Adaptation of MPC via RL: fundamental principles

Sébastien Gros

Dept. of Cybernetic, NTNU
Faculty of Information Tech.

Freiburg
Outline

1. Forewords
2. MPC & MDPs
3. A central result on Learning-based MPC
4. RL for Learning-based MPC
Outline

1. Forewords
2. MPC & MDPs
3. A central result on Learning-based MPC
4. RL for Learning-based MPC
Why RL and MPC?

Policy

\( a = \pi_{\theta}(s) \)

from model

\( s_{+} = f(s, a, \theta) \)

and cost (reward)

\( L(s, a) \)
Why RL and MPC?

Policy

\[ a = \pi_\theta(s) \]

from model

\[ s_+ = f(s, a, \theta) \]

and cost (reward)

\[ L(s, a) \]

SYSID

adjust \( \theta \) to fit model

\[ s_+ = f(s, a, \theta) \]

to data
Why RL and MPC?

Policy

\[ a = \pi_\theta(s) \]
from model
\[ s_+ = f(s, a, \theta) \]
and cost (reward)
\[ L(s, a) \]

SYSID
adjust \( \theta \) to fit model
\[ s_+ = f(s, a, \theta) \]
to data

Does this work? Not necessarily...

- Problem: does not capture the real system
- E.g. what \( f \) should be if real system is stochastic?
- Can degrade performance compared to keeping initial \( \theta \)
- Well-known issue is data-based process optimization (RTO)
- Well-known issue in adaptive control
Why RL and MPC?

Policy
\[ a = \pi_\theta(s) \]
from model
\[ s_+ = f(s, a, \theta) \]
and cost (reward)
\[ L(s, a) \]

SYSID
adjust \( \theta \) to fit model
\[ s_+ = f(s, a, \theta) \]
to data

SYSID-like solutions

- Learn real dynamics \( s_{k+1} \sim \mathbb{P} [ \cdot | s_k, a_k ] \) using statistical tools (Gaussian Processes, RKHS, etc)
- Embed these statistical models in MPC
- Increments towards \( \pi_* \) via SYSID+MPC are “exponentially” costly

Affordable MPC?
Why RL and MPC?

Policy

\[ a = \pi_{\theta}(s) \]

from model

\[ s_+ = f(s, a, \theta) \]

and cost (reward)

\[ L(s, a) \]

SYSID

adjust \( \theta \) to fit model

\[ s_+ = f(s, a, \theta) \]

to data

**RL-MPC approach**

- “Milk” the performance of the MPC scheme for a given MPC structure / modelling choice
- Focuses directly on closed-loop performance rather than on “ever better models”
- Not a competing strategy to “better models”, can be used in combination

**In this lecture: basic principles / Next lecture: recent results**
Notation

- Real system dynamics

\[ \mathbb{P}[s_+ | s, a] \in \mathbb{R}_+ \]

denotes the probability (density) of observing a transition from the state-action pair \(s, a\) to the subsequent state \(s_+\).
Notation

- Real system dynamics
  \[ \mathbb{P}[s_+ | s, a] \in \mathbb{R}_+ \]
  denotes the probability (density) of observing a transition from the state-action pair \( s, a \) to the subsequent state \( s_+ \)

- Cost (reward):
  \[ L(s, a) \in \mathbb{R} \]
  assigns a value to each state-action pair. To be minimized here (RL often wants to maximize, no difference)
Notation

- Real system dynamics

\[ P[s_+ | s, a] \in \mathbb{R}_+ \]

denotes the probability (density) of observing a transition from the state-action pair \( s, a \) to the subsequent state \( s_+ \)

- Cost (reward):

\[ L(s, a) \in \mathbb{R} \]

assigns a value to each state-action pair. To be minimized here (RL often wants to maximize, no difference)

- Deterministic policy

\[ a = \pi(s) \]

maps a state \( s \) into an action \( a \)
Notation

- Real system dynamics

\[ P(s_+ | s, a) \in \mathbb{R}_+ \]

denotes the probability (density) of observing a transition from the state-action pair \( s, a \) to the subsequent state \( s_+ \)

- Cost (reward):

\[ L(s, a) \in \mathbb{R} \]

assigns a value to each state-action pair. To be minimized here (RL often wants to maximize, no difference)

- Deterministic policy

\[ a = \pi(s) \]

maps a state \( s \) into an action \( a \)

- Stochastic policy

\[ \pi(a | s) \in \mathbb{R}_+ \]

assigns the probability (density) of taking action \( a \) for a given state \( s \)
Markov Decision Processes (MDP)

A very general way of describing optimal control

- Expected cost (return):
  \[ J(\pi) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, \pi(s_k)) \right] \]

  with discount \( \gamma \in [0, 1] \)

- Fixed or random initial conditions \( s_0 \)

State-action spaces can be continuous or discrete (e.g. integer)
Markov Decision Processes (MDP)

A very general way of describing optimal control

- Expected cost (return):
  \[ J(\pi) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, \pi(s_k)) \right] \]
  with discount \( \gamma \in [0, 1] \)
- Fixed or random initial conditions \( s_0 \)
- MDP: find \( \pi_* \) solution of
  \[ \min_{\pi} J(\pi) \]
  (optimization over policies, i.e. functions)

State-action spaces can be continuous or discrete (e.g. integer)
Markov Decision Processes (MDP)

A very general way of describing optimal control

- Expected cost (return):
  \[
  J(\pi) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, \pi(s_k)) \right]
  \]
  with discount $\gamma \in [0, 1]$

- Fixed or random initial conditions $s_0$

- MDP: find $\pi_\star$ solution of
  \[
  \min_{\pi} J(\pi)
  \]
  (optimization over policies, i.e. functions)

- Optimal parametrized policy $\pi_{\theta_\star}$ given by:
  \[
  \min_{\theta} J(\pi_{\theta})
  \]

State-action spaces can be continuous or discrete (e.g. integer)
Markov Decision Processes (MDP)

A very general way of describing optimal control

- Expected cost (return):
  \[ J(\pi) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, \pi(s_k)) \right] \]
  
  with discount \( \gamma \in [0, 1] \)

- Fixed or random initial conditions \( s_0 \)

- MDP: find \( \pi^* \) solution of
  \[ \min_{\pi} J(\pi) \]
  
  (optimization over policies, i.e. functions)

- Optimal parametrized policy \( \pi_{\theta^*} \) given by:
  \[ \min_{\theta} J(\pi_{\theta}) \]

State-action spaces can be continuous or discrete (e.g. integer)
Markov Decision Processes (MDP)

A very general way of describing optimal control

- Expected cost (return):

\[ J(\pi) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, \pi(s_k)) \right] \]

with discount \( \gamma \in [0, 1] \)
- Fixed or random initial conditions \( s_0 \)
- MDP: find \( \pi_* \) solution of

\[ \min_{\pi} J(\pi) \]

(optimization over policies, i.e. functions)
- Optimal parametrized policy \( \pi_{\theta_*} \) given by:

\[ \min_{\theta} J(\pi_{\theta}) \]

State-action spaces can be continuous or discrete (e.g. integer)
Markov Decision Processes (MDP)

A very general way of describing optimal control

- Expected cost (return):
  \[
  J(\pi) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, \pi(s_k)) \right]
  \]
  
  with discount $\gamma \in [0, 1]$

- Fixed or random initial conditions $s_0$

- MDP: find $\pi_*$ solution of
  \[
  \min_{\pi} J(\pi)
  \]
  (optimization over policies, i.e. functions)

- Optimal parametrized policy $\pi_{\theta_*}$ given by:
  \[
  \min_{\theta} J(\pi_{\theta})
  \]

State-action spaces can be continuous or discrete (e.g. integer)
Markov Decision Processes (MDP)

A very general way of describing optimal control

- Expected cost (return):
  \[ J(\pi) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, \pi(s_k)) \right] \]

  with discount \( \gamma \in [0, 1] \)

- Fixed or random initial conditions \( s_0 \)

- MDP: find \( \pi^* \) solution of
  \[ \min_{\pi} J(\pi) \]

  (optimization over policies, i.e. functions)

- Optimal parametrized policy \( \pi_{\theta^*} \) given by:
  \[ \min_{\theta} J(\pi_{\theta}) \]

State-action spaces can be continuous or discrete (e.g. integer)
Markov Decision Processes (MDP)

A very general way of describing optimal control

- Expected cost (return):
  \[
  J(\pi) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, \pi(s_k)) \right]
  \]
  with discount \( \gamma \in [0, 1] \)

- Fixed or random initial conditions \( s_0 \)

- **MDP**: find \( \pi^* \) solution of
  \[
  \min_{\pi} J(\pi)
  \]
  (optimization over policies, i.e. functions)

- Optimal parametrized policy \( \pi_{\theta^*} \) given by:
  \[
  \min_{\theta} J(\pi_{\theta})
  \]

State-action spaces can be continuous or discrete (e.g. integer)
Markov Decision Processes (MDP)

A very general way of describing optimal control

- Expected cost (return):
  \[ J(\pi) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, \pi(s_k)) \right] \]

  with discount \( \gamma \in [0, 1] \)

- Fixed or random initial conditions \( s_0 \)

- MDP: find \( \pi_* \) solution of
  \[ \min_{\pi} J(\pi) \]

  (optimization over policies, i.e. functions)

- Optimal parametrized policy \( \pi_{\theta_*} \) given by:
  \[ \min_{\theta} J(\pi_{\theta}) \]

State-action spaces can be continuous or discrete (e.g. integer)
Markov Decision Processes (MDP)

A very general way of describing optimal control

- Expected cost (return):
  \[ J(\pi) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, \pi(s_k)) \right] \]

  with discount \( \gamma \in [0, 1] \)

- Fixed or random initial conditions \( s_0 \)

- MDP: find \( \pi_\star \) solution of
  \[ \min_{\pi} J(\pi) \]

  (optimization over policies, i.e. functions)

- Optimal parametrized policy \( \pi_{\theta_\star} \) given by:
  \[ \min_{\theta} J(\pi_{\theta}) \]

State-action spaces can be continuous or discrete (e.g. integer)
Markov Decision Processes (MDP)

A very general way of describing optimal control

- Expected cost (return):
  \[ J(\pi) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, \pi(s_k)) \right] \]
  
  with discount \( \gamma \in [0, 1] \)

- Fixed or random initial conditions \( s_0 \)

- MDP: find \( \pi^*_s \) solution of
  \[ \min_\pi J(\pi) \]

  (optimization over policies, i.e. functions)

- Optimal parametrized policy \( \pi_{\theta^*_s} \) given by:
  \[ \min_{\theta} J(\pi_\theta) \]

State-action spaces can be continuous or discrete (e.g. integer)
Markov Decision Processes (MDP)

A very general way of describing optimal control

- Expected cost (return):
  \[ J(\pi) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, \pi(s_k)) \right] \]
  with discount \( \gamma \in [0, 1] \)

- Fixed or random initial conditions \( s_0 \)

- MDP: find \( \pi_* \) solution of
  \[ \min_{\pi} J(\pi) \]
  (optimization over policies, i.e. functions)

- Optimal parametrized policy \( \pi_{\theta_*} \) given by:
  \[ \min_{\theta} J(\pi_{\theta}) \]

State-action spaces can be continuous of discrete (e.g. integer)
Markov Decision Processes (MDP)

A very general way of describing optimal control

- Expected cost (return):

\[
J(\pi) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, \pi(s_k)) \right]
\]

with discount \( \gamma \in [0, 1] \)
- Fixed or random initial conditions \( s_0 \)
- MDP: find \( \pi_* \) solution of

\[
\min_\pi J(\pi)
\]

(optimization over policies, i.e. functions)
- Optimal parametrized policy \( \pi_{\theta_*} \) given by:

\[
\min_\theta J(\pi_{\theta})
\]

State-action spaces can be continuous or discrete (e.g. integer)
Optimal Value Functions

- **Value function:**

\[ V_\star (s) = \mathbb{E}_{\pi_\star} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right| s_0 = s, a_k = \pi_\star (s_k) \]  

This equation gives the expected cost for policy \( \pi_\star \), starting from given initial conditions \( s \).
Optimal Value Functions

- **Value function:**
  \[
  V_\star (s) = \mathbb{E}_{\pi_\star} \left[ \sum_{k=0}^{\infty} \gamma^k L (s_k, a_k) \middle| s_0 = s, a_k = \pi_\star (s_k) \right]
  \]
  gives the expected cost for policy \( \pi_\star \), starting from given initial conditions \( s \)

- **Action-Value function:**
  \[
  Q_\star (s, a) = \mathbb{E}_{\pi_\star} \left[ \sum_{k=0}^{\infty} \gamma^k L (s_k, a_k) \middle| s_0 = s, a_0 = a, a_{k>0} = \pi_\star (s_k) \right]
  \]
  gives the expected cost for policy \( \pi_\star \), starting from given initial conditions \( s \), and using action \( a \) as first input (policy \( \pi_\star \) after that)
Optimal Value Functions

- **Value function:**
  \[
  V_\star (s) = \mathbb{E}_{\pi_\star} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \middle| s_0 = s, a_k = \pi_\star (s_k) \right]
  \]
  gives the expected cost for policy \( \pi_\star \), starting from given initial conditions \( s \)

- **Action-Value function:**
  \[
  Q_\star (s, a) = \mathbb{E}_{\pi_\star} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \middle| s_0 = s, a_0 = a, a_{k>0} = \pi_\star (s_k) \right]
  \]
  gives the expected cost for policy \( \pi_\star \), starting from given initial conditions \( s \), and using action \( a \) as first input (policy \( \pi_\star \) after that)

- **Relationship:**
  \[
  V_\star (s) = \min_a Q_\star (s, a)
  \]
Optimal Value Functions

- **Value function:**
  \[
  V_\star (s) = \mathbb{E}_{\pi_\star} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \middle| s_0 = s, a_k = \pi_\star (s_k) \right]
  \]
  gives the expected cost for policy $\pi_\star$, starting from given initial conditions $s$

- **Action-Value function:**
  \[
  Q_\star (s, a) = \mathbb{E}_{\pi_\star} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \middle| s_0 = s, a_0 = a, a_{k>0} = \pi_\star (s_k) \right]
  \]
  gives the expected cost for policy $\pi_\star$, starting from given initial conditions $s$, and using action $a$ as first input (policy $\pi_\star$ after that)

- **Relationship:**
  \[
  V_\star (s) = \min_a Q_\star (s, a)
  \]

- **Optimal Policy:**
  \[
  \pi_\star (s) = \arg \min_a Q_\star (s, a)
  \]
Optimal Value Functions

- **Value function:**
  \[ V_\star (s) = \mathbb{E}_{\pi_\star} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \bigg| s_0 = s, a_k = \pi_\star (s_k) \right] \]
  gives the expected cost for policy \( \pi_\star \), starting from given initial conditions \( s \)

- **Action-Value function:**
  \[ Q_\star (s, a) = \mathbb{E}_{\pi_\star} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \bigg| s_0 = s, a_0 = a, a_{k>0} = \pi_\star (s_k) \right] \]
  gives the expected cost for policy \( \pi_\star \), starting from given initial conditions \( s \), and using action \( a \) as first input (policy \( \pi_\star \) after that)

- **Relationship:**
  \[ V_\star (s) = \min_a Q_\star (s, a) \]

- **Optimal Policy:**
  \[ \pi_\star (s) = \arg \min_a Q_\star (s, a) \]

Can be computed via the Bellman equations, intractable for “large” state-action spaces
Value Functions

- **Value function:**
  \[
  V_\pi (s) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right] \bigg| s_0 = s, a_k = \pi (s_k)
  \]
  gives the expected cost for policy \(\pi\), starting from given initial conditions \(s\)

- **Action-Value function:**
  \[
  Q_\pi (s, a) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right] \bigg| s_0 = s, a_0 = a, a_{k>0} = \pi (s_k)
  \]
  gives the expected cost for policy \(\pi\), starting from given initial conditions \(s\), and using action \(a\) as first input (policy \(\pi^*_\) after that)

- **Relationship:**
  \[
  V_\pi (s) = Q_\pi (s, \pi (s_k))
  \]
  Note:
  \[
  V_\pi \neq V_*
  \]
  \[
  Q_\pi \neq Q_*
  \]

- **Advantage function:**
  \[
  A_\pi (s, a) = Q_\pi (s, a) - V_\pi (s)
  \]
  compares \(a\) to policy \(\pi\). Instrumental in policy gradient methods.

Can be computed via the Bellman equations, intractable for “large” state-action spaces.
MDPs and “forbidden” states

What if the system is not allowed to leave a certain subset of the state space?
MDPs and “forbidden” states

What if the system is not allowed to leave a certain subset of the state space?

- Say there is a “feasible” set:

\[ F = \{ s \mid h(s) \leq 0 \} \]

where the state of the system should always be.
MDPs and “forbidden” states

What if the system is not allowed to leave a certain subset of the state space?

Say there is a “feasible” set:

\[ F = \{ s \mid h(s) \leq 0 \} \]

where the state of the system should always be.

In the “MDP theory”, assign an infinite penalty to leaving \( F \), i.e. add:

\[ I_F(s, a) = \begin{cases} 
0 & \text{if } s \in F \\
+\infty & \text{if } s \notin F
\end{cases} \]

to stage cost \( L \).
MDPs and “forbidden” states

What if the system is not allowed to leave a certain subset of the state space?

- Say there is a “feasible” set:
  \[ F = \{ s \mid h(s) \leq 0 \} \]
  where the state of the system should always be.
- In the “MDP theory”, assign an infinite penalty to leaving \( F \), i.e. add:
  \[ I_F(s, a) = \begin{cases} 
  0 & \text{if } s \in F \\
  +\infty & \text{if } s \notin F 
  \end{cases} \]
  to stage cost \( L \).
- In RL, \( \infty \) penalties are not meaningful: “There is no backup from death”
MDPs and “forbidden” states

What if the system is not allowed to leave a certain subset of the state space?

- Say there is a “feasible” set:
  \[
  F = \{ s \mid h(s) \leq 0 \}
  \]
  where the state of the system should always be.
- In the “MDP theory”, assign an infinite penalty to leaving $F$, i.e. add:
  \[
  I_F(s, a) = \begin{cases} 
  0 & \text{if } s \in F \\
  +\infty & \text{if } s \notin F
  \end{cases}
  \]
  to stage cost $L$.
- In RL, $\infty$ penalties are not meaningful: “There is no backup from death”
- Common approach: assign a “very large” penalty to $s \notin F$ instead of $+\infty$. 
MDPs and “forbidden” states

What if the system is not allowed to leave a certain subset of the state space?

Say there is a “feasible” set:

\[ F = \{ s \mid h(s) \leq 0 \} \]

where the state of the system should always be.

In the “MDP theory”, assign an infinite penalty to leaving \( F \), i.e. add:

\[ I_F(s, a) = \begin{cases} 
0 & \text{if } s \in F \\
+\infty & \text{if } s \notin F 
\end{cases} \]

to stage cost \( L \).

In RL, \( \infty \) penalties are not meaningful: “There is no backup from death”

Common approach: assign a “very large” penalty to \( s \notin F \) instead of \( +\infty \).

Use of “barrier functions” in RL
Outline

1. Forewords
2. MPC & MDPs
3. A central result on Learning-based MPC
4. RL for Learning-based MPC
A conceptual comparison...

**MDP:**

\[
\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and system dynamics

\[ s_{k+1} \sim P[\cdot \mid s_k, a_k] \]
MPC & MDPs

A conceptual comparison...

**MDP:**

\[ \min_{\pi} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right] \]

where \( a_k = \pi(s_k) \) and system dynamics

\[ s_{k+1} \sim P[\cdot | s_k, a_k] \]

**MPC:** (\( s_0 \) given)

\[ \min_{s,a} T(s_N) + \sum_{k=0}^{N-1} L(s_k, a_k) \]

s.t. \( s_{k+1} = f(s_k, a_k) \)

yields \( a_0^*, \ldots, a_{N-1}^*(s_0) \) and \( \pi_{\text{MPC}}(s_0) = a_0^* \)
MPC & MDPs

A conceptual comparison...

**MDP:**

$$\min_{\pi} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]$$

where $a_k = \pi(s_k)$ and system dynamics

$$s_{k+1} \sim P[\cdot | s_k, a_k]$$

**MPC:** ($s_0$ given)

$$\min_{s, a} \sum_{k=0}^{\infty} L(s_k, a_k)$$

s.t. $s_{k+1} = f(s_k, a_k)$

yields $a_0^*, \ldots, \infty(s_0)$ and $\pi_{\text{MPC}}(s_0) = a_0^*$

Assume:

- MPC has an infinite horizon
MPC & MDPs

A conceptual comparison...

**MDP:**
\[
\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \( a_k = \pi(s_k) \) and system dynamics
\[
s_{k+1} \sim \delta(s_{k+1} - f(s_k, a_k))
\]

**MPC:** (\(s_0\) given)
\[
\min_{s, a} \sum_{k=0}^{\infty} L(s_k, a_k)
\quad \text{s.t.} \quad s_{k+1} = f(s_k, a_k)
\]
yields \( a_0^*,...,\infty(s_0) \) and \( \pi_{\text{MPC}}(s_0) = a_0^* \)

Assume:
- MPC has an infinite horizon
- MDP has a deterministic dynamics \( f \)
MPC & MDPs

A conceptual comparison...

**MDP:**

\[
\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \(a_k = \pi(s_k)\) and system dynamics

\(s_{k+1} \sim \delta(s_{k+1} - f(s_k, a_k))\)

Assume:
- MPC has an infinite horizon
- MDP has a deterministic dynamics \(f\)
- MPC is discounted

**MPC:** (\(s_0\) given)

\[
\min_{s, a} \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k)
\]

s.t. \(s_{k+1} = f(s_k, a_k)\)

yields \(a^*_0, \ldots, \infty(s_0)\) and \(\pi_{\text{MPC}}(s_0) = a^*_0\)
MPC & MDPs

A conceptual comparison...

**MDP:**
\[
\min_{\pi} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \( a_k = \pi(s_k) \) and system dynamics
\[
s_{k+1} \sim \delta(s_{k+1} - f(s_k, a_k))
\]

**MPC:** (\( s_0 \) given)
\[
\min_{s,a} \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k)
\]
\[
s.t. \quad s_{k+1} = f(s_k, a_k)
\]
yields \( a^*_0, \ldots, \infty(s_0) \) and \( \pi_{\text{MPC}}(s_0) = a^*_0 \)

Assume:
- MPC has an infinite horizon
- MDP has a deterministic dynamics \( f \)
- MPC is discounted

Then (without model error):
\[
\begin{align*}
\pi^*(s_k) &= a^*_k(s_0) = a^*_0(s_k) = \pi_{\text{MPC}}(s_k) \\
\text{MDP solution} & \quad \text{MPC sequence} & \quad \text{MPC 1st control}
\end{align*}
\]
on the trajectories \( s_0, \ldots, \infty \)
MPC & MDPs

A conceptual comparison...

**MDP:**
\[
\min_\pi \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \( a_k = \pi(s_k) \) and system dynamics
\[
s_{k+1} \sim \delta(s_{k+1} - f(s_k, a_k))
\]

**MPC:** (\( s_0 \) given)
\[
\min_{s,a} \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k)
\]
s.t. \( s_{k+1} = f(s_k, a_k) \)
yields \( a^*_0, \ldots, a^*_\infty(s_0) \) and \( \pi_{\text{MPC}}(s_0) = a^*_0 \)

Assume:
- MPC has an infinite horizon
- MDP has a deterministic dynamics \( f \)
- MPC is discounted

Then (without model error):
\[
\pi^*(s_k) = a^*_k(s_0) = a^*_0(s_k) = \pi_{\text{MPC}}(s_k)
\]
on the trajectories \( s_0, \ldots, \infty \)

**Bottom line:** MPC provides optimal policy approximation (finite horizon, deterministic model), i.e. \( \pi_{\text{MPC}} \approx \pi^* \)
MPC & MDPs

A conceptual comparison...

**MDP:**
\[
\min_{\pi} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \( a_k = \pi(s_k) \) and system dynamics
\[
s_{k+1} \sim \mathbb{P} [ \cdot | s_k, a_k ]
\]

**MPC:** (\( s_0 \) given)
\[
\min_{s,a} \gamma^N T(s_N) + \sum_{k=0}^{N-1} \gamma^k L(s_k, a_k)
\]
s.t. \( s_{k+1} = f(s_k, a_k) \)

yields \( a^{\star}_{0,\ldots,N-1}(s_0) \) and \( \pi_{\text{MPC}}(s_0) = a^{\star}_0 \)

**Assume:**
- MPC has an infinite horizon
- MDP has a deterministic dynamics \( f \)
- MPC is discounted

Then (without model error):
\[
\pi^{\star}(s_k) = a^{\star}(s_0) = a^{\star}_0(s_k) = \pi_{\text{MPC}}(s_k)
\]

**Bottom line:** MPC provides optimal policy approximation (finite horizon, deterministic model), i.e. \( \pi_{\text{MPC}} \approx \pi^{\star} \)

**MPC with stochastic model:** better approximation, higher computational cost

---

S. Gros (NTNU) Intro to RL-MPC August 2021 11 / 24
Why discounting?

**MDP:**

\[
\min_{\pi} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and system dynamics

\[
s_{k+1} \sim \mathbb{P} \left[ \cdot \mid s_k, a_k \right]
\]

Discounting is (in general) needed to make the MDP well defined, is that all?
Why discounting?

**MDP:**

\[
\min_\pi \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and system dynamics

\[
s_{k+1} \sim \mathbb{P}[\cdot | s_k, a_k]
\]

Discounting is (in general) needed to make the MDP well defined, is that all?

**System lifetime:** assuming that the system can (irremediably) fail at any time \( k \) with probability \( 1 - \gamma \), then discounting accounts for resulting probabilistic lifetime.
Why discounting?

**MDP:**

\[
\min_\pi \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \(a_k = \pi(s_k)\) and system dynamics

\[
s_{k+1} \sim \mathbb{P}[\cdot \mid s_k, a_k]
\]

Discounting is (in general) needed to make the MDP well defined, is that all?

**System lifetime:** assuming that the system can (irremediably) fail at any time \(k\) with probability \(1 - \gamma\), then discounting accounts for resulting probabilistic lifetime.

*E.g.* a system with a sampling time of 1 second, and a 90% chance of having a lifetime of 20 years, should have \(\gamma = 0.99999999996349275\)
Why discounting?

**MDP:**

\[
\min_\pi \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and system dynamics

\[
s_{k+1} \sim P[\cdot|s_k, a_k]
\]

Discounting is (in general) needed to make the MDP well defined, is that all?

**Investment model:** expected economic growth \( r \) (per time unit) implies that earning at time \( k \) is worth \((1 + r)^{-k}\) the same earning at time 0. Hence \( \gamma = (1 + r)^{-1} \).
Why discounting?

MDP:

$$\min_{\pi} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]$$

where $a_k = \pi(s_k)$ and system dynamics

$$s_{k+1} \sim P[\cdot|s_k, a_k]$$

Discounting is (in general) needed to make the MDP well defined, is that all?

**Investment model**: expected economic growth $r$ (per time unit) implies that earning at time $k$ is worth $(1 + r)^{-k}$ the same earning at time 0. Hence $\gamma = (1 + r)^{-1}$.

E.g. a system with a sampling time of 1 second and an expected return of 10% per year should have $\gamma = 0.99999999848887$
**Why discounting?**

MDP:

$$\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]$$

where $a_k = \pi(s_k)$ and system dynamics

$$s_{k+1} \sim \mathbb{P} [ \cdot | s_k, a_k ]$$

Discounting is (in general) needed to make the MDP well defined, is that all?

**Investment model:** expected economic growth $r$ (per time unit) implies that earning at time $k$ is worth $(1 + r)^{-k}$ the same earning at time 0. Hence $\gamma = (1 + r)^{-1}$.

*E.g. a system with a sampling time of 1 second and an expected return of 10% per year should have $\gamma = 0.999999999848887$*

**Bottom line:** on “engineering applications”, the discount tends to (should) be extremely close to 1
Why discounting?

Gain optimal MDP:

$$\min_{\pi} \lim_{N \to \infty} \mathbb{E}_\pi \left[ \sum_{k=0}^{N} \frac{1}{N} L(s_k, a_k) \right]$$

where $a_k = \pi(s_k)$ and system dynamics

$$s_{k+1} \sim \mathbb{P} [ \cdot | s_k, a_k ]$$

What about considering average cost?

Policy $\pi$

- is said to achieve "gain optimality"
- transients are irrelevant as they have no contribution in the average return
- tend to yield "bang-bang" actions until optimal steady state is reached
- is not unique!
Why discounting?

Gain optimal MDP:

$$\min_{\pi} \lim_{N \to \infty} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{N} \frac{1}{N} L(s_k, a_k) \right]$$

where $$a_k = \pi(s_k)$$ and system dynamics

$$s_{k+1} \sim P[\cdot | s_k, a_k]$$

What about considering average cost?

Policy $$\pi$$

- is said to achieve “gain optimality”
- transients are irrelevant as they have no contribution in the average return
- tend to yield “bang-bang” actions until optimal steady state is reached
- is not unique!

... gain optimal are of questionable use for control
Why discounting?

Bias optimal MDP:

$$\min_\pi \mathbb{E}_\pi \left[ \sum_{k=0}^{N} L(s_k, a_k) - V^*_G(s_0) \right]$$

where $a_k = \pi(s_k)$ and system dynamics

$$s_{k+1} \sim \mathbb{P}[\cdot | s_k, a_k]$$

where $V^*_G$ is the value function associated to gain optimal problem.

Policy $\pi$

- is said to achieve “bias optimality”
- “best transient to gain-optimal state”
- there are RL algorithms for bias optimality
Why discounting?

**Bias optimal MDP:**

\[
\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^N L(s_k, a_k) - V_G^*(s_0) \right]
\]

where \( a_k = \pi(s_k) \) and system dynamics

\[
s_{k+1} \sim \mathbb{P} [ \cdot | s_k, a_k ]
\]

where \( V_G^* \) is the value function associated to gain optimal problem.

**Policy \( \pi \)**

- is said to achieve “bias optimality”
- “best transient to gain-optimal state”
- there are RL algorithms for bias optimality

Outline

1. Forewords
2. MPC & MDPs
3. A central result on Learning-based MPC
4. RL for Learning-based MPC
MPC-based value functions

**MDP:**

\[
\min_{\pi} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and

\[
s_{k+1} \sim \mathbb{P} [ \cdot \mid s_k, a_k ]
\]

**MPC:** (\( s_0 \) given)

\[
\min_{s,a} \gamma^N T(s_N) + \sum_{k=0}^{N-1} \gamma^k L(s_k, a_k)
\]

s.t. \( s_{k+1} = f(s_k, a_k) \)

yields \( a_0^*, \ldots, a_{N-1}^* (s_0) \) and \( \pi_{\text{MPC}}(s_0) = a_0^* \)
MPC-based value functions

**MDP:**
\[
\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \( a_k = \pi(s_k) \) and
\[
s_{k+1} \sim \mathbb{P}[\cdot \mid s_k, a_k]
\]

**MPC: \((s_0 \text{ given})\)**
\[
\min_{s, a \gamma^N T(s_N) + \sum_{k=0}^{N-1} \gamma^k L(s_k, a_k)} \text{ s.t. } s_{k+1} = f(s_k, a_k)
\]
yields \( a_0^*, \ldots, N-1 (s_0) \) and \( \pi_{\text{MPC}}(s_0) = a_0^* \)

**Value Functions:**
\[
V_*(s) = \mathbb{E}_{\pi_*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
\[
Q_*(s, a) = \mathbb{E}_{\pi_*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \mid a_0 = a \right]
\]
MPC-based value functions

**MDP:**

\[
\min_\pi \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and

\[ s_{k+1} \sim P[ \cdot | s_k, a_k ] \]

**MPC:** (\( s_0 \) given)

\[
V_{\text{MPC}}(s_0) = \min_{s,a} \gamma^N T(s_N) + \sum_{k=0}^{N-1} \gamma^k L(s_k, a_k)
\]

\[ \text{s.t. } s_{k+1} = f(s_k, a_k) \]

i.e. MPC scheme provides a value function

- MPC delivers a value function \( V_{\text{MPC}} \)

**Value Functions:**

\[
V_*(s) = \mathbb{E}_{\pi_*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

\[
Q_*(s,a) = \mathbb{E}_{\pi_*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \mid a_0 = a \right]
\]
MPC-based value functions

**MDP:**

\[
\min_\pi \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and

\[
s_{k+1} \sim P[\cdot | s_k, a_k]
\]

**Value Functions:**

\[
V_\pi(s) = \mathbb{E}_{\pi_\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

\[
Q_\pi(s, a) = \mathbb{E}_{\pi_\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \middle| a_0 = a \right]
\]

**MPC:** (\( s_0 \) given)

\[
Q_{MPC}(s_0, a) = \min_{s, a} \gamma^N T(s_N) + \sum_{k=0}^{N-1} \gamma^k L(s_k, a_k)
\]

s.t. \( s_{k+1} = f(s_k, a_k) \)

\( a_0 = a \)

i.e. MPC scheme provides an action-value function

- MPC delivers a value function \( V_{MPC} \)
- MPC (can) deliver an action-value function \( Q_{MPC} \)
MPC-based value functions

**MDP:**
\[
\min_{\pi} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \( a_k = \pi(s_k) \) and
\[
s_{k+1} \sim \mathbb{P}[ \cdot | s_k, a_k ]
\]

**Value Functions:**
\[
V^*(s) = \mathbb{E}_{\pi^*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
\[
Q^*(s, a) = \mathbb{E}_{\pi^*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right] | a_0 = a
\]

**MPC:** (\( s_0 \) given)
\[
Q_{\text{MPC}}(s_0, a) = \min_{s, a} \gamma^N T(s_N) + \sum_{k=0}^{N-1} \gamma^k L(s_k, a_k)
\]
\[
s.t. \quad s_{k+1} = f(s_k, a_k) \quad a_0 = a
\]
i.e. MPC scheme provides an action-value function

- MPC delivers a value function \( V_{\text{MPC}} \)
- MPC (can) deliver an action-value function \( Q_{\text{MPC}} \)
- MPC delivers a policy \( \pi_{\text{MPC}} \)
MPC-based value functions

**MDP:**

\[
\min_\pi \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and

\( s_{k+1} \sim \mathbb{P}[\cdot|s_k, a_k] \)

**Value Functions:**

\[
V^* (s) = \mathbb{E}_{\pi^*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

\[
Q^* (s, a) = \mathbb{E}_{\pi^*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right| a_0 = a
\]

**MPC:** \((s_0 \text{ given})\)

\[
Q_{\text{MPC}} (s_0, a) = \min_{s, a} \gamma^N T(s_N) + \sum_{k=0}^{N-1} \gamma^k L(s_k, a_k)
\]

s.t. \( s_{k+1} = f(s_k, a_k) \)

\( a_0 = a \)

i.e. MPC scheme provides an action-value function

- MPC delivers a value function \( V_{\text{MPC}} \)
- MPC (can) deliver an action-value function \( Q_{\text{MPC}} \)
- MPC delivers a policy \( \pi_{\text{MPC}} \)
- Fundamental relationships satisfied:

\[
V_{\text{MPC}} (s) = \min_a Q_{\text{MPC}} (s, a)
\]

\[
\pi_{\text{MPC}} (s) = \arg \min_a Q_{\text{MPC}} (s, a)
\]
**MPC-based value functions**

**MDP:**

\[
\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and

\[ s_{k+1} \sim P[\cdot|s_k, a_k] \]

**Value Functions:**

\[
V_\ast(s) = \mathbb{E}_{\pi_\ast} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

\[
Q_\ast(s, a) = \mathbb{E}_{\pi_\ast} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \mid a_0 = a \right]
\]

Similarly to \( \pi_{MPC} \approx \pi_\ast \):

\[
V_{MPC}(s) \approx V_\ast(s)
\]

\[
Q_{MPC}(s, a) \approx Q_\ast(s, a)
\]

**MPC:** (\( s_0 \) given)

\[
Q_{MPC}(s_0, a) = \min_{s, a} \gamma^N T(s_N) + \sum_{k=0}^{N-1} \gamma^k L(s_k, a_k)
\]

\[
s.t. \quad s_{k+1} = f(s_k, a_k)
\]

\[ a_0 = a \]

i.e. MPC scheme provides an action-value function

- MPC delivers a value function \( V_{MPC} \)
- MPC (can) deliver an action-value function \( Q_{MPC} \)
- MPC delivers a policy \( \pi_{MPC} \)
- Fundamental relationships satisfied:

\[
V_{MPC}(s) = \min_a Q_{MPC}(s, a)
\]

\[
\pi_{MPC}(s) = \arg\min_a Q_{MPC}(s, a)
\]
A central result...

**MDP:**
\[
\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \(a_k = \pi(s_k)\) and system dynamics
\[
s_{k+1} \sim \mathbb{P}[\cdot | s_k, a_k]
\]

Value and Action-Value Functions:
\[
V^*(s) = \mathbb{E}_{\pi^*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
\[
Q^*(s, a) = \mathbb{E}_{\pi^*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \mid a_0 = a \right]
\]

**MPC:** (\(s_0\) given)
\[
\min_{s, a} \gamma^N T(s_N) + \sum_{k=0}^{N-1} \gamma^k L(s_k, a_k)
\]
s.t. \(s_{k+1} = f(s_k, a_k)\)
yields \(\pi_{\text{MPC}}, V_{\text{MPC}}, \text{and } Q_{\text{MPC}}\)
A central result...

**MDP:**
\[
\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \( a_k = \pi(s_k) \) and system dynamics
\[
s_{k+1} \sim \mathbb{P} [ \cdot | s_k, a_k ]
\]

Value and Action-Value Functions:
\[
V_*(s) = \mathbb{E}_{\pi_*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
\[
Q_*(s, a) = \mathbb{E}_{\pi_*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \mid a_0 = a \right]
\]

**MPC:** (\( s_0 \) given)
\[
\min_{s,a} \gamma^N T(s_N) + \sum_{k=0}^{N-1} \gamma^k L(s_k, a_k)
\]
\[
s.t. \quad s_{k+1} = f(s_k, a_k)
\]
yields \( \pi_{\text{MPC}}, V_{\text{MPC}}, \) and \( Q_{\text{MPC}} \)

In general
\[
\pi_{\text{MPC}} \neq \pi_*, \quad V_{\text{MPC}} \neq V_*, \quad Q_{\text{MPC}} \neq Q_*
\]
but...
A central result...

**MDP:**

\[
\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and system dynamics

\[
s_{k+1} \sim \mathbb{P}\left[ \cdot \mid s_k, a_k \right]
\]

**Value and Action-Value Functions:**

\[
V_* (s) = \mathbb{E}_{\pi_*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

\[
Q_* (s, a) = \mathbb{E}_{\pi_*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \mid a_0 = a \right]
\]

**MPC:** (\( s_0 \) given)

\[
\min_{s,a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k)
\]

s.t. \( s_{k+1} = f(s_k, a_k) \)

yields \( \pi_{\text{MPC}}, V_{\text{MPC}}, \) and \( Q_{\text{MPC}} \)

In general

\[
\pi_{\text{MPC}} \neq \pi_*, \ V_{\text{MPC}} \neq V_*, \ Q_{\text{MPC}} \neq Q_*
\]

but...
A central result...

**MDP:**
\[
\min_\pi \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and system dynamics
\[ s_{k+1} \sim \mathbb{P} [ \cdot \mid s_k, a_k ] \]

**Value and Action-Value Functions:**
\[
V^* (s) = \mathbb{E}_{\pi^*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
\[
Q^* (s, a) = \mathbb{E}_{\pi^*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \mid a_0 = a \right]
\]

**MPC:**
\[
\min_{s,a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k)
\]

s.t. \[ s_{k+1} = f(s_k, a_k) \]

yields \( \pi_{\text{MPC}}, V_{\text{MPC}}, \) and \( Q_{\text{MPC}} \)

Under some assumptions, there are \( \tilde{L}, \tilde{T} \) s.t.
\[
\pi_{\text{MPC}} = \pi^*, \ V_{\text{MPC}} = V^*, \ Q_{\text{MPC}} = Q^*
\]

**Assumption:** trajectories of model \( f \) under optimal policy \( \pi^* \) should yield bounded
\[ \gamma^k L(s_k, a_k) \] for \( k = 0, \ldots, \infty \)
A central result...

**MDP:**

\[
\min_{\pi} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and system dynamics

\[
s_{k+1} \sim \mathbb{P} [ \cdot | s_k, a_k ]
\]

**MPC:** (\( s_0 \) given)

\[
\min_{s, a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k)
\]

s.t. \( s_{k+1} = f(s_k, a_k) \)

yields \( \pi_{\text{MPC}}, V_{\text{MPC}}, \text{and } Q_{\text{MPC}} \)

**Value and Action-Value Functions:**

\[
V_*(s) = \mathbb{E}_{\pi_*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

\[
Q_*(s, a) = \mathbb{E}_{\pi_*} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) | a_0 = a \right]
\]

Under some assumptions, there are \( \tilde{L}, \tilde{T} \) s.t.

\[
\pi_{\text{MPC}} = \pi_*, V_{\text{MPC}} = V_*, Q_{\text{MPC}} = Q_*
\]

**Assumption:** trajectories of model \( f \) under optimal policy \( \pi_* \) should yield bounded \( \gamma^k L(s_k, a_k) \) for \( k = 0, \ldots, \infty \)

- MPC can “capture” \( \pi_*, Q_*, V_* \), **even if MPC model is inaccurate**
- Requires modifications of the stage cost & constraints
- Valid for all MPC schemes (classic, robust, stochastic, economic, etc)
A central result...

MDP:
\[
\min_{\pi} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \( a_k = \pi(s_k) \) and system dynamics
\[
s_{k+1} \sim \mathbb{P} \left[ \cdot \mid s_k, a_k \right]
\]

Value and Action-Value Functions:
\[
V_\star(s) = \mathbb{E}_{\pi_\star} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
\[
Q_\star(s, a) = \mathbb{E}_{\pi_\star} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right] \quad \text{a}_0 = a
\]

MPC: (\( s_0 \) given)
\[
\min_{s, a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k)
\]
\[
s.t. \quad s_{k+1} = f(s_k, a_k)
\]
yields \( \pi_{\text{MPC}}, V_{\text{MPC}}, \text{and } Q_{\text{MPC}} \)

Under some assumptions, there are \( \tilde{L}, \tilde{T} \) s.t.
\[
\pi_{\text{MPC}} = \pi_\star, \quad V_{\text{MPC}} = V_\star, \quad Q_{\text{MPC}} = Q_\star
\]

Assumption: trajectories of model \( f \) under optimal policy \( \pi_\star \) should yield bounded
\[
\gamma^k L(s_k, a_k) \text{ for } k = 0, \ldots, \infty
\]

- MPC can “capture” \( \pi_\star, Q_\star, V_\star \), even if MPC model is inaccurate
- Requires modifications of the stage cost & constraints
- Valid for all MPC schemes (classic, robust, stochastic, economic, etc)

Data-driven Economic NMPC using Reinforcement Learning, S. Gros, M. Zanon, Transaction on Automatic Control, 2019
A central result...

**MDP:**
\[ \min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right] \]

where \( a_k = \pi(s_k) \) and system dynamics
\( s_{k+1} \sim \mathbb{P}[\cdot | s_k, a_k] \)

**Value and Action-Value Functions:**
\[ V_\star(s) = \mathbb{E}_{\pi_\star} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right] \]
\[ Q_\star(s, a) = \mathbb{E}_{\pi_\star} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right] | a_0 = a \]

**MPC:** (\( s_0 \) given)
\[ \min_{s, a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k) \]

s.t. \( s_{k+1} = f(s_k, a_k) \)

yields \( \pi_{MPC}, V_{MPC}, \) and \( Q_{MPC} \)

Under some assumptions, there are \( \tilde{L}, \tilde{T} \) s.t.
\[ \pi_{MPC} = \pi_\star, \quad V_{MPC} = V_\star, \quad Q_{MPC} = Q_\star \]

**Assumption:** trajectories of model \( f \) under optimal policy \( \pi_\star \) should yield bounded
\[ \gamma^k L(s_k, a_k) \] for \( k = 0, \ldots, \infty \)

If you do (any) Learning-MPC and adjust the cost and/or constraints, then this paper is formally justifying what you are doing.

Data-driven Economic NMPC using Reinforcement Learning, S. Gros, M. Zanon, Transaction on Automatic Control, 2019
MDP:  
\[
\min_\pi \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \( a_k = \pi(s_k) \) and system dynamics
\[
s_{k+1} \sim P[\cdot | s_k, a_k]
\]

MPC: (s_0 given)  
\[
\min_{s,a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k)
\]
\[
s.t. \quad s_{k+1} = f(s_k, a_k)
\]

yields \( \pi_{\text{MPC}}, V_{\text{MPC}}, \text{and } Q_{\text{MPC}} \)
Practical consequences...

**MDP:**

\[
\min_{\pi} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and system dynamics

\[
s_{k+1} \sim \mathbb{P} \left[ \cdot \mid s_k, a_k \right]
\]

**MPC:** (\( s_0 \) given)

\[
\min_{s, a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k)
\]

\[
\text{s.t.} \quad s_{k+1} = f(s_k, a_k)
\]

yields \( \pi_{\text{MPC}}, V_{\text{MPC}}, \) and \( Q_{\text{MPC}} \)

In principle, it is possible to “modify” the MPC scheme such that it produces

\[
\pi_{\text{MPC}} = \pi_*, \quad V_{\text{MPC}} = V_*, \quad Q_{\text{MPC}} = Q_*
\]
Practical consequences...

**MDP:**
\[
\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \( a_k = \pi(s_k) \) and system dynamics
\[s_{k+1} \sim \mathbb{P} \left[ \cdot \mid s_k, a_k \right]\]

**MPC:** (\( s_0 \) given)
\[
\min_{s, a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k)
\]
s.t. \( s_{k+1} = f(s_k, a_k) \)
yields \( \pi_{\text{MPC}}, V_{\text{MPC}}, \text{and } Q_{\text{MPC}} \)

- In principle, it is possible to “modify” the MPC scheme such that it produces
  \[
  \pi_{\text{MPC}} = \pi_*, \ V_{\text{MPC}} = V_*, \ Q_{\text{MPC}} = Q_*
  \]
- Unfortunately, computing \( \tilde{L}, \tilde{T} \) is as difficult as solving the Bellman equations
Practical consequences...

**MDP:**

$$\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]$$

where $$a_k = \pi(s_k)$$ and system dynamics

$$s_{k+1} \sim \mathbb{P}[\cdot | s_k, a_k]$$

**MPC:** ($$s_0$$ given)

$$\min_{s, a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k)$$

s.t. $$s_{k+1} = f(s_k, a_k)$$

yields $$\pi_{MPC}, V_{MPC},$$ and $$Q_{MPC}$$

- In principle, it is possible to “modify” the MPC scheme such that it produces

  $$\pi_{MPC} = \pi_*, V_{MPC} = V_*, Q_{MPC} = Q_*$$

- Unfortunately, computing $$\tilde{L}, \tilde{T}$$ is as difficult as solving the Bellman equations

- Not very useful in practice, **unless** we are working in a “learning” context...
Practical consequences...

**MDP:**

$$\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]$$

where $a_k = \pi(s_k)$ and system dynamics

$$s_{k+1} \sim P([\cdot|s_k, a_k])$$

**MPC:** (given $s_0$)

$$\min_{s, a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k)$$

s.t. $s_{k+1} = f(s_k, a_k)$

yields $\pi_{MPC}$, $V_{MPC}$, and $Q_{MPC}$

- In principle, it is possible to “modify” the MPC scheme such that it produces
  $$\pi_{MPC} = \pi_\star, \ V_{MPC} = V_\star, \ Q_{MPC} = Q_\star$$

- Unfortunately, computing $\tilde{L}$, $\tilde{T}$ is as difficult as solving the Bellman equations

- Not very useful in practice, unless we are working in a “learning” context...

- Then $\tilde{L}$, $\tilde{T}$ is something that we learn from the closed-loop trajectories
Practical consequences...

**MDP:**
\[
\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \( a_k = \pi(s_k) \) and system dynamics
\[
s_{k+1} \sim \mathbb{P}[\cdot | s_k, a_k]
\]

**MPC:** (\( s_0 \) given)
\[
\min_{s, a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k)
\]
\[
s.t. \quad s_{k+1} = f(s_k, a_k)
\]

yields \( \pi_{MPC}, V_{MPC}, \) and \( Q_{MPC} \)

- In principle, it is possible to “modify” the MPC scheme such that it produces
  \[
  \pi_{MPC} = \pi_*, \quad V_{MPC} = V_*, \quad Q_{MPC} = Q_*
  \]
- Unfortunately, computing \( \tilde{L}, \tilde{T} \) is as difficult as solving the Bellman equations
- Not very useful in practice, unless we are working in a “learning” context...
- Then \( \tilde{L}, \tilde{T} \) is something that we learn from the closed-loop trajectories
- E.g. RL can be used to learn \( \tilde{L}, \tilde{T} \) (+possibly MPC model)
Outline

1. Forewords
2. MPC & MDPs
3. A central result on Learning-based MPC
4. RL for Learning-based MPC
Classic RL vs. RL-MPC

**MDP:**
\[
\min_{\pi} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \(a_k = \pi(s_k)\) and system dynamics
\[
s_{k+1} \sim P[\cdot|s_k, a_k]
\]

**MPC:**
\[
\min_{s,a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k)
\]
s.t. \(s_{k+1} = f(s_k, a_k)\)
yields \(\pi_{\text{MPC}}, V_{\text{MPC}},\) and \(Q_{\text{MPC}}\)
Classic RL vs. RL-MPC

**MDP:**

\[
\min_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]

where \( a_k = \pi(s_k) \) and system dynamics

\[
s_{k+1} \sim \mathbb{P}[\cdot | s_k, a_k]
\]

**MPC:**

\[
\min_{s, a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k)
\]

s.t. \( s_{k+1} = f(s_k, a_k) \)

yields \( \pi_{\text{MPC}}, V_{\text{MPC}}, \) and \( Q_{\text{MPC}} \)

**RL with DNN**

- correct structure is unknown
- good initialization is difficult
- respecting constraints is difficult & implicit
**Classic RL vs. RL-MPC**

**MDP:**
\[
\min_{\pi} \quad \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
where \( a_k = \pi(s_k) \) and system dynamics
\[
s_{k+1} \sim P[\cdot|s_k, a_k]
\]

**MPC:**
\[
\min_{s,a} \quad \gamma^N \tilde{L}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k)
\]
s.t. \( s_{k+1} = f(s_k, a_k) \)
yields \( \pi_{MPC}, V_{MPC}, \text{and } Q_{MPC} \)

**RL with DNN**
- correct structure is unknown
- good initialization is difficult
- respecting constraints is difficult & implicit

---

S. Gros (NTNU)
Intro to RL-MPC
August 2021 18 / 24
Classic RL vs. RL-MPC

**MDP:**
\[ \min_{\pi} \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right] \]
where \( a_k = \pi(s_k) \) and system dynamics
\[ s_{k+1} \sim \mathbb{P}[\cdot|s_k, a_k] \]

**MPC:**
\[ \min_{s,a} \gamma^N \tilde{T}(s_N) + \sum_{k=0}^{N-1} \gamma^k \tilde{L}(s_k, a_k) \]
s.t. \( s_{k+1} = f(s_k, a_k) \)

yields \( \pi_{\text{MPC}}, V_{\text{MPC}}, \text{ and } Q_{\text{MPC}} \)

**RL with DNN**
- correct structure is unknown
- good initialization is difficult
- respecting constraints is difficult & implicit

**MPC**
- Provides \( V_{\text{MPC}} \equiv \hat{V}_*, Q_{\text{MPC}} \equiv \hat{Q}_*, \pi_{\text{MPC}} \equiv \hat{\pi}_* \)
- Structure and initialization given
- Constraints enforced explicitly
- Theory says that we can get \( V_*, Q_*, \pi_* \) from MPC
**Parametrized MPC:**

\[
\min_{s,a} \quad \gamma^N T_\theta(s_N) + \sum_{k=0}^{N-1} \gamma^k L_\theta(s_k, a_k)
\]

s.t.

\[ s_{k+1} = f_\theta(s_k, a_k) \]
\[ h_\theta(s_k, a_k) \leq 0 \]

yields \(\pi_\theta\), \(V_\theta\), and \(Q_\theta\)

**RL:** does

\[
\min_{\theta} \quad J(\pi_\theta)
\]

on the real system, where

\[
J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[ \sum_{k=0}^{\infty} \gamma^k L(s_k, a_k) \right]
\]
**RL and MPC**

**Parametrized MPC:**

\[
\min_{s,a} \gamma^N T_\theta (s_N) + \sum_{k=0}^{N-1} \gamma^k L_\theta (s_k, a_k)
\]

s.t. \( s_{k+1} = f_\theta (s_k, a_k) \)

\( h_\theta (s_k, a_k) \leq 0 \)

yields \( \pi_\theta, V_\theta, \) and \( Q_\theta \)

- Parametrize all functions
- Constraints \( h_\theta \) for forbidden state-actions

**RL:** does

\[
\min_{\theta} J (\pi_\theta)
\]

on the real system, where

\[
J (\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[ \sum_{k=0}^{\infty} \gamma^k L (s_k, a_k) \right]
\]
RL and MPC

**Parametrized MPC:**
\[
\min_{s,a} \gamma^N T_\theta (s_N) + \sum_{k=0}^{N-1} \gamma^k L_\theta (s_k, a_k)
\]
\[
\text{s.t. } s_{k+1} = f_\theta (s_k, a_k)
\]
\[
h_\theta (s_k, a_k) \leq 0
\]

yields \( \pi_\theta, V_\theta, \) and \( Q_\theta \)

- Parametrize all functions
- Constraints \( h_\theta \) for forbidden state-actions

**RL:** does
\[
\min_\theta J(\pi_\theta)
\]
on the real system, where
\[
J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[ \sum_{k=0}^{\infty} \gamma^k L (s_k, a_k) \right]
\]

All RL techniques can be applied to an MPC scheme. RL adjusts the MPC parameters to minimize the closed-loop cost \( J(\pi_\theta) \)
RL and MPC

**Parametrized MPC:**

\[
\min_{s,a} \gamma^N T_\theta(s_N) + \sum_{k=0}^{N-1} \gamma^k L_\theta(s_k, a_k)
\]

subject to:

\[
s_{k+1} = f_\theta(s_k, a_k)
\]

\[
h_\theta(s_k, a_k) \leq 0
\]

yields \(\pi_\theta, V_\theta,\) and \(Q_\theta\)

- Parametrize all functions
- Constraints \(h_\theta\) for forbidden state-actions

**RL:** does

\[
\min_{\theta} J(\pi_\theta)
\]

on the real system, where

\[
J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{k=0}^{\infty} \gamma^k L(s_k, a_k)\right]
\]

**Good starting point:** (MPC as usual)

- \(L_{\theta_0} = L, h_{\theta_0}\) selected according to the desired constraints
- \(f_{\theta_0}\) selected from SYSID

but departing from that can help!!

All RL techniques can be applied to an MPC scheme. RL adjusts the MPC parameters to minimize the closed-loop cost \(J(\pi_\theta)\)
RL and MPC

Parametrized MPC:

$$\min_{s,a} \gamma^N T_\theta (s_N) + \sum_{k=0}^{N-1} \gamma^k L_\theta (s_k, a_k)$$

s.t.  $$s_{k+1} = f_\theta (s_k, a_k)$$
      $$h_\theta (s_k, a_k) \leq 0$$

yields $$\pi_\theta$$, $$V_\theta$$, and $$Q_\theta$$

- Parametrize all functions
- Constraints $$h_\theta$$ for forbidden state-actions

All RL techniques can be applied to an MPC scheme. RL adjusts the MPC parameters to minimize the closed-loop cost $$J(\pi_\theta)$$

RL: does

$$\min_{\theta} J(\pi_\theta)$$
on the real system, where

$$J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[ \sum_{k=0}^{\infty} \gamma^k L (s_k, a_k) \right]$$

Good starting point: (MPC as usual)

- $$L_\theta_0 = L$$, $$h_\theta_0$$ selected according to the desired constraints
- $$f_\theta_0$$ selected from SYSID

but departing from that can help!!

Note: MPC model tuning via RL ≠ SYSID
Form function approximators:

\[ Q_\theta(s, a), \ V_\theta(s), \ \pi_\theta(s) \]

via ad-hoc parametrization
RL methods - Reminder

- **Q-learning methods** adjust $\theta$ to get

  $Q_\theta (s, a) \approx Q_\star (s, a)$

Yields policy:

$\pi_\theta (s) = \min_a Q_\theta (s, a) \approx \min_a Q_\star (s, a) = \pi_\star (s)$

E.g. basic Q-learning uses:

$$\theta \leftarrow \theta + \alpha \delta \nabla_\theta Q_\theta (s_k, a_k)$$

$$\delta = L (s_k, a_k) + \gamma V_\theta (s_{k+1}) - Q_\theta (s_k, a_k)$$
RL methods - Reminder

Form function approximators:

\[ Q_\theta(s, a), \ V_\theta(s), \ \pi_\theta(s) \]
via ad-hoc parametrization

- **Q-learning methods** adjust \( \theta \) to get

\[ Q_\theta(s, a) \approx Q_*(s, a) \]

Yields policy:

\[ \pi_\theta(s) = \min_a Q_\theta(s, a) \approx \min_a Q_*(s, a) = \pi_*(s) \]

E.g. basic Q-learning uses:

\[
\begin{align*}
\theta &\leftarrow \theta + \alpha \delta \nabla_\theta Q_\theta(s_k, a_k) \\
\delta &= L(s_k, a_k) + \gamma V_\theta(s_{k+1}) - Q_\theta(s_k, a_k)
\end{align*}
\]

- **Policy gradient methods** adjust \( \theta \) to get

\[ \nabla_\theta J(\pi_\theta) = 0 \]

yields policy \( \pi_\theta(x) \approx \pi_*(x) \) directly.
**RL methods - Reminder**

- **Q-learning methods** adjust $\theta$ to get
  \[
  Q_\theta (s, a) \approx Q_\star (s, a)
  \]
  Yields policy:
  \[
  \pi_\theta (s) = \min_a Q_\theta (s, a) \approx \min_a Q_\star (s, a) = \pi_\star (s)
  \]
  E.g. basic Q-learning uses:
  \[
  \theta \leftarrow \theta + \alpha \delta \nabla_\theta Q_\theta (s_k, a_k)
  \]
  \[
  \delta = L (s_k, a_k) + \gamma V_\theta (s_{k+1}) - Q_\theta (s_k, a_k)
  \]

- **Policy gradient methods** adjust $\theta$ to get
  \[
  \nabla_\theta J (\pi_\theta) = 0
  \]
  yields policy $\pi_\theta (x) \approx \pi_\star (x)$ directly. E.g.
  \[
  \nabla_\theta J (\pi_\theta) = \mathbb{E} [\nabla_\theta \pi_\theta \nabla_a Q_{\pi_\theta}]
  \]

- **Derivative-free methods**
  - Build a surrogate of $J (\pi_\theta)$
  - Optimize over that model
  - Difficult over large parameter spaces

Form function approximators:
- $Q_\theta (s, a)$, $V_\theta (s)$, $\pi_\theta (s)$
- via ad-hoc parametrization
Form function approximators:
\[ Q_\theta(s, a), \ V_\theta(s), \ \pi_\theta(s) \]
via ad-hoc parametrization

---

**Q-learning methods** adjust \( \theta \) to get
\[
Q_\theta(s, a) \approx Q_*(s, a)
\]
Yields policy:
\[
\pi_\theta(s) = \min_a Q_\theta(s, a) \approx \min_a Q_*(s, a) = \pi_*(s)
\]
E.g. basic Q-learning uses:
\[
\theta \leftarrow \theta + \alpha \delta \nabla_\theta Q_\theta(s_k, a_k)
\]
\[
\delta = L(s_k, a_k) + \gamma V_\theta(s_{k+1}) - Q_\theta(s_k, a_k)
\]

**Policy gradient methods** adjust \( \theta \) to get
\[
\nabla_\theta J(\pi_\theta) = 0
\]
yields policy \( \pi_\theta(x) \approx \pi_*(x) \) directly. E.g.
\[
\nabla_\theta J(\pi_\theta) = \mathbb{E} [\nabla_\theta \pi_\theta \nabla_a Q_{\pi_\theta}]
\]

**Derivative-free methods**
- Build a surrogate of \( J(\pi_\theta) \)
- Optimize over that model
- Difficult over large parameter spaces
**RL methods - Reminder**

- **Q-learning methods** adjust $\theta$ to get
  \[
  Q_\theta (s, a) \approx Q_\star (s, a)
  \]
  Yields policy:
  \[
  \pi_\theta (s) = \min_a Q_\theta (s, a) \approx \min_a Q_\star (s, a) = \pi_\star (s)
  \]
  E.g. basic Q-learning uses:
  \[
  \theta \leftarrow \theta + \alpha \delta \nabla_\theta Q_\theta (s_k, a_k)
  \]
  \[
  \delta = L (s_k, a_k) + \gamma V_\theta (s_{k+1}) - Q_\theta (s_k, a_k)
  \]

- **Policy gradient methods** adjust $\theta$ to get
  \[
  \nabla_\theta J(\pi_\theta) = 0
  \]
  yields policy $\pi_\theta (x) \approx \pi_\star (x)$ directly. E.g.
  \[
  \nabla_\theta J(\pi_\theta) = \mathbb{E} [\nabla_\theta \pi_\theta \nabla_a Q_{\pi_\theta}]
  \]

- **Derivative-free methods**
  - Build a surrogate of $J(\pi_\theta)$
  - Optimize over that model
  - Difficult over large parameter spaces

In the RL-MPC context, $Q_\theta$, $V_\theta$, $\pi_\theta$ are coming from an MPC scheme, typically cast as Nonlinear Program. What about the sensitivities?

Form function approximators:
- $Q_\theta (s, a)$, $V_\theta (s)$, $\pi_\theta (s)$
  via ad-hoc parametrization

Derivative-based methods require $Q_\theta$, $V_\theta$, $\pi_\theta$ and computing their sensitivities (i.e. $\nabla_\theta$ or $\frac{\partial}{\partial \theta}$)
Implementation of Basic RL Algorithms for MPC

**MPC is a Nonlinear Program**

Optimal value

\[ V_{\theta}(s) = \min_w \Phi(w, s, \theta) \]

\[ \text{s.t. } g(w, s, \theta) = 0 \]

\[ h(w, s, \theta) \leq 0 \]

Optimal solution

\[ w^*_\theta(s) = \min_w \Phi(w, s, \theta) \]

\[ \text{s.t. } \ldots \]
Implementation of Basic RL Algorithms for MPC

**MPC is a Nonlinear Program**

Optimal value

\[ V_{\theta} (s) = \min_w \Phi (w, s, \theta) \]

s.t.

\[ g (w, s, \theta) = 0 \]
\[ h (w, s, \theta) \leq 0 \]

Optimal solution

\[ w_{\theta}^* (s) = \min_w \Phi (w, s, \theta) \]

s.t.

... ?

How to obtain:

\[ \nabla_{\theta} V_{\theta}, \nabla_{\theta} Q_{\theta}, \nabla_{\theta} w_{\theta}^* \]
Implementation of Basic RL Algorithms for MPC

**MPC is a Nonlinear Program**

Optimal value

\[ V_\theta (s) = \min_w \Phi (w, s, \theta) \]

s.t. \[ g (w, s, \theta) = 0 \]
\[ h (w, s, \theta) \leq 0 \]

Optimal solution

\[ w_\theta^* (s) = \min_w \Phi (w, s, \theta) \]

s.t. \[ \ldots \]

NLP solution satisfies (KKT conditions)

\[ r = \begin{bmatrix} \nabla_w \mathcal{L} \\ g \\ h_i \mu_i \end{bmatrix} = 0 \]

\[ h \leq 0, \mu \geq 0 \]

where Lagrange function is

\[ \mathcal{L} = \Phi + \lambda^\top g + \mu^\top h \]

and \( \lambda, \mu \) are “auxiliary variables” (multipliers)

How to obtain:

\[ \nabla_\theta V_\theta, \ \nabla_\theta Q_\theta, \ \nabla_\theta w_\theta^* \]

\[ ? \]
MPC is a Nonlinear Program

Optimal value

\[ V_\theta (s) = \min_w \Phi (w, s, \theta) \]

s.t. \[ g (w, s, \theta) = 0 \]
\[ h (w, s, \theta) \leq 0 \]

Optimal solution

\[ w_\theta^* (s) = \min_w \Phi (w, s, \theta) \]

s.t. \[ \ldots \]

NLP solution satisfies (KKT conditions)

\[
\begin{bmatrix}
\nabla_w \mathcal{L} \\
g \\
h_i \mu_i
\end{bmatrix} = 0
\]

\[ h \leq 0, \mu \geq 0 \]

where Lagrange function is

\[ \mathcal{L} = \Phi + \lambda ^\top g + \mu ^\top h \]

and \( \lambda, \mu \) are “auxiliary variables” (multipliers)

How to obtain:

\[ \nabla_\theta V_\theta, \nabla_\theta Q_\theta, \nabla_\theta w_\theta^* \]

?
Implementation of Basic RL Algorithms for MPC

MPC is a Nonlinear Program

Optimal value
\[ V_\theta (s) = \min_w \Phi (w, s, \theta) \]
\[ \text{s.t. } g(w, s, \theta) = 0 \]
\[ h(w, s, \theta) \leq 0 \]

Optimal solution
\[ w^*_\theta (s) = \min_w \Phi (w, s, \theta) \]
\[ \text{s.t. } \ldots \]

How to obtain:
\[ \nabla_\theta V_\theta, \nabla_\theta Q_\theta, \nabla_\theta w^*_\theta \]

NLP solution satisfies (KKT conditions)
\[ r = \begin{bmatrix} \nabla_w \mathcal{L} \\ g \\ h_i \mu_i \end{bmatrix} = 0 \]
\[ h \leq 0, \mu \geq 0 \]

where Lagrange function is
\[ \mathcal{L} = \Phi + \lambda^\top g + \mu^\top h \]

and \( \lambda, \mu \) are “auxiliary variables” (multipliers)

Solve NLP for \( x, \theta \), provides \( w, \lambda, \mu \), then:
\[ \nabla_\theta V_\theta (s) = \nabla_\theta \mathcal{L} (w, s, \theta, \lambda, \mu) \]

is a simple function evaluation
Implementation of Basic RL Algorithms for MPC

MPC is a Nonlinear Program

Optimal value

\[ V_\theta(s) = \min_w \Phi(w, s, \theta) \]

s.t. \( g(w, s, \theta) = 0 \)
\( h(w, s, \theta) \leq 0 \)

Optimal solution

\[ w_\theta^*(s) = \min_w \Phi(w, s, \theta) \]

s.t. \( \ldots \)

NLP solution satisfies (KKT conditions)

\[ r = \begin{bmatrix} \nabla_w \mathcal{L} \\ g \\ h_i \mu_i \end{bmatrix} = 0 \]

\( h \leq 0, \mu \geq 0 \)

where Lagrange function is

\[ \mathcal{L} = \Phi + \lambda^\top g + \mu^\top h \]

and \( \lambda, \mu \) are “auxiliary variables” (multipliers)

Solve NLP for \( s, \theta \), provides \( w, \lambda, \mu \), then:

\[ \frac{\partial w_\theta^*}{\partial \theta} = -\frac{\partial r}{\partial w}^{-1} \frac{\partial r}{\partial \theta} \]

with \( \frac{\partial r}{\partial w}^{-1} \) already built in the solver, exists if LICQ / SOSC

How to obtain:

\( \nabla_\theta V_\theta, \nabla_\theta Q_\theta, \nabla_\theta w_\theta^* \)

\?
Implementation of Basic RL Algorithms for MPC

MPC is a Nonlinear Program

Optimal value

\[ V_\theta(s) = \min_w \Phi(w, s, \theta) \]

s.t. \[ g(w, s, \theta) = 0 \]

\[ h(w, s, \theta) \leq 0 \]

Optimal solution

\[ w^*_\theta(s) = \min_w \Phi(w, s, \theta) \]

s.t. \[ \ldots \]

NLP solution satisfies (KKT conditions)

\[ r = \begin{bmatrix} \nabla_w \mathcal{L} \\ g \\ h_i \mu_i \end{bmatrix} = 0 \]

\[ h \leq 0, \mu \geq 0 \]

where Lagrange function is

\[ \mathcal{L} = \Phi + \lambda^\top g + \mu^\top h \]

and \( \lambda, \mu \) are “auxiliary variables” (multipliers)

Sensitivities do not exist for all \( s, a \). Does that matter?

How to obtain:

\[ \nabla_\theta V_\theta, \nabla_\theta Q_\theta, \nabla_\theta w^*_\theta \]

?
Implementation of Basic RL Algorithms for MPC

**MPC is a Nonlinear Program**

Optimal value

\[ V_\theta (s) = \min_w \Phi (w, s, \theta) \]

s.t. \( g (w, s, \theta) = 0 \)
\( h (w, s, \theta) \leq 0 \)

Optimal solution

\[ w^*_\theta (s) = \min_w \Phi (w, s, \theta) \]

s.t.  

NLP solution satisfies (KKT conditions)

\[ \begin{bmatrix} \nabla_w \mathcal{L} \\ g \\ h_i \mu_i \end{bmatrix} = 0 \]

\( h \leq 0, \mu \geq 0 \)

where Lagrange function is

\[ \mathcal{L} = \Phi + \lambda^\top g + \mu^\top h \]

and \( \lambda, \mu \) are "auxiliary variables" (multipliers)

**Sensitivities do not exist for all s, a. Does that matter?**

In general no: they exist *almost everywhere*, and always appear inside \( \mathbb{E} [\cdot] \). If the MDP has well-defined underlying densities, then we are good.
Model-based RL methods vs. RL-MPC: Data flow

Common setup for “classic RL:
- Build statistical model of the real system
- Generate simulated samples
- Feed RL with real and simulated samples

Remarks:
- Simulated data much cheaper than real ones, most data will be simulated ones
- With mostly simulated data:
  - $\approx$equivalent to approximate DP
  - policy optimality relies on model quality
Model-based RL methods vs. RL-MPC: Data flow

Basic setup for “RL-MPC”:
- Build MPC model of the real system
- Pass it to MPC scheme
- Feed RL with real samples

Remarks:
- RL tunes MPC for real system
- MPC model may be “detuned” from SYSID version
- Real data are expensive...
Model-based RL methods vs. RL-MPC: Data flow

“Mixed” setup for “RL-MPC”:
- Build MPC model of the real system
- MPC model is typically “simple”
- Build statistical model of the real system
- Generate simulated samples
- Feed RL with real and simulated samples

Remarks:
- Simple MPC model
- Complex simulation model
- MPC model may be “detuned” from SYSID version
What did we discuss?

- Learning-based MPC: we accept that the **MPC model will never be “right”**, seek closed-loop performance rather than model fitting.

- MPC serves as a **policy & value functions approximation**. This is a classic object in RL, but MPC is **highly structured**, while classic approximations in RL are not.

- **Modifying the MPC cost and constraints** allows MPC to be close-to optimality despite inaccurate model.

- ... but it is also **formally justified**: in principle it allows to capture the optimal policy and value functions with a wrong model.

- We discussed how to implement RL methods on MPC (basics).

- There is still room for high-fidelity modelling, can be used to produce virtual training data.
So what’s next?

- **Stability** of MPC under learning?
- **Safety** of MPC under learning?

General MPC stability theory for deterministic, undiscounted problems. How to extend it to MDPs?

Some more results:

- Bias in policy gradient methods with constrained policies
- Combining RL and SYSID?
- RL and MPC for mixed-integer problems?
- RL and MPC with state observers?
- RL and MPC with strongly economic policies?
- RL for tuning the “meta” MPC parameters?