Robot Mobility: Lecture 6
Lecture Notes

Victor Risager

October 30, 2023

1 Model predictive control

It is optimal control.
Can handle both states and control constraints.
A state constraint can be an obstacle. I.e. some states unattainable. Can merge motion control and path planning.
Reinforcement learning is also optimal control, but it does not handle constraints that well.
In *Model predictive control*, we need a model to predict the motion, thus, we do not predict the model.
Idea Minimize a cost function. We optimize with respect to the cost function, over a certain prediction horizon H_p. This prediction horizon is a **Tuning parameter**

1.1 Algorithm

1. Sample the system and obtain x_k.
2. Solve the finite horizon optimal control problem, which results in $U = [u_k, u_{k+1}, u_{H_p-1}]$
3. Apply u_k to the robot, which results in x_{k+1}

1.2 State space model

Consider the discrete time linear system

$$x(k+1) = Ax(k) + Bu(k) \tag{1}$$
$$y(k) = C_y x(k) \tag{2}$$
$$z(k) = C_z x(k) \tag{3}$$

where z is the output to be controlled.
Let the quadratic cost function be

$$V(k) = \sum_{i=H_w}^{H_p} ||\hat{z}(k+i|k) - r(k+i|k)||^2_{Q(i)} + \sum_{i=0}^{H_u-1} ||\Delta\hat{u}(k+i|k)||^2_{R(i)} \tag{4}$$

Where the first part is the error, between the reference signal r and the controlled output z H_w is the window parameter, which is a **tuning parameter** because it determines from where in the future we should predict from, to reduce computational requirements.
Note that Δu is the difference in control inputs and H_u is the control horizon, where $H_p \geq H_u$
Q is the running cost, which determines the weight of a given timestep. The system is still time invariant. It is depenendent on i, and it is static for every time i am calculating.

1.3 Constraints

- Actuator slew rate (Maximum rate of change of the actuator)
- Actuator range
- Controlled variable constraints.

MPC is very dependent on the model. If we have a bad model, bad control inputs are given, and vice versa.
The colums of the control variable constraints are determined by the number of constraints * the number of timesteps + 1.
(4) is minimized w.r.t. $\Delta U(k)$
Subject to

$$x(k+i+1) = Ax(k+i|k) + Bu(k+i|k) \tag{5}$$
$$y(k+i|k) = C_y x(k+i|k) \tag{6}$$
$$z(k+i|k) = C_z x(k+i|k) \tag{7}$$

and the constraints

$$E\begin{bmatrix}\Delta U(k)\\ 1\end{bmatrix} \leq 0, F\begin{bmatrix}U(k)\\ 1\end{bmatrix} \leq 0, G\begin{bmatrix}Z(k)\\ 1\end{bmatrix} \leq 0 \tag{8}$$

The problem here is that the function is optimized w.r.t ΔU and not u
So we need to reformulate the dynamics and constraints such that it is dependent on ΔU.
Now we have found the entire computation. (slide 10)
This can be collected and rewritten into a sequence of state vectors. nlopt and acado. They compute the big matrices.
Note that these matrices are often ill-conditioned.