

# Model-predictive control

## Computational Intelligence, Extra 1

by Sergei Savin

Spring 2024

- Dynamical systems, linearization, discretization
- Trajectory planning
- OCP / MPC for DT-LTV
- Zonotopes and explicit MPC
  - ▶ Definition
  - ▶ Operations
  - ▶ Containment, convex hull
  - ▶ MPC

Given a dynamical system described as an ODE:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \quad (1)$$

we can find a pair of time functions - control law  $\mathbf{u}^* = \mathbf{u}^*(t)$  and state trajectory  $\mathbf{x}^* = \mathbf{x}^*(t)$ , that conform with the ODE (1).

The problem of finding such pair is called *trajectory planning*.

## Example

A forced spring-damper dynamics is described with the following state-space model:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k & -\mu \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ u \end{bmatrix}$$

Once we found control law  $\mathbf{u}^* = \mathbf{u}^*(t)$  and state trajectory  $\mathbf{x}^* = \mathbf{x}^*(t)$  we can find *linearization* of the dynamical system, via Taylor expansion:

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) \approx \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x} - \mathbf{x}^*) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u} - \mathbf{u}^*) \quad (2)$$

We denote Jacobian matrices as  $\mathbf{A}$  and  $\mathbf{B}$ , giving as *state matrix* and *control matrix*

$$\mathbf{A}(t) = \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}, \quad (3)$$

$$\mathbf{B}(t) = \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}, \quad (4)$$

...for a time-varying linear system (LTV):

$$\dot{\mathbf{e}} = \mathbf{A}(t)\mathbf{e} + \mathbf{B}(t)\mathbf{u} \quad (5)$$

Linear system  $\dot{\mathbf{e}} = \mathbf{A}(t)\mathbf{e} + \mathbf{B}(t)\mathbf{u}$  is represented as a *continuous-time* model. We could re-write it as a discrete time model:

$$\mathbf{z}_{i+1} = \bar{\mathbf{A}}_i \mathbf{z}_i + \bar{\mathbf{B}}_i \mathbf{u}_i \quad (6)$$

This can be done via a number of methods (with different assumptions), such as *zero-order hold* or by exact discretization.

We can formulate the trajectory planning as an *optimal control problem* (OCP):

$$\begin{aligned}
 &\underset{\mathbf{x}(t), \mathbf{u}(t)}{\text{minimize}} && l_f(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} l(\mathbf{x}(t), \mathbf{u}(t)) dt, \\
 &\text{subject to:} && \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \\
 &&& \mathbf{x}(t_0) = \mathbf{x}_0 \\
 &&& \mathbf{x}(t) \in \text{workspace} \\
 &&& \mathbf{u}_{\min} \leq \mathbf{u}(t) \leq \mathbf{u}_{\max}
 \end{aligned} \tag{7}$$

This is an optimization problem with continuous variables and there are no solvers that can solve it (in a general case). We can replace the continuous time variables with a finite number of parameters. This is called *transcription*.

Given a discrete linear system  $\mathbf{x}_{i+1} = \mathbf{A}_i \mathbf{x}_i + \mathbf{B}_i \mathbf{u}_i$  and a quadratic objective, we can formulate the OCP as a quadratic program:

$$\begin{aligned}
 & \underset{\mathbf{x}_i, \mathbf{u}_i}{\text{minimize}} && \mathbf{x}_N^\top \mathbf{Q}_N \mathbf{x}_N + \sum_{i=1}^{N-1} \left( \mathbf{x}_i^\top \mathbf{Q}_i \mathbf{x}_i + \mathbf{u}_i^\top \mathbf{R}_i \mathbf{u}_i \right), \\
 & \text{subject to:} && \mathbf{x}_{i+1} = \mathbf{A}_i \mathbf{x}_i + \mathbf{B}_i \mathbf{u}_i \\
 & && \mathbf{F}_i \mathbf{x}_i \leq \mathbf{h}_i \\
 & && \mathbf{u}_{\min} \leq \mathbf{u}_i \leq \mathbf{u}_{\max}
 \end{aligned} \tag{8}$$

This can be solved efficiently; setting  $N$  to be relatively small we make it into a *receding horizon model-predictive control* (MPC).

Solution to (8) gives us a sequence of control actions  $\mathbf{u}_i$ .

- If problem (8) has no inequality constraints, it can be solved analytically.
- Robust control (uncertain input or uncertain model) with MPC based on quadratic program is easy to implement.
- Linear-quadratic regulator (LQR) problems are closely related. LQR allows us to obtain linear control law  $\mathbf{u} = \mathbf{K}\mathbf{x}$ . LQR does not depend on knowing exact initial conditions and can be solved as a single SDP (semidefinite program).
- Given a trajectory of a non-linear system, we can linearize its dynamics along the trajectory, and then find control law either via MPC or LQR.



# Zonotopes and explicit MPC

A zonotope can be defined as a point-symmetric set in  $n$ -dimensional space, described as a center  $c \in \mathbb{R}^n$  and  $p$  generators  $g^{(i)} \in \mathbb{R}^n$ ,  $i \in \{1, \dots, p\}$ ; the latter can be presented as columns of matrix  $G \in \mathbb{R}^{n \times p}$ :

$$\mathbb{Z} = \langle c, G \rangle = \left\{ c + \sum_{i=1}^p \beta_i g^{(i)} : -1 \leq \beta_i \leq 1 \right\} \quad (9)$$

where  $\beta_i$  are scalar multipliers. Eq. (9) defines a *vector zonotope*.

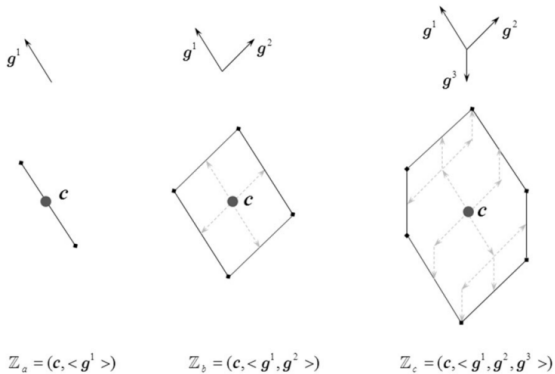


Figure 1: An example of a zonotope; [source](#).

Vector zonotopes are closed under addition and linear transformation:

$$\langle x, X \rangle + \langle y, Y \rangle = \langle x + y, X + Y \rangle \quad (10)$$

$$A \langle c, G \rangle = \langle Ac, AG \rangle, \quad (11)$$

where  $A$  is a linear operator.

Minkowski sum for vector zonotopes is defined as:

$$\langle x, X \rangle \oplus \langle y, Y \rangle = \langle x + y, (X, Y) \rangle \quad (12)$$

where notation  $(X, Y)$  refers to horizontal matrix concatenation.

We can define the addition of a vector zonotope and a vector as follows:

$$\langle c, G \rangle + v = \langle c + v, G \rangle \quad (13)$$

Given two zonotopes  $\mathbb{X} = \langle x, X \rangle$  and  $\mathbb{Y} = \langle y, Y \rangle$ , where  $X \in \mathbb{R}^{n \times n_x}$ ,  $Y \in \mathbb{R}^{n \times n_y}$ , if there exists  $\Gamma \in \mathbb{R}^{n_y \times n_x}$  and  $\beta \in \mathbb{R}^{n_y}$ , such that:

$$X = Y\Gamma, \quad y - x = Y\beta, \quad \|(\Gamma, \beta)\|_\infty \leq 1 \quad (14)$$

then zonotope  $\mathbb{X}$  is contained in zonotope  $\mathbb{Y}$ . This method is convenient, as it requires solving a single linear program.

The following is an approximation of the convex hull of two zonotopes  $\mathbb{X} = \langle x, X \rangle$  and  $\mathbb{Y} = \langle y, Y \rangle$ :

$$\text{Co}(\mathbb{X}, \mathbb{Y}) \subseteq \left\langle \frac{x+y}{2}, \left( \frac{X+Y}{2}, \frac{x-y}{2}, \frac{X-Y}{2} \right) \right\rangle \quad (15)$$

This method is conservative and computationally inexpensive. Most importantly for our purpose, it can be incorporated in a convex optimization problem formulation as an equality constraint.

We can describe how the state changes with respect to time:

$$x_{k+1} = A_k x_k + B_k u_k + d_k + w_k \quad (16)$$

where  $w_k \in \mathbb{W}_k$  is random external disturbance.



State propagation problem can be described as an evolution (propagation) of the initial zonotope  $\mathbb{X}_1$ :

$$\mathbb{X}_{k+1} = (A_k \mathbb{X}_k + B_k \mathbb{U}_k + d_k) \oplus \mathbb{W}_k \quad (17)$$

where  $\mathbb{X}_k = \langle \bar{x}_k, G_k \rangle$ ,  $\mathbb{U}_k = \langle \bar{u}_k, \theta_k \rangle$ , and  $\mathbb{W}_k = \langle 0_{n \times 1}, W(k) \rangle$  are zonotopes representing state, control actions, and process noise,  $G_k \in \mathbb{R}^{n \times p}$ ,  $\theta_k \in \mathbb{R}^{m \times p}$ , and  $W(k) \in \mathbb{R}^{n \times n_w}$  are generators or these zonotopes,  $\bar{x}_k \in \mathbb{R}^n$  and  $\bar{u}_k \in \mathbb{R}^m$  are their centers.

We can re-write the zonotope propagation as:

$$\begin{cases} G_{k+1} = (A_k G_k + B_k \theta_k, W_k) \\ \bar{x}_{k+1} = A_k \bar{x}_k + B_k \bar{u}_k + d_k \end{cases} \quad (18)$$

Disturbance  $w_k \in \mathbb{W}_k$  act on the system on each time step; therefore the order of the zonotopes  $\mathbb{X}_k$  grows on each time step, unless order reduction techniques are employed.

We know that state is given as  $x = \bar{x}_k + G_k \beta$  and control is  $u = \bar{u}_k + \theta_k \beta$ . Expressing  $\beta$  from the first and substituting it to the second, we find control law as  $u_k = K_k(x_k - \bar{x}_k) + \bar{u}_k$ , where  $K_k = \theta_k G_k^+$ .

Lecture slides are available via Github, links are on Moodle:

[github.com/SergeiSa/Computational-Intelligence-2024](https://github.com/SergeiSa/Computational-Intelligence-2024)

