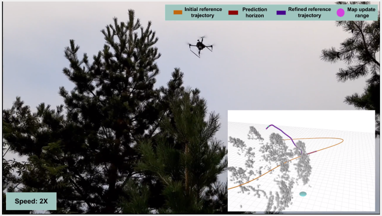
Motion Planning for Autonomous Vehicles

Model Predictive Control (MPC)

Geesara Kulathunga

April 16, 2023

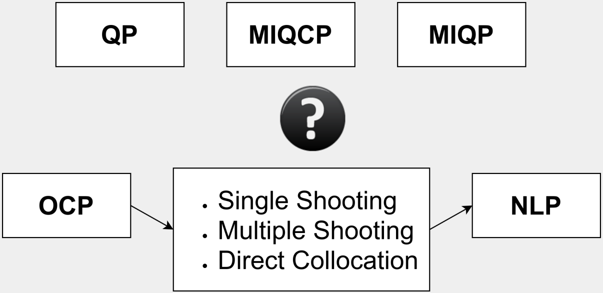


Model Predictive Control (MPC)

Contents

Ways to solve Optimal Control (OCP) Problems

An OCP problem can be transformed into an NLP problem. A problem is always solved better in a **nonlinear manner** as opposed to a **linearizing motion model** at every time since the motion model is nonlinear.



A **OCP** problem can **transform** into **NLP** in various ways, including MS (**Multiple-Shooting**) and DC (**Direct-Collocation**).

OCP Using Nonlinear Programming Problem (NLP)

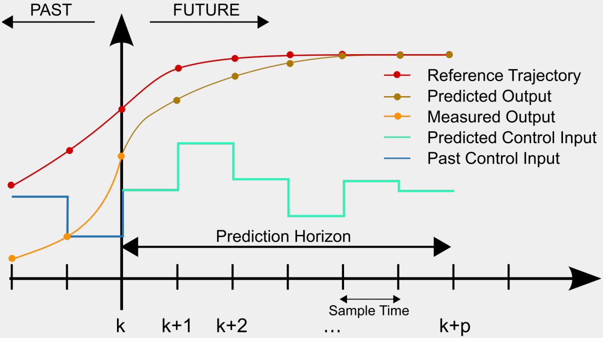
OCP

$$
\begin{aligned}
\min_{u} \quad & J_n(x_0,u) = \sum_{k=0}^{n-1} c(x(k),u(k)) \\
\text{s.t.} \quad & x_{k+1} = f(x(k),u(k)) \\
& x(0) = x_0 \\
& u(k) \in U \, \forall k \in [0,n-1],\; u \in U \subseteq \mathbf{R}^{n_u} \\
& x(k) \in X \, \forall k \in [0,n],\; x \in X \subseteq \mathbf{R}^{n_x}
\end{aligned}
$$

NLP

$$
\begin{aligned}
\min_{w} \quad & \varphi(F(w,x_0,t_k),w) \\
\text{s.t.} \quad & x_{k+1} = f(x(k),u(k)) \\
& g_1(F(w,x_0,t_k),w) \leq 0 \\
& g_2(F(w,x_0,t_k),w) = 0
\end{aligned}
$$

In general, use the specified **model** to **predict** the motion of the system, **generate** a **locally optimal or feasible trajectory**, and **repeat** the procedure



Simon, D. (2014). Model Predictive Control in Flight Control Design: Stability and Reference Tracking (Doctoral dissertation, Linköping University Electronic Press).

Prediction Simulate states forward in time up to a defined horizon, **prediction horizon**, N_e from the **current state**

$$\underbrace{\mathbf{u}_k = \begin{bmatrix} u_{0|k} \\ u_{1|k} \\ \vdots \\ u_{N_e-1|k} \end{bmatrix}}_{\text{control inputs}} \xrightarrow{\text{estimate or calculate}} \underbrace{\mathbf{x}_k = \begin{bmatrix} x_{1|k} \\ x_{2|k} \\ \vdots \\ x_{N_e|k} \end{bmatrix}}_{\text{state vector}} \tag{1}$$

where $x_{i|k}$ denoted, current state x_k and $x_{i|k}$ denoted, i steps into the future, same for control as well

The **prediction** by **minimizing** a **stage cost**

$$J_{N_e}(x_k, \mathbf{u}_k) = \sum_{h=0}^{N_e} \left\| \mathbf{x}_{k+h} - \mathbf{x}_{k+h}^{ref} \right\|_Q^2 + \left\| \mathbf{u}_{k+h} - \mathbf{u}_{k+h}^{ref} \right\|_R^2$$

This can be solved numerically to estimate optimal $\mathbf{u}_k^*$

$$\mathbf{u}_k^* = \min_{\mathbf{u}} \quad J_{N_e}(x_k, \mathbf{u}_k), \quad Q \in \mathbb{R}^{n_x \times n_x} \succeq 0, \quad R \in \mathbb{R}^{n_u \times n_u} \succ 0$$

$$\begin{aligned} \text{s.t.} \quad & g_1(\mathbf{u}) = 0, \quad g_2(\mathbf{u}) \leq 0 \\ & p^{lower} \leq \mathbf{x}_{k+h} \leq p^{upper} \quad \forall 0 \leq h \leq N_e \\ & u^{lower} \leq \mathbf{u}_{k+h} \leq u^{upper} \quad \forall 0 \leq h \leq N_e - 1, \end{aligned}$$

Apply the **first** element of $\mathbf{u}_k^*$ on the **system** and **repeat** the optimization

- The **model** can be defined in **various ways**, multivariable, linear, or nonlinear, deterministic, stochastic or fuzzy
- Can handle **different types of constraints**, e.g., linear, quadratic, and nonlinear
- **Near-optimal** control inputs
- , however, requires **online optimization** that may be costly

- If the plant **model** is linear, the model's **state depends** linearly on **control** inputs, i.e., $x_{k+1} = f(x_k, u_k)$. Hence, **cost**, in general, is quadratic in u_k subject to linear **constraints**. Such problems can be formulated as a convex quadratic program and **guaranteed** to have **global optimal solution** all the time.

$$\min_{\mathbf{u}} \mathbf{u}^\top R \mathbf{u} + 2r^\top \mathbf{u} \quad s.t \; A\mathbf{u} \leq b$$

$$\underset{x}{\text{minimize}} \quad f(x)$$

where $f(x) = \frac{1}{2}x^\top Qx + b^\top x + c$, where $c \in \mathbb{R}, x \in \mathbb{R}^2$, and Q is 2×2 matrix. First order necessary condition $\Delta f(x) = 0$

$$\begin{aligned} df &= \frac{1}{2}x^\top Q^\top dx + \frac{1}{2}x^\top Q dx + b^\top dx \\ &= \underbrace{\left(x^\top \frac{Q^\top + Q}{2} + b^\top\right)}_{d\dot{f}(x) = \Delta f(x)} dx \end{aligned} \tag{2}$$

Since $Q^\top = Q$, $\Delta f(x) = Qx + b$. Hence, the critical point: $Qx = -b$
Second order necessary condition $\Delta^2 f(x) = Q$. It can be either **minimum**, **maximum**, saddle point, or **singular** point, i.e., **at least one eigenvalue becomes zero**.

- If the plant **model** is nonlinear, the model's **state depends** non-linearly on **control** inputs, i.e., $x_{k+1} = Ax_k + Bu_k$. Hence, **cost**, in general, is nonconvex in u_k subject to convex and nonconvex **constraints**. Such problems are formulated as a nonlinear program and **does not guarantee** to have **global optimal solution** all the time. Therefore, the solution can have local minima, locally optimal, and may not be solved efficiently or reliably

$$\min_{\mathbf{u}} J(x_k, \mathbf{u}) \quad s.t\; g(x_k, \mathbf{u}) \leq 0$$

Model Predictive Control: Prediction model

- **Discrete-time** necessary to have sampling interval δ, piecewise optimization is carried out
- **Continuous time** not necessary to have sampling interval δ, nor piece wise optimization is carried out. Can be linearized, good for nonlinear continuous-time systems

- **Hard** constraints are **satisfied** all the time, it is **not possible** to satisfy, the **problem** is **infeasible**
  - Box constraints

$$
\begin{aligned}
p^{lower} \leq \mathbf{x}_{k+h} \leq p^{upper} \quad & \forall 0 \leq h \leq N_e \\
u^{lower} \leq \mathbf{u}_{k+h} \leq u^{upper} \quad & \forall 0 \leq h \leq N_e - 1,
\end{aligned} \tag{3}
$$

  - System dynamics constraints

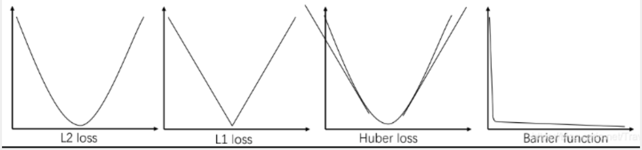
$$
g_1(\mathbf{w}) = \begin{bmatrix} \bar{\mathbf{x}}_k - \mathbf{x}_k \\ \vdots \end{bmatrix}. \tag{4}
$$

- **Soft** constraints may be **violated** to **avoid infeasibility**
  Consider the following **hard-constraints** optimization problem

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & g_1(x) = c, \quad g_2(x) \leq d \end{aligned}$$

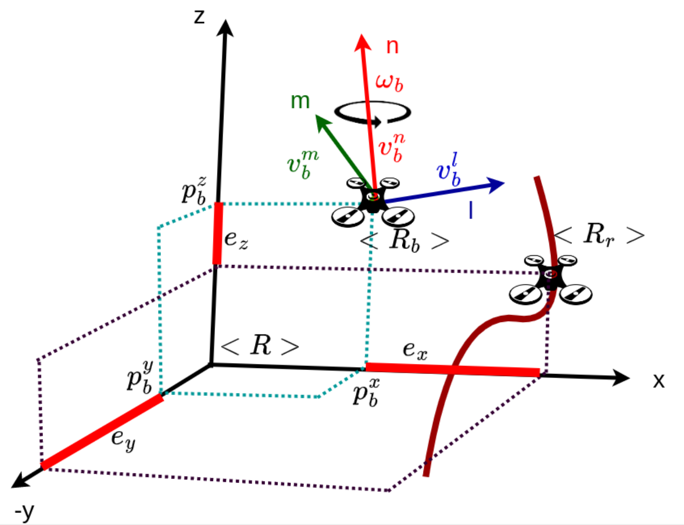
  Can be **converted** into a **soft constraints optimization problem**

$$\min_{x} \quad f(x) + \lambda_1 g_1(x) + \lambda_2 g_2(x)$$



  by using penalty terms or loss functions

Reference trajectory tracking



Simplified Motion Model

The system states $\mathbf{x}_k = [p_k^x, p_k^y, p_k^z, \alpha_k^z]^T \in \mathbb{R}^{n_x}$ and control inputs $\mathbf{u}_k = [v_k^x, v_k^y, v_k^z, \omega_k^z]^T \in \mathbb{R}^{n_u}$, where p_k^i and $v_k^i, i \in \{x, y, z\}$ denotes the quadrotor center position(m) and velocity (m/s) in each direction, i.e., x,y,z, at time t = k in the world coordinate frame; α_k^z and ω_k^z denote the yaw angle (rad) and yaw rate (rad/s) around the z-axis, respectively.
The simplified motion model is expressed by $\dot{\mathbf{x}}_k = \mathbf{f}_c(\mathbf{x}_k, \mathbf{u}_k)$

$$\dot{\mathbf{x}}_k = \begin{bmatrix} \dot{p}_k^x \\ \dot{p}_k^y \\ \dot{p}_k^z \\ \dot{\alpha}_k^z \end{bmatrix} = \begin{bmatrix} v_k^x cos(\alpha_k^z) - v_k^y sin(\alpha_k^z) \\ v_k^x sin(\alpha_k^z) + v_k^y cos(\alpha_k^z) \\ v_k^z \\ \omega_k^z \end{bmatrix}, \tag{5}$$

where $\mathbf{f}_c(\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ and $n_x = n_u = 4$

Simplified Motion Model

Forward Euler discretization, $\mathbf{x}_{k+1} = \mathbf{f}_d(\mathbf{x}_k, \mathbf{u}_k)$ is introduced for a given sampling period in seconds, $\delta \in \mathbb{R} > 0$, e.g., $\delta = 0.1s$

$$\mathbf{x}_{k+1} = \begin{bmatrix} p_k^x \\ p_k^y \\ p_k^z \\ \alpha_k^z \end{bmatrix} + \delta \begin{bmatrix} v_k^x cos(\alpha_k^z) - v_k^y sin(\alpha_k^z) \\ v_k^x sin(\alpha_k^z) + v_k^y cos(\alpha_k^z) \\ v_k^z \\ \omega_k^z \end{bmatrix}, \tag{6}$$

where $\mathbf{f}_d(\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$.

With Multiple Shooting

$$
\begin{aligned}
J_{N_e}(\mathbf{x}_k, \mathbf{u}_k) &= \sum_{h=0}^{N_e} \left\| \mathbf{x}_{k+h} - \mathbf{x}_{k+h}^{ref} \right\|_Q^2 + \left\| \mathbf{u}_{k+h} - \mathbf{u}_{k+h}^{ref} \right\|_R^2 \\
\min_{\mathbf{w}} \quad & J_{N_e}(\mathbf{x}_k, \mathbf{u}_k), \quad Q \in \mathbb{R}^{n_x \times n_x} \succeq 0, \quad R \in \mathbb{R}^{n_u \times n_u} \succ 0 \\
\text{s.t.} \quad & g_1(\mathbf{w}) = 0, \quad g_2(\mathbf{w}) \leq 0 \\
& p^{lower} \leq \mathbf{x}_{k+h} \leq p^{upper} \quad \forall 0 \leq h \leq N_e \\
& u^{lower} \leq \mathbf{u}_{k+h} \leq u^{upper} \quad \forall 0 \leq h \leq N_e - 1,
\end{aligned}
\tag{7}
$$

where $\mathbf{w} = [\mathbf{u}_k, \ldots, \mathbf{u}_{k+N_e-1}, \mathbf{x}_k, \ldots, \mathbf{x}_{k+N_e}]$ denotes the decision variables set to be minimized.

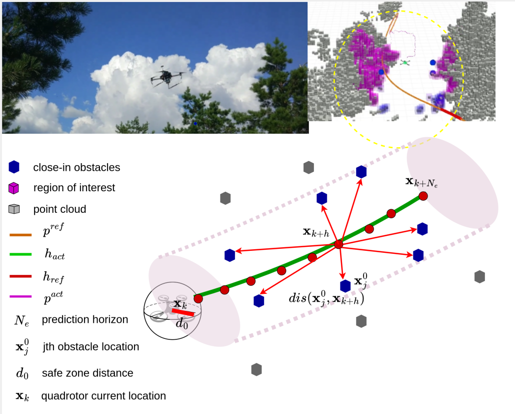
Notations u^{lower} and u^{upper} define the minimum and maximum linear and angular velocities allowed

Term $g_1(\mathbf{w})$ depicts the constraints that system dynamics imposes as follows:

$$g_1(\mathbf{w}) = \begin{bmatrix} \bar{\mathbf{x}}_k - \mathbf{x}_k \\ \vdots \\ f_d(\mathbf{x}_{k+h}, \mathbf{u}_{k+h}) - \mathbf{x}_{k+h+1} \\ \vdots \\ f_d(\mathbf{x}_{k+N_e-1}, \mathbf{u}_{k+N_e-1}) - \mathbf{x}_{k+N_e} \end{bmatrix}. \tag{8}$$

With Multiple Shooting

Reconstructing obstacle constraints in each iteration is necessary to **incorporate the dynamic environment changes** into the trajectory tracker



With Multiple Shooting

Term $g_2(\mathbf{w})$ describes the constraints imposed by obstacles.

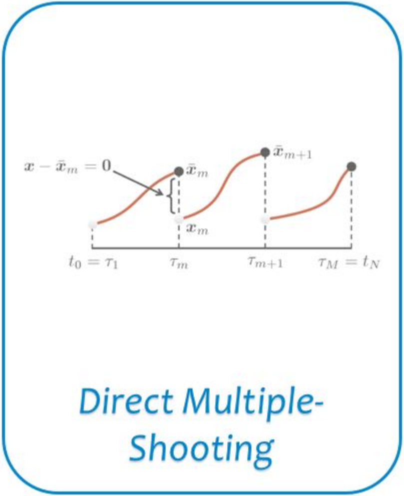
$$g_2(\mathbf{w}) = \begin{bmatrix} dis(\mathbf{x}_j^o, \mathbf{x}_k) \\ \vdots \\ dis(\mathbf{x}_j^o, \mathbf{x}_{k+h}) \\ \vdots \\ dis(\mathbf{x}_j^o, \mathbf{x}_{k+N_e}) \end{bmatrix}, \; j = 1, \ldots, N_o, \tag{9}$$

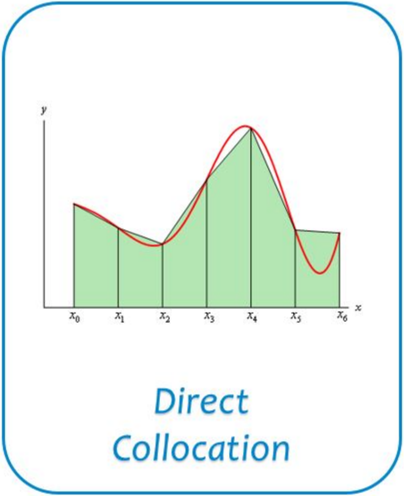
where $\bar{\mathbf{x}}_k$ is the initial position and N_o is the number of obstacles, and $dis(\mathbf{x}_j^o, \mathbf{x}_{k+h})$ can be calculated as follows:

$$-\sqrt{(x_j^o - x_{k+h})^2 + (y_j^o - y_{k+h})^2 + (z_j^o - z_{k+h})^2} + d^o$$

where d^o is the safe zone distance between the robot and close-in obstacles

Multiple shooting vs Direct Collocation



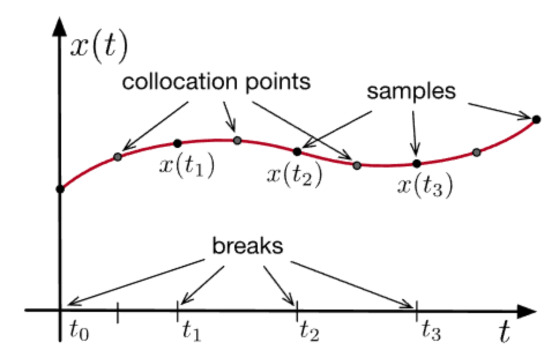


http://www.ee.ic.ac.uk/ICLOCS/Overview.html

With Direct Collocation (DC)

- **Multiple shooting**: **nonlinearity** with a sparsity structure to reduce the nonlinearity
- **Direct collocation**: add more degrees of freedom. Thus, exploits even more, but computation power increases dramatically
- **Collocation points** with respect to a **chosen polynomial**: Lagrangian 3rd order (N_d) polynomial, B-spline or Bézier
- **Fixed time interval** in multiple-shooting, but in DC, it has more **freedom** to determine how should **define points between two consecutive time interval**

- Kept the same discretization as in the multiple-shooting, i.e., $u(t) = u_k$, for $t \in [t_k, t_{k+1}], \quad k = 0, .., N_e - 1$, where N_e is the prediction horizon length

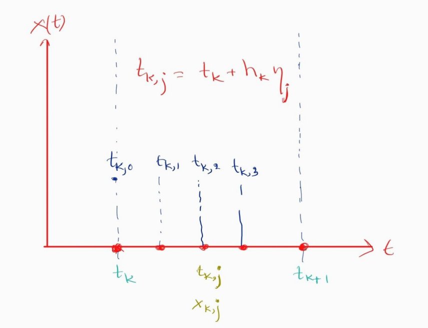


https://underactuated.csail.mit.edu/trajopt.html

- Consecutive time interval (t_k and t_{k+1}) is divided into small sub-intervals

$$t_{k,j} := t_k + h_k \eta_j, \quad k = 0, \dots, N_e - 1,\ j = 0, \dots, N_d$$

where Legendre points of order $N_d = 3$
$\eta = [0, 0.112, 0.500, 0.888]$ and $h_k = t_{k+1} - t_k$ and $x_{k,j}$ denote the states at $t_{k,j}$



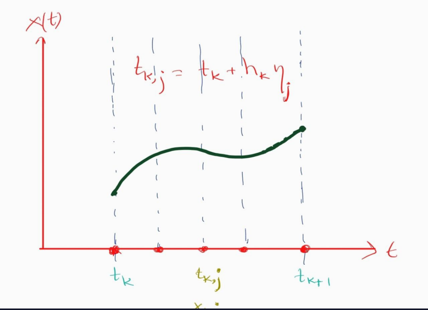
With Direct Collocation (DC)

- In each control interval, the Langrangian polynomial is defined as

$$L_j(\eta) = \prod_{r=0, r\neq j}^{N_d} \frac{\eta - \eta_r}{\eta_j - \eta_r}$$

with property

$$L_j(\eta) = \begin{cases} 1, & \text{if j =r} \\ 0, & \text{otherwise} \end{cases}$$



- State trajectory can be approximated using these basis functions

$$\bar{x}_k(t) = \Sigma_{r=0}^{N_d} L_r\left(\frac{t - t_k}{h_k}\right)x_{k,r}$$

Also, state at the end of the control interval

$$\bar{x}_{k+1}(t) = \Sigma_{r=0}^{N_d} L_r\left(1\right)x_{k,r}$$

And state time derivative at each collocation point except η_0

$$\dot{\bar{x}}_k(t) = \frac{1}{h_k}\Sigma_{r=0}^{N_d} \dot{L}_r\left(\eta_j\right)x_{k,r} := \frac{1}{h_k}\Sigma_{r=0}^{N_d} C_{r,j}x_{k,r}$$

- Hence, these collocation equations that necessary to satisfy every state at every collocation point

$$h_k f_c(x_{k,j}, u_k) - \Sigma_{r=0}^{N_d} C_{r,j} x_{k,r} = 0, \quad k = 0, ..., N_e - 1, \quad j = 0, ..., N_d$$

And the approximation of the end state

$$x_{k+1}(t) - \Sigma_{r=0}^{N_d} L_r(1) x_{k,r} = 0 \quad k = 0, ..., N_e - 1$$