Trajectory optimization using LQR and MPC

Kutay Demiralay
Spring 2024

1. Introduction

In this project, our aim is to optimize the trajectory of a free-flying robot with respect to given dynamics and constraints. We will be using the Linear Quadratic Regulator (LQR) method for a non-noisy flying environment, and to achieve robustness against disturbance (wind in our case), we will use the Model Predictive Control (MPC) method.

We model the dynamics of our free-flying robot using linear, double integrator dynamics.

$$\underbrace{\begin{bmatrix}\dot{p}_x\\ \dot{p}_y\\ \dot{v}_x\\ \dot{v}_y\end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}}_{A}\underbrace{\begin{bmatrix}p_x\\ p_y\\ v_x\\ v_y\end{bmatrix}}_{x} + \underbrace{\begin{bmatrix}0 & 0\\ 0 & 0\\ 1 & 0\\ 0 & 1\end{bmatrix}}_{B}\underbrace{\begin{bmatrix}a_x\\ a_y\end{bmatrix}}_{u} \tag{1}$$

And our problem parameters and constraint values are defined as follows.

Starting state:

$$\mathbf{x}_{\text{start}} = \begin{bmatrix}-4\\ 0\\ -1\\ 2\end{bmatrix}$$

Desired end state:

$$\mathbf{x}_{\text{final}} = \begin{bmatrix}0\\ 0\\ 0\\ 0\end{bmatrix}$$

Acceleration bounds:

$$\|\mathbf{u}\|_2 \leq a_{\max}$$

where

$$a_{\max} = 2$$

Number of time steps:

$$K = 50$$

Time step duration:

$$\Delta t = 0.1 \text{ seconds}$$

Optimization variables:

$$\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{K+1})$$
$$\mathbf{u} = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_K) \tag{2}$$

We will be using discrete time dynamics with time step delta t thoroughly and assume zero-order hold over the control u for simplicity.

2. Linear Quadratic Regulator (LQR) method for a non-windyflying environment

We first need to derive the exact discrete-time dynamics with time step delta t assuming zero-order hold over the control u. Continuous-time state space representation of the system is shown as:

$$\dot{x} = f(x, u) = Ax + Bu \tag{3}$$

In discrete time step we can represent our linear system as:

$$x_{k+1} = A_d \cdot x_k + B_d \cdot u_k \tag{4}$$

Where Ad can be calculated as:

$$A_d = e^{A\Delta t}$$

$$e^{A\Delta t} = I + A\Delta t + \frac{(A\Delta t)^2}{2!} + \frac{(A\Delta t)^3}{3!} + \dots$$

$$A\Delta t = \begin{bmatrix} 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$(A\Delta t)^2 = 0$$

$$A_d = e^{A\Delta t} = I + A\Delta t = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

And where Bd can be calculated as:

$$B_d = \int_0^{\Delta t} e^{A(\Delta t - \tau)} B d\tau$$

$$e^{A(\Delta t - \tau)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \Delta t - \tau & 0 & 1 & 0 \\ 0 & \Delta t - \tau & 0 & 1 \end{bmatrix} \tag{6}$$

$$B_d = \int_0^{\Delta t} \begin{bmatrix} 1 & 0 & \Delta t - \tau & 0 \\ 0 & 1 & 0 & \Delta t - \tau \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}^T d\tau = \begin{bmatrix} \frac{(\Delta t)^2}{2} & 0 \\ 0 & \frac{(\Delta t)^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \tag{7}$$

Our Objective function in LQR method can be defined as:

$$\min \sum_{k=0}^{K} \left(x_k^T Q x_k + u_k^T R u_k \right) + x_{K+1}^T Q_{\text{terminal}} x_{K+1} \tag{8}$$

Where Q and R are user defined weights. Q, Q_terminal and R matrices are present in the cost function. Q is the state weight matrix and R is the control weight matrix and Q_terminal is the final state weight matrix . With modifying these matrices we can influence the cost we attribute to states and controls. If some state or control is costly, the optimization system will try to keep that specific state or control low with respect to others, because it will have a higher effect on cost with respect to other costs, it is more valuable to preserve.

With increased control weights, we can see that the trajectory is smoother and less sharp, indicating reduced control effort for aggressive maneuvers. I aimed to find the optimal Q, Q_terminal R weights for a balanced, smooth trajectory. The best weights turned out to be:

$$Q = \text{np.diag}([1., 1., 1., 1.]) \times 3$$

$$Q_{\text{terminal}} = \text{np.eye}(\text{state\_dim}) \times 5.0$$

$$R = \text{np.diag}([1., 1.]) \times 1 \tag{9}$$

And I used the same Q, Q_terminal R weights throughout the entire project.

Now that we have a well-defined discrete-time LQR problem with a well-defined objective function and constraints, we can find the minimum point of this convex problem using Python's built-in CVXPY library.

Results:

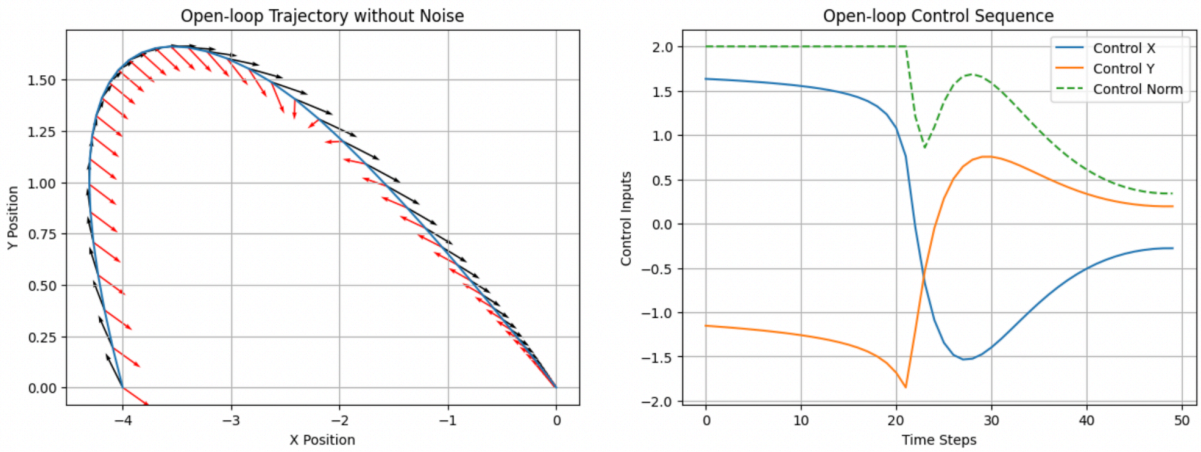

Figure 1: Trajectory and Control Sequence obtained with LQR method without disturbance

The trajectory path and control sequence using the LQR method are shown in Figure 1 above. As can be seen, the trajectory is feasible, satisfying the maximum control input constraint as well as the initial and final state constraints. This represents our optimal trajectory using LQR.

3. Model Predictive Control (MPC) for windy flying environment

In the previous step, we used the LQR method to find an optimized trajectory for a non-disturbance or non-windy environment. In real-world scenarios, there will be disturbances. Therefore, we need a robust algorithm to withstand these external disturbances. Model Predictive Control (MPC) will provide the necessary robustness in this case

Before using MPC, we want to see how well LQR works with wind disturbances in our problem. To simulate this, we add noise_cov=np.diag([0.,0.,0.1,0.1]) to our dynamics and plot the results. The disturbance is an additive factor.

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k$$

where $\mathbf{w}_k$ is Gaussian white noise, $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \Sigma)$, and

$$\Sigma = \mathrm{diag}([0.0, 0.0, 0.1, 0.1]). \tag{10}$$

Results:

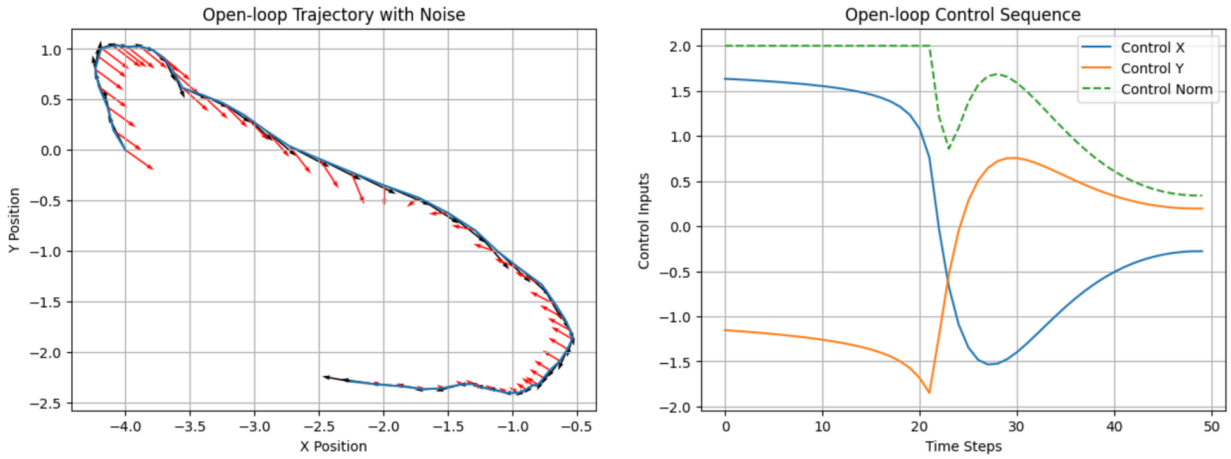

Figure 2: Trajectory and Control Sequence obtained with LQR method with disturbance

As you can see in the Figure 2 above, with the exact same control scheme as in the previous step, the path deviates significantly from the previous path in Figure 1 due to disturbance. The path starts at the point [−4,0] but does not end anywhere close to the desired final point [0,0]. The LQR is blind to wind and assumes the dynamics given to it work perfectly, showing no robustness to disturbances.

The MPC approach recalculates the control actions at each time step using a for loop, solving a finite-horizon optimization problem and using only the first control action. The finite-horizon length is selected to be 10. (Not 50, we don't want to calculate the whole horizon length at each loop) MPC Solves an optimization problem at each time step, which is computationally more demanding but provides better adaptability to changes and disturbances.

Since my MPC horizon is shorter than the actual time needed to reach the goal, you need to remove the goal state constraint and instead include a heuristic cost-to-go term in your MPC cost. Our new cost/objective function is as follows

$$\text{Objective} = \sum_{t=0}^{N_{\text{MPC}}-1} \mu^t \left(\mathbf{x}_t^T \mathbf{Q} \mathbf{x}_t + \mathbf{u}_t^T \mathbf{R} \mathbf{u}_t\right) + (\mathbf{x}_{N_{\text{MPC}}} - \mathbf{x}_{\text{goal}})^T \mathbf{Q}_t (\mathbf{x}_{N_{\text{MPC}}} - \mathbf{x}_{\text{goal}}) \tag{11}$$

Where,

$$\text{Heuristic Cost-to-Go} = (\mathbf{x}_{N_{\text{MPC}}} - \mathbf{x}_{\text{goal}})^T \mathbf{Q}_t (\mathbf{x}_{N_{\text{MPC}}} - \mathbf{x}_{\text{goal}}) \tag{12}$$

- $\mathbf{x}_t$ is the state vector at time step t.
- $\mathbf{u}_t$ is the control input vector at time step t.
- $\mathbf{Q}$ is the state weighting matrix, $\mathbf{Q} = \text{diag}([1., 1., 1., 1.]) \times 3$.
- $\mathbf{R}$ is the control input weighting matrix, $\mathbf{R} = \text{diag}([1., 1.]) \times 1$.
- μ is a discount factor.
- N_{MPC} is the length of the MPC prediction horizon.
- $\mathbf{x}_{N_{\text{MPC}}}$ is the state vector at the end of the MPC horizon.
- $\mathbf{x}_{\text{goal}}$ is the desired goal state vector.
- $\mathbf{Q}_t$ is the terminal state weighting matrix, $\mathbf{Q}_t = \mathbf{I} \times 5.0$, where $\mathbf{I}$ is the identity matrix in appropriate dimensions.

Results:

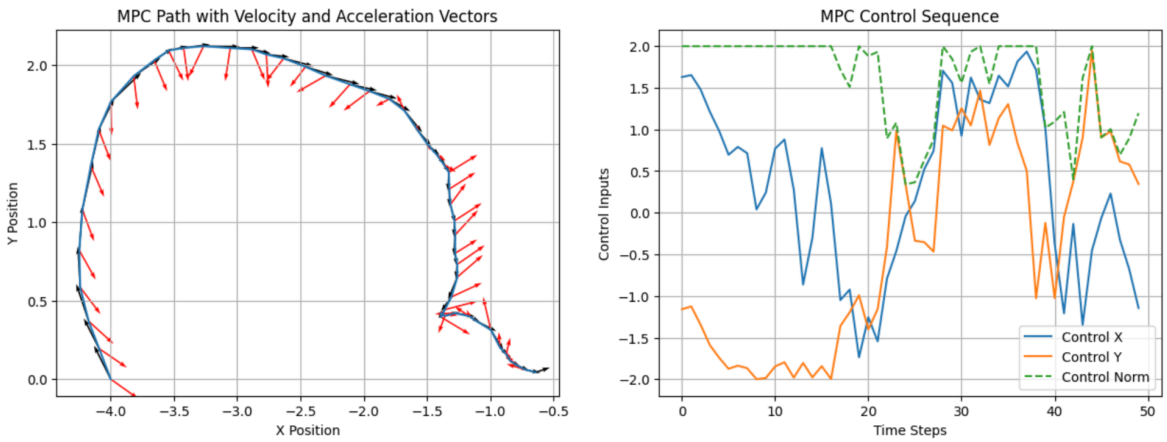

Figure 3: Trajectory and Control Sequence obtained with MPC method with disturbance

The free-flyer robot struggles to reach the goal state as you can see on the left hand side plot on Figure 3, it gets close however its final state is not exactly the desired state. Maybe if we were to loop for more than 50 time steps it would get closer, but it is not guaranteed. It would still struggle because the wind is powerful, and if you check out the left hand side plot on Figure 3 you will see the control sequence of the robot. The total control input norm, shown by green line, is equal to 2 for the most of the time, suggesting that the robot inputs are at its max capability limit throughout most of the flight time. Meaning it gives its best to hold against the wind. This suggests that more control authority would definitely help our robot. To increase the maximum input limit, we will need more powerful accelerators. But thanks to the MPC structure of our control algorithm the robot is somewhat robust against the disturbances, or wind, it clearly takes necessary control actions to uphold against the wind.

4. Conclusion

The LQR method works perfectly well in undisturbed environments, where there is no wind. It gives out a feasible solution that satisfies the constraints while minimizing the cost function. The code is not that complex; we simply plug in the quadratic convex objective term along with constraints into CVXPY to minimize it. However, in real-world conditions, there are disturbances like wind. When we add disturbances to our system, the LQR becomes non-robust against these unpredictable disturbances. Thus, we implemented MPC, which is a bit more complex algorithm. With the MPC algorithm, the robot performed well with its trajectory, getting close enough to the desired final state and giving out a feasible solution. In this project, I explored the capabilities of both LQR and MPC.