Institute of Robotics, University of Innopolis

Computational Intelligence
Model Predictive Control

May 16, 2021

We are going to simulate moving a object, typically a point, in 3D space. Assume the system dynamics is described in terms of the LTI (linear time-invariant) state-space model

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k, \end{aligned} \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, $y_k \in \mathbb{R}^{n_y}$, and A and B are given by:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

x_k and u_k describe position and velocity, respectively in x,y, and z direction in 3D space at time index i.e., k. $-6 \le x_k \le 6$ and $-0.1 \le u_k \le 0.1$ are the constraints that are being applied on the point when moving through space. Let x_{init} be the start position where the point is moved, and x_{ref} be the target position. The objective is to formulate the preceding problem as an optimization problem in which an optimal control policy should be calculated at each time instance solving the following quadratic optimization problem and apply the first potion to the system.

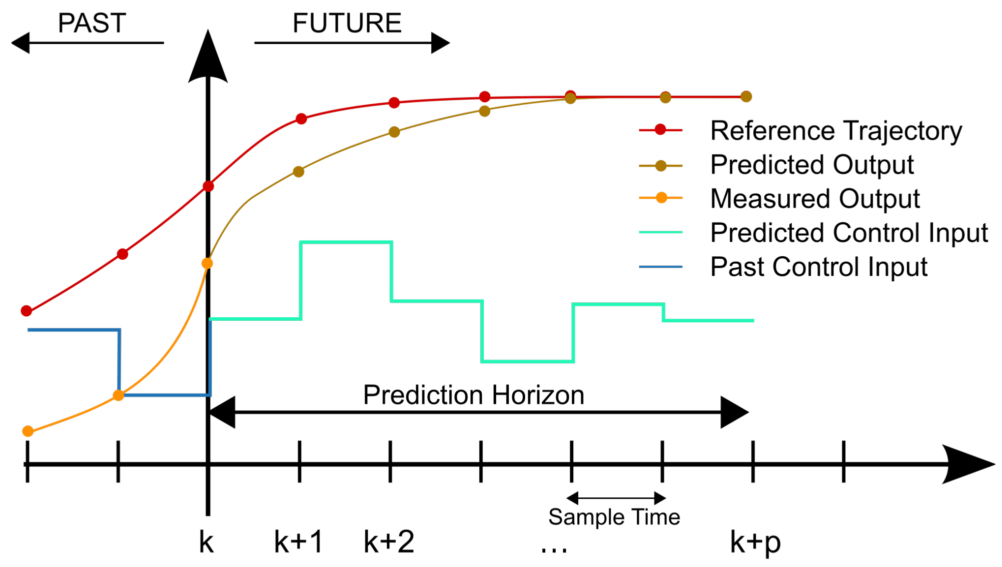

Figure 1: Model Predictive Control in a nutshell[1]

$$
\begin{aligned}
\min_{\mathbf{u},\mathbf{x}} \quad & \sum_{k=0}^{n-1}(x_k - x_{ref})^T Q (x_k - x_{ref}) + u_k^T R u_k + (x_n - x_{ref})^T Q_n (x_n - x_{ref}) \\
\text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k \\
& x_{\min} \le x_k \le x_{\max} \\
& u_{\min} \le u_k \le u_{\max} \\
& x_0 = x_{init},
\end{aligned} \tag{3}
$$

where $Q_n = Q = R = diag(1,1,1)$. You may decide a value for N appropriately. Further instructions are provided in the skeleton code. Expected output should be similar to Fig.2 for $x_{init} =< 5, 5, 1 >$ and $x_{ref} =, -2, -3., 5. >$

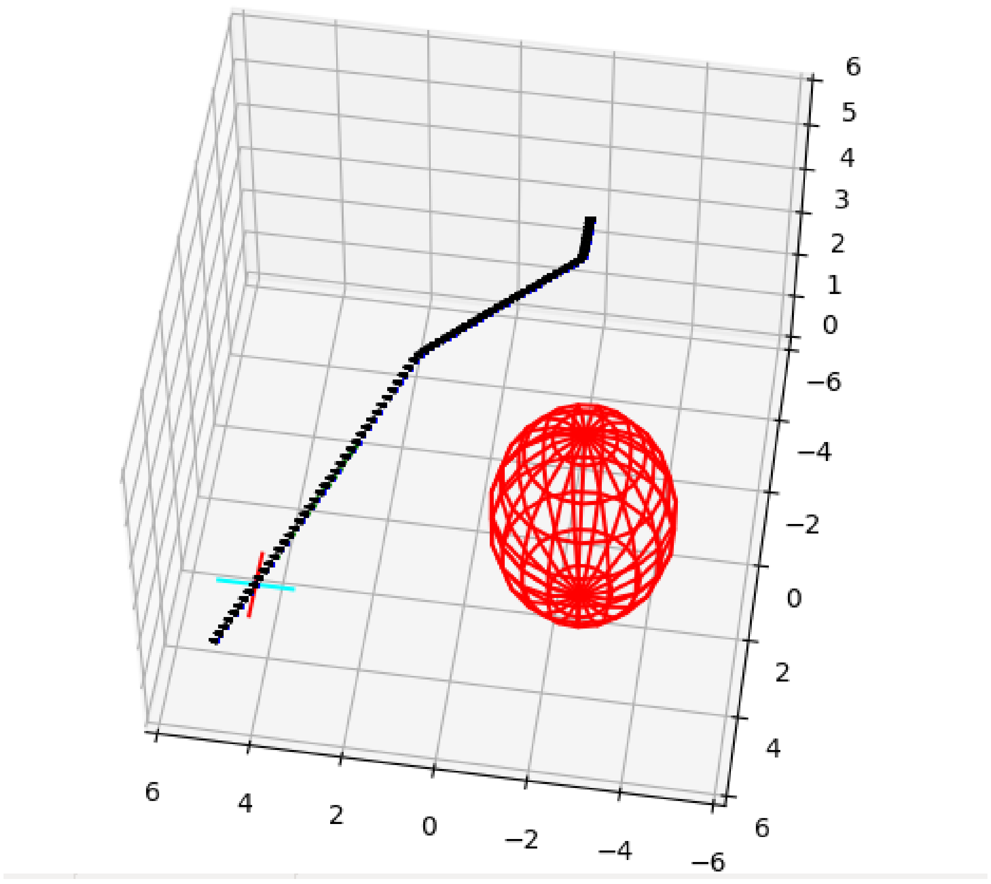

Figure 2

In order to simulate and verify your implementation, you may use the following code in which you have to stack state of the system at each time instance, i.e., stacked states are in the form (n_s, 3) matrix where n_s is the number tries solver used to get to the x_{ref}. For n_s times, predicted trajectory describes by a matrix with dimension (n_s, 21, 3). Yet, simulator is required it is be (n_s, 21, 4). You may have to to add zero column to the original matrix, i.e., $prediction\_horizon\_poses$, before constituting as input to the the simulator.

Comment on how can we incorporate a set of static obstacles as a new constraints set and heading of the moving object.

```
df = np.array(current_state)
```

```
quadcopter = QuadCopter()
def control_loop(i):
    state = np.array(df[i])
    return quadcopter.world_frame(np.append(state, 0)
                    , prediction_horizon_poses[i])

obs_map = np.array([[-2.5, 1.5, 2, 2]])
quadcopter.plot_quad_3d(control_loop, obs_map)
```

References

[1] *https://math.stackexchange.com/questions/1098070/model-predictive-control*. 2021.