajectories, while the LQR cost function is purely quadratic. Overall, the ILQC cost function includes the same terms as the LQR cost function but additionally has some extra terms as well. Still, the cost of the ILQC controller is smaller compared to the cost of the LQR controller. In such a way it can be stated, that ILQC produces a better solution than LQR.
As a result, the trajectories computed by ILQC are smoother (less control effort needed) and the state oscillations are smaller than the ones of LQR.

Question 8: Why does the ILQC perform better for distant and similarly well for close goal states?
The gain matrix K which is included in the LQR control is constant throughout the whole trajectory.
As described in question 5, this may lead to unstable conditions (e.g. too large initial rotor speed).
For ILQC it is not the case, that K remains constant. For ILQC the system dynamics are linearized locally at each time step. At each time step this information is then used to compute an appropriate gain matrix K. This allows the ILQC to come up with more stable configurations.

Include via-points for the ILQC controller to pass through (2.)

Question 9: How must the via-point weighting matrix Q_vp be chosen for the algorithm to determine the optimal via-point velocity on its own?
The weighting matrix has to be chosen as a 12x12 (according to the size of the state vector) diagonal matrix. All the off-diagonal elements are zero, since the differences of the different states are penalized uncorrelated to each other (meaning, that there are no mixture terms of multiple states in the weighing matrix Q_vp).
In order to allow the algorithm to determine the optimal via-point velocity on its own, the corresponding elements of the diagonal weighting matrix have to be set to zero. In our case, the last six elements on the diagonal have to be set to zero, in order to not penalize deviations in linear and angular velocities.

Question 10: How can exact passing through the via-point be enforced and how can it be included only as a soft suggestions on top of the other performance criteria?
The importance of passing through the via-point depends on the magnitude of Q_vp compared to the magnitude of the other weighting matrices (Qm_f, Qm, Rm, …). If the magnitude of Q_vp is considerably larger than the ones of the other weighting matrices, the passing through the via-point is enforced while a smaller magnitude soften this constraint.