

MODEL PREDICTIVE CONTROL

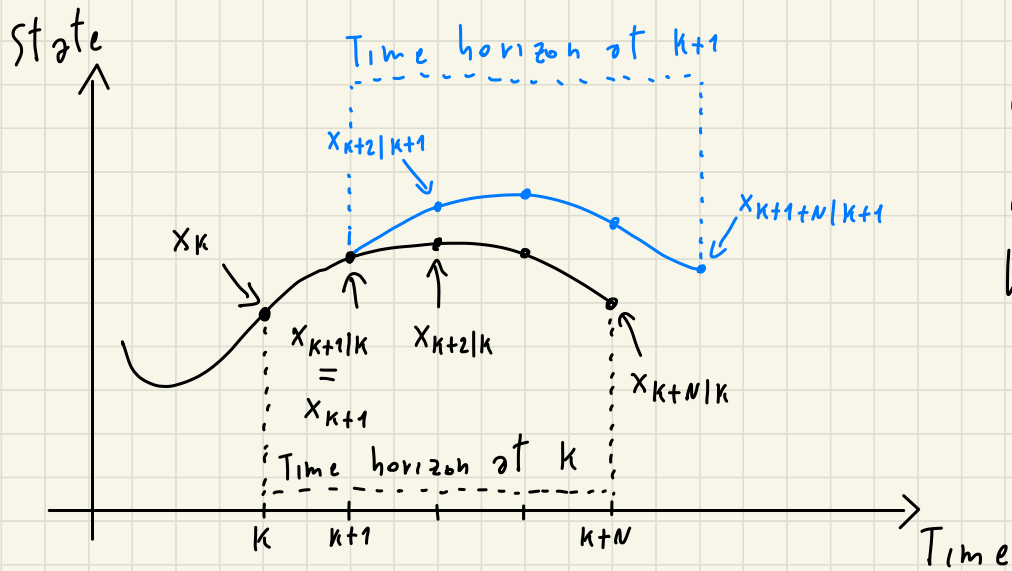
- Use optimal control for controlling system (robot)
- Solve a finite-horizon OCP using **current state** as initial state

$$X^*, U^* = \underset{X, U}{\operatorname{argmin}} \sum_{i=0}^{N-1} l(x_i, u_i)$$

$$\begin{aligned} \text{subject to } & x_{i+1} = f(x_i, u_i, k) \quad i = 0 \dots N-1 \\ & x_{i+1} \in \mathcal{X}, \quad u_i \in \mathcal{U} \quad i = 0 \dots N-1 \\ & x_0 = x^{\text{meas}} \end{aligned}$$

- Apply computed control for this time step: $\tau = u_0^*$
- Do not use the rest of U^* , nor X^*
- Repeat the same process at next control loop

PREDICTED VS ACTUAL TRAJECTORIES



OCP at time k and $k+1$
optimize over different
horizons so can result in
different trajectories

Even without disturbances, predicted and actual trajectories can be different!

CHALLENGES IN MPC

① FEASIBILITY

- Can I ensure the OCP is always feasible?
- What can I do if it's not feasible?

② STABILITY

- Can I ensure MPC stabilizes the system?

③ COMPUTATION TIME

- Can I solve the OCP sufficiently fast?

INFINITE - HORIZON MPC

- If $N = \infty$ the time horizon seen by the solved OCP, is the same
- In this case predicted and actual trajectories are the same (assuming no disturbances)
- This follows from Bellman's optimality principle:

Given optimal traj. $x^*(0) \dots x^*(N)$, the subtraj. $x^*(k) \dots x^*(N)$ is optimal for OCP on horizon $[k, N]$ starting from $x(k)$

INFINITE HORIZON STABILITY & FEASIBILITY

With $N = \infty$, if cost $l(x, u) \geq d \|x\| \forall x, u$ for some $d > 0$.
Then having a finite cost implies stability.

Moreover, since predicted and actual traj. are the same,
feasibility of the first OCP implies feasibility of all
successive OCP's (recursive feasibility)

IDEA:

Add terminal cost and constraints to finite-horizon OCP
to mimic an infinite horizon.

Reminiscent of VALUE FUNCTION

TERMINAL COST & CONSTRAINTS

$$V_N^*(\bar{x}) = \underset{U}{\text{minimize}} \quad \sum_{n=0}^{N-1} l(x_n, u_n) + l_f(x_N) \quad \text{TERMINAL COST}$$

$$\text{subject to} \quad x_{n+1} = f(x_n, u_n) \quad k = 0 \dots N-1$$

$$x_{n+1} \in \mathcal{X}, \quad u_n \in \mathcal{U} \quad k = 0 \dots N-1$$

$$x_0 = \bar{x}$$

$$x_N \in \mathcal{X}_f$$

TERMINAL
CONSTRAINT

How do we choose $l_f(\cdot)$, $\mathcal{X}_f$?

1

FEASIBILITY

FEASIBILITY

Is the OCP going to be feasible at all sampling instants t ?

- INPUT CONSTRAINTS ONLY $\Rightarrow$ Always feasible
- HARD STATE CONSTRAINTS $\Rightarrow$ If $N < \infty$ there is no guarantee that OCP remains feasible, even in the nominal case.

Maximum output admissible set theory:

- $N < \infty$ is enough
- Hard to know which value of N is enough
- Computation time may be too high for such N

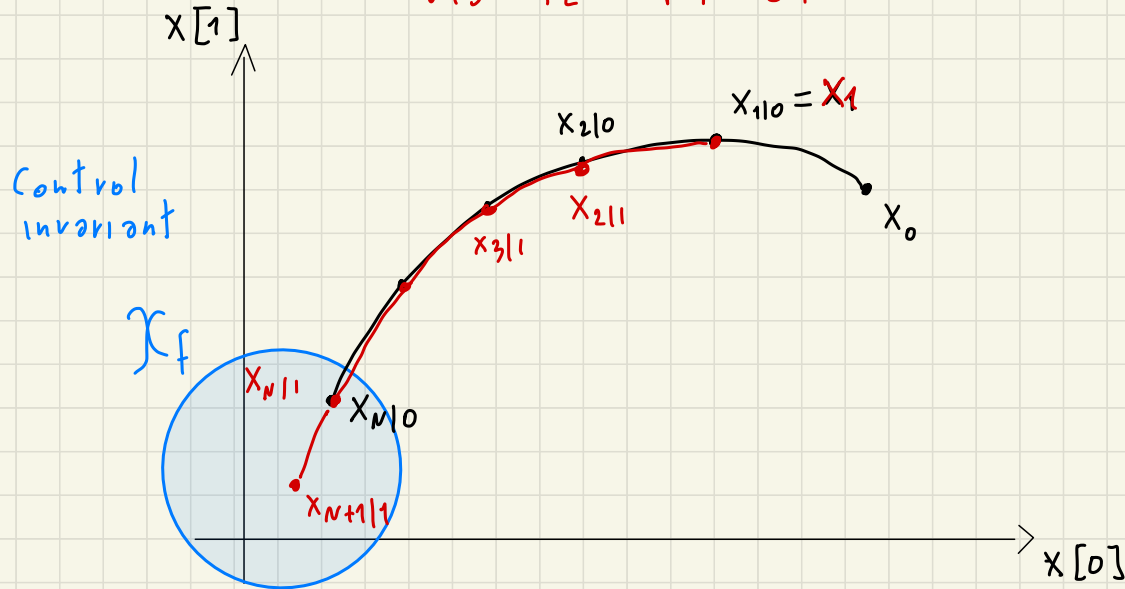
THEOREM

If $\mathcal{X}_f$ is control-invariant $\Rightarrow$ MPC is recursively feasible

DEFINITION

- A set S is control invariant if $\forall x \in S, \exists u \in \mathcal{U}$ s.t. $f(x, u) \in S$
i.e. if you start in S , you can remain in S .

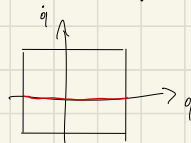
VISUAL PROOF



$$\mathcal{X}_f \subseteq \mathcal{X}$$

CONTROL INVARIANT SETS

Example: for a manipulator, set of zero velocity states is control invariant if $|y(q)| \leq \tau^{\max} \quad \forall q \in [q^{\min}, q^{\max}]$

$$S = \{(q, 0) \mid q^{\min} \leq q \leq q^{\max}, q \in \mathbb{R}^n\}$$


How do I compute C.I. sets in general?

- Hard problem for nonlinear systems
- Set of equilibria S is always C.I., but small:

$$S = \{x \in X \mid \exists u \in \mathcal{U}: x = f(x, u)\}$$

- Can use backward reachable set of S , i.e. states starting from which you can reach S

COMPUTING N-STEP BACKWARD REACHABLE SETS

- We can easily check whether a state $\bar{x} \in \mathcal{B}_N(S)$ (N-step backward reachable set of S) by solving this OCP:

$$\underset{x, u}{\text{minimize}} \quad 1$$

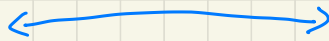
subject to

$$x_{i+1} = f(x_i, u_i) \quad i = 0 \dots N$$

$$x_{i+1} \in \mathcal{X}, \quad u_i \in \mathcal{U} \quad i = 0 \dots N$$

(1)

$$x_N \in S$$



$$x_N = x_{N+1}$$

$$x_0 = \bar{x}$$

- If a solution exists $\Rightarrow \bar{x} \in \mathcal{B}_N(S)$
- Otherwise $\Rightarrow \bar{x} \notin \mathcal{B}_N(S)$

- Use function approximation techniques to learn an approximate representation of $B_n(s)$ from examples
- To build dataset, repeat the following steps:
 - Sample state $\bar{x}$
 - If $\bar{x} \notin \mathcal{X} \Rightarrow$ Add data point $(\bar{x}, 0)$ to dataset
 - Else, try to solve OCP (1)
 - If solution found $\Rightarrow$ Add $(\bar{x}, 1)$ to dataset
 - Else $\Rightarrow$ Add $(\bar{x}, 0)$ to dataset
- When dataset is built, train classifier (e.g. Neural network) to predict label (0 or 1) given state.
- Finally, use classifier as terminal constraint in MPC

2

S T A B I L I T Y

LYAPUNOV FUNCTIONS

Def: $V: \mathbb{R}^n \rightarrow \mathbb{R}$ is exponential Lyapunov function if $\exists d_1, d_2, d_3 > 0$
s.t. $\forall x \in \mathcal{X}$:

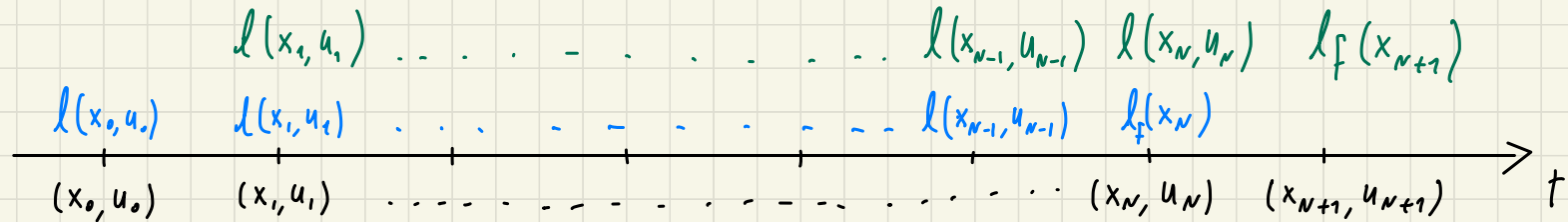
$$d_1 |x| \leq V(x) \leq d_2 |x|$$

$$V(f(x)) - V(x) \leq -d_3 |x|$$

Th: If $\exists$ Lyap. func. $\Rightarrow$ origin $\overset{[x=0]}{\downarrow}$ is exponentially stable in $\mathcal{X}$.

$$\|x_k\| \leq c \gamma^k \|x_0\| \text{ for some } c > 0, \gamma \in [0, 1]$$

VALUE FUNCTION AS LYAPUNOV FUNCTION



$$V_0^*(x_0) = \sum_{i=0}^{N-1} l(x_i, u_i) + l_f(x_N)$$

$$V_1(x_1) = \sum_{i=1}^N l(x_i, u_i) + l_f(x_{N+1})$$

$$V_1(x_1) - V_0^*(x_0) = l(x_N, u_N) + l_f(x_{N+1}) - l_f(x_N) - l(x_0, u_0) \leq -\alpha_3 |x_0| \quad \forall x_N \in \mathcal{X}_f$$

Sufficient but not necessary because $V_1(x_1)$ is not optimal!

STABILITY ASSUMPTIONS

Ass #1: $f(0,0) = 0$, $l(0,0) = 0$, $l_f(0) = 0$

Ass. #2: $\forall x \in X_f, \exists u \in U$ s.t.:

$$l_f(f(x,u)) - l_f(x) \leq -l(x,u)$$

Terminal cost decrease

Ass. #3: $\exists d_1 > 0$ s.t. $\forall x \in X_N, \forall u \in U$

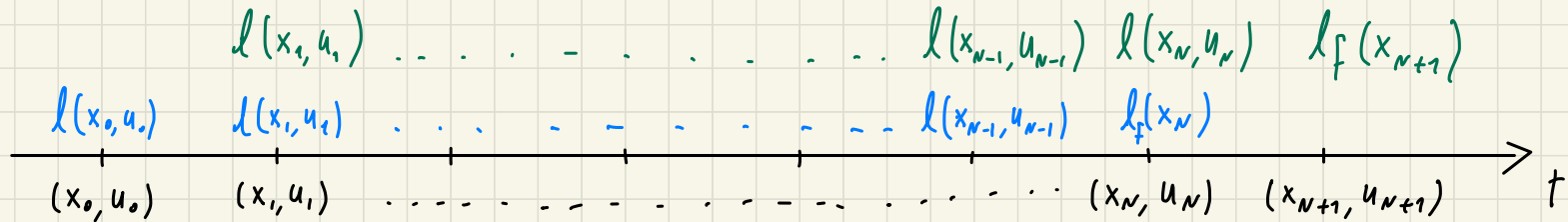
Set of states from which O.C.P. has a solution

$$l(x,u) \geq d_1 |x|$$

Cost lower bound

If #1, #2, #3 hold $\Rightarrow V^*(\cdot)$ is an exp. Lyapunov function

VALUE FUNCTION AS LYAPUNOV FUNCTION



Horizon for $t=0$

$$V_0^*(x_0) = \sum_{i=0}^{N-1} l(x_i, u_i) + l_f(x_N)$$

Horizon for $t=1$

$$V_1(x_1) = \sum_{i=1}^N l(x_i, u_i) + l_f(x_{N+1})$$

$$V_1(x_1) - V_0^*(x_0) = \underbrace{l(x_N, u_N) + l_f(x_{N+1}) - l_f(x_N)}_{\leq 0} - \underbrace{l(x_0, u_0)}_{\leq -\alpha_1 |x_0|} \leq -\alpha_3 |x_0| \quad \forall x_N \in \mathcal{X}_f$$

by Ass. #2 by Ass. #3

STABILITY ASSUMPTIONS

Note that assuming $f(0,0)=0$ is not restrictive, if we want to stabilize $x^* \neq 0$, s.t. $f(x^*, u^*) = x^*$, i.e. x^* must be an equilibrium $\Rightarrow$ Define new state $\bar{x} = x - x^*$, control $\bar{u} = u - u^*$, dynamics:

$$\left. \begin{array}{l} x^+ = f(x, u) \\ x^* = f(x^*, u^*) \end{array} \right\} \begin{array}{l} x^+ - x^* = f(\bar{x} + x^*, \bar{u} + u^*) - x^* \\ \bar{x}^+ = \bar{f}(\bar{x}, \bar{u}) \end{array}$$

So that $\bar{f}(0, 0) = f(0 + x^*, 0 + u^*) - x^* = 0$

STABILITY OPTIONS

① Pick $\mathcal{X}_f$ and l_f (e.g. using sampling-based approach and function approximators) such that:

- $\mathcal{X}_f$ is control invariant

- $\forall x \in \mathcal{X}_f \quad \exists u \in \mathcal{U} \quad \text{s.t.} \quad l_f(f(x,u)) - l_f(x) \leq -l(x,u)$

② Pick $\mathcal{X}_f = \{0\} \Rightarrow$ small basin of attraction

② is special case of ①

③ Pick N "large enough" and set $l_f(x) = 0$, $\mathcal{X}_f = \mathcal{X}$



Most common in practice, especially for nonlinear systems!

Downside: large $N \Rightarrow$ large computation time!

STABILITY EXTENSIONS

- What if system is time varying?
 - e.g., trajectory tracking can be cast as regulation of time-varying system
- Stability + recursive feasibility can still be ensured with time-varying $x_f, l_f(\cdot)$
- What if cost measures some kind of energy consumption
 - $\Rightarrow$ does not satisfy $l(x, u) \geq d|x|$
 - $\Rightarrow$ Economic MPC

③ COMPUTATION TIME

- KEY IDEA: warm start, i.e. use solution of previous OCP as initial guess for current OCP
- Shift trajectory back by 1 time step: $u_n^{\text{guess}} \leftarrow u_{n+1}^*$
- Use zero as initial guess for last time step: $u_{n-1}^{\text{guess}} \leftarrow 0$
- Don't iterate until convergence, just do 1 Newton step
 - Possibly skip line search
- Use hardware accelerators (e.g. GPU)