Differentiating MPC with applications in Reinforcement Learning

Mario Zanon

IMT School for Advanced Studies Lucca

Joint work with Sébastien Gros

ACC 2023

MPC

Iteratively solve:

$$
\begin{aligned}
\mathbf{u}^\star(\mathrm{s},\boldsymbol{\theta}),\mathbf{x}^\star(\mathrm{s},\boldsymbol{\theta}) = \ & \arg\min_{\mathbf{u},\mathbf{x}} \ V^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N)+\sum_{k=0}^{N-1}\ell_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k) \\
& \mathrm{s.t.}\ \mathbf{x}_0 = \mathbf{s}, \\
& \qquad \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right), \\
& \qquad \mathbf{h}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right) \leq 0, \\
& \qquad \mathbf{h}^{\mathrm{f}}_{\boldsymbol{\theta}}\left(\mathbf{x}_N\right) \leq 0,
\end{aligned}
$$

MPC

Iteratively solve:

$$\begin{aligned} \mathbf{u}^\star(\mathrm{s},\boldsymbol{\theta}),\mathbf{x}^\star(\mathrm{s},\boldsymbol{\theta}) = \ & \arg\min_{\mathbf{u},\mathbf{x}} \ V^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N) + \sum_{k=0}^{N-1} \ell_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k) \\ & \quad \mathrm{s.t.} \ \mathbf{x}_0 = \mathbf{s}, \\ & \qquad\quad \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right), \\ & \qquad\quad \mathbf{h}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right) \leq 0, \\ & \qquad\quad \mathbf{h}^{\mathrm{f}}_{\boldsymbol{\theta}}\left(\mathbf{x}_N\right) \leq 0, \end{aligned}$$

Optimal policy: $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s}) := \mathbf{u}^\star_0(\mathrm{s},\boldsymbol{\theta})$ deterministic

MPC

Iteratively solve:

$$\begin{aligned} \mathbf{u}^\star(\mathrm{s},\boldsymbol{\theta}),\mathbf{x}^\star(\mathrm{s},\boldsymbol{\theta}) = \ & \arg\min_{\mathbf{u},\mathbf{x}} \ V^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N) + \sum_{k=0}^{N-1} \ell_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k) \\ & \quad \mathrm{s.t.} \ \mathbf{x}_0 = \mathbf{s}, \\ & \quad \qquad \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right), \\ & \quad \qquad \mathbf{h}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right) \leq 0, \\ & \quad \qquad \mathbf{h}^{\mathrm{f}}_{\boldsymbol{\theta}}\left(\mathbf{x}_N\right) \leq 0, \end{aligned}$$

Optimal policy: $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s}) := \mathbf{u}_0^\star(\mathrm{s},\boldsymbol{\theta})$ deterministic

Remarks:

MPC

Iteratively solve:

$$\begin{aligned}\mathbf{u}^\star(\mathrm{s},\boldsymbol{\theta}),\mathbf{x}^\star(\mathrm{s},\boldsymbol{\theta}) = \ & \arg\min_{\mathbf{u},\mathbf{x}} \ V^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N)+\sum_{k=0}^{N-1}\ell_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k)\\ & \ \mathrm{s.t.}\ \ \mathbf{x}_0=\mathbf{s},\\ & \qquad\ \ \mathbf{x}_{k+1}=\mathbf{f}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right),\\ & \qquad\ \ \mathbf{h}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right)\le 0,\\ & \qquad\ \ \mathbf{h}^{\mathrm{f}}_{\boldsymbol{\theta}}\left(\mathbf{x}_N\right)\le 0,\end{aligned}$$

Optimal policy: $\boldsymbol{\pi}_\theta(\mathrm{s}) := \mathbf{u}_0^\star(\mathrm{s},\boldsymbol{\theta})$ deterministic

Remarks:

- Typically
  - $\mathbf{f}_{\boldsymbol{\theta}}$ “exact” or from system identification
  - $\ell_{\boldsymbol{\theta}}, \mathbf{h}_{\boldsymbol{\theta}}$ given
  - $\mathbf{h}^{\mathrm{f}}_{\boldsymbol{\theta}}$ invariant set, $V^{\mathrm{f}}_{\boldsymbol{\theta}}$ Lyapunov function

MPC

Iteratively solve:

$$\begin{aligned}\mathbf{u}^\star(\mathrm{s},\boldsymbol{\theta}),\mathbf{x}^\star(\mathrm{s},\boldsymbol{\theta}) = \ & \arg\min_{\mathbf{u},\mathbf{x}} \ V^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N)+\sum_{k=0}^{N-1}\ell_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k)\\ & \quad \mathrm{s.t.}\ \mathbf{x}_0=\mathbf{s},\\ & \qquad\quad \mathbf{x}_{k+1}=\mathbf{f}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right),\\ & \qquad\quad \mathbf{h}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right)\leq 0,\\ & \qquad\quad \mathbf{h}^{\mathrm{f}}_{\boldsymbol{\theta}}\left(\mathbf{x}_N\right)\leq 0,\end{aligned}$$

Optimal policy: $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s}) := \mathbf{u}_0^\star(\mathrm{s},\boldsymbol{\theta})$ deterministic

Remarks:

- Typically
  - $\mathbf{f}_{\boldsymbol{\theta}}$ "exact" or from system identification
  - $\ell_{\boldsymbol{\theta}}, \mathbf{h}_{\boldsymbol{\theta}}$ given
  - $\mathbf{h}^{\mathrm{f}}_{\boldsymbol{\theta}}$ invariant set, $V^{\mathrm{f}}_{\boldsymbol{\theta}}$ Lyapunov function
- If $\mathbf{f}_{\boldsymbol{\theta}}$ exact and $N\rightarrow\infty$: asymptotic stability + optimal closed-loop performance

MPC

Iteratively solve:

$$\begin{aligned}\mathbf{u}^\star(\mathrm{s},\boldsymbol{\theta}),\mathbf{x}^\star(\mathrm{s},\boldsymbol{\theta}) = \ & \arg\min_{\mathbf{u},\mathbf{x}} \ V^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N)+\sum_{k=0}^{N-1}\ell_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k)\\ & \mathrm{s.t.}\ \mathbf{x}_0 = \mathbf{s},\\ & \qquad \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right),\\ & \qquad \mathbf{h}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right) \leq 0,\\ & \qquad \mathbf{h}^{\mathrm{f}}_{\boldsymbol{\theta}}\left(\mathbf{x}_N\right) \leq 0,\end{aligned}$$

Optimal policy: $\boldsymbol{\pi}_\theta(\mathrm{s}) := \mathbf{u}_0^\star(\mathrm{s},\boldsymbol{\theta})$ deterministic

Remarks:

- Typically
  - $\mathbf{f}_{\boldsymbol{\theta}}$ "exact" or from system identification
  - $\ell_{\boldsymbol{\theta}}, \mathbf{h}_{\boldsymbol{\theta}}$ given
  - $\mathbf{h}^{\mathrm{f}}_{\boldsymbol{\theta}}$ invariant set, $V^{\mathrm{f}}_{\boldsymbol{\theta}}$ Lyapunov function
- If $\mathbf{f}_{\boldsymbol{\theta}}$ exact and $N\to\infty$: asymptotic stability + optimal closed-loop performance
- In practice: model inaccuracy $\implies$ some form of practical stability and suboptimal

MPC

Iteratively solve:

$$\begin{aligned}\mathbf{u}^\star(\mathrm{s},\boldsymbol{\theta}),\mathbf{x}^\star(\mathrm{s},\boldsymbol{\theta}) = \ & \arg\min_{\mathbf{u},\mathbf{x}} \ V_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N)+\sum_{k=0}^{N-1}\ell_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k)\\ & \quad \mathrm{s.t.}\ \mathbf{x}_0=\mathbf{s},\\ & \qquad\quad \mathbf{x}_{k+1}=\mathbf{f}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right),\\ & \qquad\quad \mathbf{h}_{\boldsymbol{\theta}}\left(\mathbf{x}_k,\mathbf{u}_k\right)\leq 0,\\ & \qquad\quad \mathbf{h}_{\boldsymbol{\theta}}^{\mathrm{f}}\left(\mathbf{x}_N\right)\leq 0,\end{aligned}$$

Optimal policy: $\boldsymbol{\pi}_\theta(\mathrm{s}) := \mathbf{u}_0^\star(\mathrm{s},\boldsymbol{\theta})$ deterministic

Remarks:

- Typically
  - $\mathbf{f}_{\boldsymbol{\theta}}$ "exact" or from system identification
  - $\ell_{\boldsymbol{\theta}},\mathbf{h}_{\boldsymbol{\theta}}$ given
  - $\mathbf{h}_{\boldsymbol{\theta}}^{\mathrm{f}}$ invariant set, $V_{\boldsymbol{\theta}}^{\mathrm{f}}$ Lyapunov function
- If $\mathbf{f}_{\boldsymbol{\theta}}$ exact and $N\to\infty$: asymptotic stability + optimal closed-loop performance
- In practice: model inaccuracy $\implies$ some form of practical stability and suboptimal
- Let RL adapt $\boldsymbol{\theta}$ to recover optimality

Reinforcement Learning

- A technique to solve a stochastic optimal control problem
  - Many similarities with Dynamic Programming
  - Based on sampling
  - Model based / model free

Reinforcement Learning

- A technique to solve a stochastic optimal control problem
  - Many similarities with Dynamic Programming
  - Based on sampling
  - Model based / model free
- Cross-disciplinary approach
  - Computer Science
  - Optimal Control
  - Systems / Control Theory
  - Statistics

Reinforcement Learning

- A technique to solve a stochastic optimal control problem
  - Many similarities with Dynamic Programming
  - Based on sampling
  - Model based / model free
- Cross-disciplinary approach
  - Computer Science
  - Optimal Control
  - Systems / Control Theory
  - Statistics
- Important success stories
  - Aerobatic helicopter flight
  - Win against Chess, Shogi and Go masters
  - Learn to beat the best Chess algorithm in a matter of few hours

Reinforcement Learning

- A technique to solve a stochastic optimal control problem
  - Many similarities with Dynamic Programming
  - Based on sampling
  - Model based / model free
- Cross-disciplinary approach
  - Computer Science
  - Optimal Control
  - Systems / Control Theory
  - Statistics
- Important success stories
  - Aerobatic helicopter flight
  - Win against Chess, Shogi and Go masters
  - Learn to beat the best Chess algorithm in a matter of few hours
- Limitations
  - Theoretical foundation of some algorithms not fully developed
  - **No stability and safety guarantees**

Reinforcement Learning

- A technique to solve a stochastic optimal control problem
  - Many similarities with Dynamic Programming
  - Based on sampling
  - Model based / model free
- Cross-disciplinary approach
  - Computer Science
  - Optimal Control
  - Systems / Control Theory
  - Statistics
- Important success stories
  - Aerobatic helicopter flight
  - Win against Chess, Shogi and Go masters
  - Learn to beat the best Chess algorithm in a matter of few hours
- Limitations
  - Theoretical foundation of some algorithms not fully developed
  - **No stability and safety guarantees**
- Advantages
  - **Optimality for the true system**

RL in a Nutshell

- Markov Chain defined by

$$\mathbb{P}[\,\mathbf{s}_+ \,|\, \mathbf{s}, \mathbf{a}\,] \qquad \text{equivalent of} \qquad \mathbf{s}_+ = \mathbf{f}(\mathbf{s}, \mathbf{a}, \mathbf{w})$$

  probability (density) of transitioning from state $\mathbf{s}$ to state $\mathbf{s}_+$ when taking action $\mathbf{a}$

RL in a Nutshell

- Markov Chain defined by

$$\mathbb{P}[\mathbf{s}_+ \,|\, \mathbf{s}, \mathbf{a}\,] \qquad \text{equivalent of} \qquad \mathbf{s}_+ = \mathbf{f}(\mathbf{s}, \mathbf{a}, \mathbf{w})$$

  probability (density) of transitioning from state $\mathbf{s}$ to state $\mathbf{s}_+$ when taking action $\mathbf{a}$
- Stochastic policy (includes deterministic as special case)

$$\pi\,[\,\mathbf{a}\,|\,\mathbf{s}\,] \in \mathbb{R}_+ \qquad \text{deterministic: } \mathbf{a} = \boldsymbol{\pi}(\mathbf{s})$$

  assigns the probability (density) of taking action $\mathbf{a}$ for a given state $\mathbf{s}$

RL in a Nutshell

- Markov Chain defined by

$$\mathbb{P}[\,\mathbf{s}_+ \,|\, \mathbf{s}, \mathbf{a}\,] \qquad \text{equivalent of} \qquad \mathbf{s}_+ = \mathbf{f}(\mathbf{s}, \mathbf{a}, \mathbf{w})$$

  probability (density) of transitioning from state $\mathbf{s}$ to state $\mathbf{s}_+$ when taking action $\mathbf{a}$
- Stochastic policy (includes deterministic as special case)

$$\pi\,[\,\mathbf{a}\,|\,\mathbf{s}\,] \in \mathbb{R}_+ \qquad \text{deterministic: } \mathbf{a} = \boldsymbol{\pi}(\mathbf{s})$$

  assigns the probability (density) of taking action $\mathbf{a}$ for a given state $\mathbf{s}$
- Expected discounted cost (return):

$$J(\pi) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k L(\mathbf{s}_k, \mathbf{a}_k)\right]$$

  where $\mathbf{a}_k$ is drawn from policy π. Initial conditions $\mathbf{s}_0$ can be fixed or random.
- Discount factor $\gamma \in [0, 1]$

RL in a Nutshell

- Markov Chain defined by

$$\mathbb{P}[\mathbf{s}_+ \,|\, \mathbf{s}, \mathbf{a}] \qquad \text{equivalent of} \qquad \mathbf{s}_+ = \mathbf{f}(\mathbf{s}, \mathbf{a}, \mathbf{w})$$

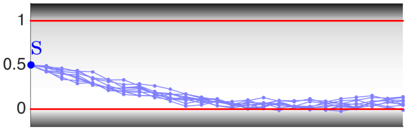
  probability (density) of transitioning from state $\mathbf{s}$ to state $\mathbf{s}_+$ when taking action $\mathbf{a}$
- Stochastic policy (includes deterministic as special case)

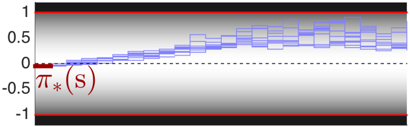
$$\pi[\mathbf{a} \,|\, \mathbf{s}] \in \mathbb{R}_+ \qquad \text{deterministic: } \mathbf{a} = \boldsymbol{\pi}(\mathbf{s})$$

  assigns the probability (density) of taking action $\mathbf{a}$ for a given state $\mathbf{s}$
- Expected discounted cost (return):

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k L(\mathbf{s}_k, \mathbf{a}_k) \right]$$

  where $\mathbf{a}_k$ is drawn from policy π. Initial conditions $\mathbf{s}_0$ can be fixed or random.
- Discount factor $\gamma \in [0, 1]$





RL in a Nutshell

- Markov Chain defined by

$$\mathbb{P}[\mathbf{s}_+ \,|\, \mathbf{s}, \mathbf{a}] \qquad \text{equivalent of} \qquad \mathbf{s}_+ = \mathbf{f}(\mathbf{s}, \mathbf{a}, \mathbf{w})$$

probability (density) of transitioning from state $\mathbf{s}$ to state $\mathbf{s}_+$ when taking action $\mathbf{a}$
- Stochastic policy (includes deterministic as special case)

$$\pi\left[\mathbf{a} \,|\, \mathbf{s}\right] \in \mathbb{R}_+ \qquad \text{deterministic: } \mathbf{a} = \boldsymbol{\pi}(\mathbf{s})$$

assigns the probability (density) of taking action $\mathbf{a}$ for a given state $\mathbf{s}$
- Expected discounted cost (return):

$$J(\pi) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k L(\mathbf{s}_k, \mathbf{a}_k)\right]$$

where $\mathbf{a}_k$ is drawn from policy π. Initial conditions $\mathbf{s}_0$ can be fixed or random.
- Discount factor $\gamma \in [0, 1]$
- Markov Decision Process (MDP): find $\pi_\star$ solution of

$$\pi_\star := \arg\min_\pi \; J(\pi)$$

RL in a Nutshell

- Markov Chain defined by

$$\mathbb{P}[\,\mathbf{s}_+ \,|\, \mathbf{s}, \mathbf{a}\,] \qquad \text{equivalent of} \qquad \mathbf{s}_+ = \mathbf{f}(\mathbf{s}, \mathbf{a}, \mathbf{w})$$

  probability (density) of transitioning from state $\mathbf{s}$ to state $\mathbf{s}_+$ when taking action $\mathbf{a}$
- Stochastic policy (includes deterministic as special case)

$$\pi\,[\,\mathbf{a}\,|\,\mathbf{s}\,] \in \mathbb{R}_+ \qquad \text{deterministic: } \mathbf{a} = \boldsymbol{\pi}(\mathbf{s})$$

  assigns the probability (density) of taking action $\mathbf{a}$ for a given state $\mathbf{s}$
- Expected discounted cost (return):

$$J\,(\pi) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k L\,(\mathbf{s}_k, \mathbf{a}_k) \right]$$

  where $\mathbf{a}_k$ is drawn from policy π. Initial conditions $\mathbf{s}_0$ can be fixed or random.
- Discount factor $\gamma \in [0, 1]$
- Markov Decision Process (MDP): find $\pi_\star$ solution of

$$\pi_\star := \arg\min_\pi \; J\,(\pi)$$

- Optimal parametrized policy $\pi_{\boldsymbol{\theta}}$ given by:

$$\boldsymbol{\theta}_\star := \arg\min_{\boldsymbol{\theta}} \; J\,(\pi_{\boldsymbol{\theta}})$$

RL in a Nutshell

- Markov Chain defined by

$$\mathbb{P}[\,\mathbf{s}_+ \,|\, \mathbf{s}, \mathbf{a}\,] \qquad \text{equivalent of} \qquad \mathbf{s}_+ = \mathbf{f}(\mathbf{s}, \mathbf{a}, \mathbf{w})$$

  probability (density) of transitioning from state $\mathbf{s}$ to state $\mathbf{s}_+$ when taking action $\mathbf{a}$
- Stochastic policy (includes deterministic as special case)

$$\pi\,[\,\mathbf{a}\,|\,\mathbf{s}\,] \in \mathbb{R}_+ \qquad \text{deterministic: } \mathbf{a} = \boldsymbol{\pi}(\mathbf{s})$$

  assigns the probability (density) of taking action $\mathbf{a}$ for a given state $\mathbf{s}$
- Expected discounted cost (return):

$$J\,(\pi) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k L\,(\mathbf{s}_k, \mathbf{a}_k) \right]$$

  where $\mathbf{a}_k$ is drawn from policy π. Initial conditions $\mathbf{s}_0$ can be fixed or random.
- Discount factor $\gamma \in [0, 1]$
- Markov Decision Process (MDP): find $\pi_\star$ solution of

$$\pi_\star := \arg\min_{\pi}\ J\,(\pi)$$

- Optimal parametrized policy $\pi_{\boldsymbol{\theta}}$ given by:

$$\boldsymbol{\theta}_\star := \arg\min_{\boldsymbol{\theta}}\ J\,(\pi_{\boldsymbol{\theta}})$$

RL (approximately) solves the MDP in a sample-based fashion.

Value functions

- **Value function**:

$$V_\pi(\mathbf{s}) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k L(\mathbf{s}_k, \mathbf{a}_k) \,\middle|\, \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_k \sim \pi[.\,|\,\mathbf{s}_k] \right]$$

gives the expected cost for policy π, starting from given initial conditions $\mathbf{s}$

Value functions

- **Value function**:

$$V_\pi(\mathbf{s}) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k L(\mathbf{s}_k, \mathbf{a}_k) \,\middle|\, \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_k \sim \pi[.\,|\,\mathbf{s}_k]\right]$$

  gives the expected cost for policy π, starting from given initial conditions $\mathbf{s}$

- **Note that**: $J(\pi) = \mathbb{E}_{\rho^{\mathbf{s}}}[V_\pi(\mathbf{s})]$, for some initial probability (density) $\rho^{\mathbf{s}}$

Value functions

- **Value function**:

$$V_\pi(\mathbf{s}) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k L(\mathbf{s}_k, \mathbf{a}_k) \,\middle|\, \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_k \sim \pi[.\,|\,\mathbf{s}_k]\right]$$

gives the expected cost for policy π, starting from given initial conditions $\mathbf{s}$

- **Note that**: $J(\pi) = \mathbb{E}_{\rho^{\mathbf{s}}}[V_\pi(\mathbf{s})]$, for some initial probability (density) $\rho^{\mathbf{s}}$

- **Action-Value function**:

$$Q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k L(\mathbf{s}_k, \mathbf{a}_k) \,\middle|\, \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_0 = \mathbf{a},\ \mathbf{a}_{k>0} \sim \pi[.\,|\,\mathbf{s}_k]\right]$$

gives the expected cost for policy π, starting from given initial condition $\mathbf{s}$, and using action $\mathbf{a}$ as first input (policy π after that)

Value functions

- **Value function**:

$$V_\pi(\mathbf{s}) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k L(\mathbf{s}_k, \mathbf{a}_k) \,\middle|\, \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_k \sim \pi[.\,|\,\mathbf{s}_k]\right]$$

gives the expected cost for policy π, starting from given initial conditions $\mathbf{s}$

- **Note that**: $J(\pi) = \mathbb{E}_{\rho^{\mathrm{s}}}[V_\pi(\mathbf{s})]$, for some initial probability (density) ρ^{s}

- **Action-Value function**:

$$Q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k L(\mathbf{s}_k, \mathbf{a}_k) \,\middle|\, \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_0 = \mathbf{a},\ \mathbf{a}_{k>0} \sim \pi[.\,|\,\mathbf{s}_k]\right]$$

gives the expected cost for policy π, starting from given initial condition $\mathbf{s}$, and using action $\mathbf{a}$ as first input (policy π after that)

- **Optimal Value functions**
  - Notation:

$$V_\star(\mathbf{s}) = V_{\boldsymbol{\pi}_\star}(\mathbf{s}) \qquad\qquad Q_\star(\mathbf{s}, \mathbf{a}) = Q_{\boldsymbol{\pi}_\star}(\mathbf{s}, \mathbf{a})$$

Value functions

- **Value function**:

$$V_\pi(\mathbf{s}) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k L(\mathbf{s}_k, \mathbf{a}_k) \,\middle|\, \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_k \sim \pi[.\,|\,\mathbf{s}_k]\right]$$

gives the expected cost for policy π, starting from given initial conditions $\mathbf{s}$

- **Note that**: $J(\pi) = \mathbb{E}_{\rho^\mathbf{s}}[V_\pi(\mathbf{s})]$, for some initial probability (density) $\rho^\mathbf{s}$

- **Action-Value function**:

$$Q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k L(\mathbf{s}_k, \mathbf{a}_k) \,\middle|\, \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_0 = \mathbf{a},\ \mathbf{a}_{k>0} \sim \pi[.\,|\,\mathbf{s}_k]\right]$$

gives the expected cost for policy π, starting from given initial condition $\mathbf{s}$, and using action $\mathbf{a}$ as first input (policy π after that)

- **Optimal Value functions**
  - Notation:

$$V_\star(\mathbf{s}) = V_{\boldsymbol{\pi}_\star}(\mathbf{s}) \qquad\qquad Q_\star(\mathbf{s}, \mathbf{a}) = Q_{\boldsymbol{\pi}_\star}(\mathbf{s}, \mathbf{a})$$

  - Properties:

$$V_\star(\mathbf{s}) = \min_{\mathbf{a}} Q_\star(\mathbf{s}, \mathbf{a}) \qquad\qquad \boldsymbol{\pi}_\star(\mathbf{s}) = \arg\min_{\mathbf{a}} Q_\star(\mathbf{s}, \mathbf{a})$$

Reinforcement Learning - Algorithms

Form function approximators:

$$Q_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}), \quad V_{\boldsymbol{\theta}}(\mathbf{s}), \quad \boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s})$$

via ad-hoc parametrization

Reinforcement Learning - Algorithms

Form function approximators:

$$Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}), \; V_{\boldsymbol{\theta}}(\mathbf{s}), \; \boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s})$$

via ad-hoc parametrization

- *Q*-**learning methods** adjust $\boldsymbol{\theta}$ to get

$$\min_{\boldsymbol{\theta}} \mathbb{E}\left[\left(Q_\star(\mathbf{s},\mathbf{a}) - Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)^2\right]$$

Yields policy:

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s}) = \mathbf{a} \min_{\mathbf{a}} \; Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}) \approx \mathbf{a} \min_{\mathbf{a}} \; Q_\star(\mathbf{s},\mathbf{a}) = \boldsymbol{\pi}_\star(\mathbf{s})$$

Reinforcement Learning - Algorithms

Form function approximators:

$$Q_{\boldsymbol{\theta}}(\mathrm{s}, \mathrm{a}), \quad V_{\boldsymbol{\theta}}(\mathrm{s}), \quad \boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s})$$

via ad-hoc parametrization

- Q**-learning methods** adjust $\boldsymbol{\theta}$ to get

$$\min_{\boldsymbol{\theta}} \mathbb{E}\left[\left(Q_\star(\mathrm{s}, \mathrm{a}) - Q_{\boldsymbol{\theta}}(\mathrm{s}, \mathrm{a})\right)^2\right]$$

  Yields policy:

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s}) = \mathrm{a} \min_{\mathrm{a}} \; Q_{\boldsymbol{\theta}}(\mathrm{s}, \mathrm{a}) \approx \mathrm{a} \min_{\mathrm{a}} \; Q_\star(\mathrm{s}, \mathrm{a}) = \boldsymbol{\pi}_\star(\mathrm{s})$$

- **Policy gradient methods** adjust $\boldsymbol{\theta}$ to get

$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}}) \qquad \Leftrightarrow \qquad \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}}) = 0$$

  yields policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s}) \approx \boldsymbol{\pi}_\star(\mathrm{s})$ directly

Reinforcement Learning - Algorithms

Form function approximators:

$$Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}), \ V_{\boldsymbol{\theta}}(\mathbf{s}), \ \boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s})$$

via ad-hoc parametrization

- ***Q*-learning methods** adjust $\boldsymbol{\theta}$ to get

$$\min_{\boldsymbol{\theta}} \mathbb{E}\left[\left(Q_\star(\mathbf{s},\mathbf{a}) - Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)^2\right]$$

  Yields policy:

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s}) = \mathbf{a}\min_{\mathbf{a}} \ Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}) \approx \mathbf{a}\min_{\mathbf{a}} \ Q_\star(\mathbf{s},\mathbf{a}) = \boldsymbol{\pi}_\star(\mathbf{s})$$

- **Policy gradient methods** adjust $\boldsymbol{\theta}$ to get

$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}}) \qquad \Leftrightarrow \qquad \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}}) = 0$$

  yields policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s}) \approx \boldsymbol{\pi}_\star(\mathbf{s})$ directly

All approaches hinge on building either $Q_{\boldsymbol{\theta}}$ or $\{\boldsymbol{\pi}_{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}}\}$

Reinforcement Learning - Algorithms

Form function approximators:

$$Q_{\boldsymbol{\theta}}(\mathrm{s},\mathbf{a}), \quad V_{\boldsymbol{\theta}}(\mathrm{s}), \quad \boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s})$$

via ad-hoc parametrization

- *Q*-**learning methods** adjust $\boldsymbol{\theta}$ to get

$$\min_{\boldsymbol{\theta}} \mathbb{E}\left[\left(Q_\star(\mathrm{s},\mathbf{a}) - Q_{\boldsymbol{\theta}}(\mathrm{s},\mathbf{a})\right)^2\right]$$

  Yields policy:

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s}) = \mathbf{a} \min_{\mathbf{a}} \; Q_{\boldsymbol{\theta}}(\mathrm{s},\mathbf{a}) \approx \mathbf{a} \min_{\mathbf{a}} \; Q_\star(\mathrm{s},\mathbf{a}) = \boldsymbol{\pi}_\star(\mathrm{s})$$

- **Policy gradient methods** adjust $\boldsymbol{\theta}$ to get

$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}}) \qquad \Leftrightarrow \qquad \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}}) = 0$$

  yields policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s}) \approx \boldsymbol{\pi}_\star(\mathrm{s})$ directly

All approaches hinge on building either $Q_{\boldsymbol{\theta}}$ or $\{\boldsymbol{\pi}_{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}}\}$

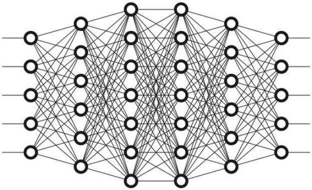
Most approaches are **derivative-based**: we need $\nabla_{\boldsymbol{\theta}}\boldsymbol{\pi}_{\boldsymbol{\theta}}, \nabla_{\boldsymbol{\theta}} V_{\boldsymbol{\theta}}, \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}$

Reinforcement Learning - Algorithms

Form function approximators:

$$Q_{\boldsymbol{\theta}}\left(\mathbf{s},\mathbf{a}\right), \;\; V_{\boldsymbol{\theta}}\left(\mathbf{s}\right), \;\; \boldsymbol{\pi}_{\boldsymbol{\theta}}\left(\mathbf{s}\right)$$

via ad-hoc parametrization



- *Q*-**learning methods** adjust $\boldsymbol{\theta}$ to get

$$\min_{\boldsymbol{\theta}} \mathbb{E}\left[\left(Q_{\star}\left(\mathbf{s},\mathbf{a}\right) - Q_{\boldsymbol{\theta}}\left(\mathbf{s},\mathbf{a}\right)\right)^2\right]$$

  Yields policy:

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}\left(\mathbf{s}\right) = \mathbf{a}\min_{\mathbf{a}}\; Q_{\boldsymbol{\theta}}\left(\mathbf{s},\mathbf{a}\right) \approx \mathbf{a}\min_{\mathbf{a}}\; Q_{\star}\left(\mathbf{s},\mathbf{a}\right) = \boldsymbol{\pi}_{\star}\left(\mathbf{s}\right)$$

- **Policy gradient methods** adjust $\boldsymbol{\theta}$ to get

$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}}) \qquad \Leftrightarrow \qquad \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}}) = 0$$

  yields policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}\left(\mathbf{s}\right) \approx \boldsymbol{\pi}_{\star}\left(\mathbf{s}\right)$ directly

All approaches hinge on building either $Q_{\boldsymbol{\theta}}$ or $\{\boldsymbol{\pi}_{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}}\}$

Most approaches are **derivative-based**: we need $\nabla_{\boldsymbol{\theta}}\boldsymbol{\pi}_{\boldsymbol{\theta}}, \nabla_{\boldsymbol{\theta}} V_{\boldsymbol{\theta}}, \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}$

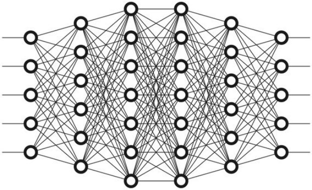
Nowadays RL typically relies on DNNs as function approximators:

Reinforcement Learning - Algorithms

Form function approximators:

$$Q_{\boldsymbol{\theta}}(\mathrm{s},\mathrm{a}), \quad V_{\boldsymbol{\theta}}(\mathrm{s}), \quad \boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s})$$

via ad-hoc parametrization



- *Q*-**learning methods** adjust $\boldsymbol{\theta}$ to get

$$\min_{\boldsymbol{\theta}} \mathbb{E}\left[\left(Q_\star(\mathrm{s},\mathrm{a}) - Q_{\boldsymbol{\theta}}(\mathrm{s},\mathrm{a})\right)^2\right]$$

Yields policy:

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s}) = \mathrm{a}\min_{\mathrm{a}} \; Q_{\boldsymbol{\theta}}(\mathrm{s},\mathrm{a}) \approx \mathrm{a}\min_{\mathrm{a}} \; Q_\star(\mathrm{s},\mathrm{a}) = \boldsymbol{\pi}_\star(\mathrm{s})$$

- **Policy gradient methods** adjust $\boldsymbol{\theta}$ to get

$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}}) \qquad \Leftrightarrow \qquad \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}}) = 0$$

yields policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s}) \approx \boldsymbol{\pi}_\star(\mathrm{s})$ directly

All approaches hinge on building either $Q_{\boldsymbol{\theta}}$ or $\{\boldsymbol{\pi}_{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}}\}$

Most approaches are **derivative-based**: we need $\nabla_{\boldsymbol{\theta}}\boldsymbol{\pi}_{\boldsymbol{\theta}}, \nabla_{\boldsymbol{\theta}} V_{\boldsymbol{\theta}}, \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}$

Nowadays RL typically relies on DNNs as function approximators:
difficult to understand, no strong guarantees

Reinforcement Learning - Algorithms

Form function approximators:

$$Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}),\ \ V_{\boldsymbol{\theta}}(\mathbf{s}),\ \ \boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s})$$

via ad-hoc parametrization

$$\begin{aligned}\min_{\mathbf{u},\mathbf{x}}\ & V^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N)+\sum_{k=0}^{N-1}\ell_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k)\\ \text{s.t.}\ & \mathbf{x}_0=\mathbf{s},\\ & \mathbf{x}_{k+1}=\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k),\\ & \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k)\leq 0,\\ & \mathbf{h}^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N)\leq 0,\end{aligned}$$

- Q**-learning methods** adjust $\boldsymbol{\theta}$ to get

$$\min_{\boldsymbol{\theta}}\ \mathbb{E}\left[\left(Q_\star(\mathbf{s},\mathbf{a})-Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)^2\right]$$

Yields policy:

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s})=\mathbf{a}\min_{\mathbf{a}}\ Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\approx\mathbf{a}\min_{\mathbf{a}}\ Q_\star(\mathbf{s},\mathbf{a})=\boldsymbol{\pi}_\star(\mathbf{s})$$

- **Policy gradient methods** adjust $\boldsymbol{\theta}$ to get

$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}})\qquad\Leftrightarrow\qquad\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}})=0$$

yields policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s})\approx\boldsymbol{\pi}_\star(\mathbf{s})$ directly

All approaches hinge on building either $Q_{\boldsymbol{\theta}}$ or $\{\boldsymbol{\pi}_{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}}\}$

Most approaches are **derivative-based**: we need $\nabla_{\boldsymbol{\theta}}\boldsymbol{\pi}_{\boldsymbol{\theta}}, \nabla_{\boldsymbol{\theta}} V_{\boldsymbol{\theta}}, \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}$

Nowadays RL typically relies on DNNs as function approximators:
difficult to understand, no strong guarantees

Alternative: use MPC as function approximator

Reinforcement Learning - Algorithms

Form function approximators:

$$Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}), \quad V_{\boldsymbol{\theta}}(\mathbf{s}), \quad \boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s})$$

via ad-hoc parametrization

$$\begin{aligned}
\min_{\mathbf{u},\mathbf{x}} \quad & V^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N)+\sum_{k=0}^{N-1}\ell_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k) \\
\text{s.t.} \quad & \mathbf{x}_0=\mathbf{s}, \\
& \mathbf{x}_{k+1}=\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k), \\
& \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k)\le 0, \\
& \mathbf{h}^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N)\le 0,
\end{aligned}$$

- *Q*-**learning methods** adjust $\boldsymbol{\theta}$ to get

$$\min_{\boldsymbol{\theta}} \mathbb{E}\left[\left(Q_\star(\mathbf{s},\mathbf{a})-Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)^2\right]$$

  Yields policy:

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s})=\mathbf{a}\min_{\mathbf{a}}\ Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\approx \mathbf{a}\min_{\mathbf{a}}\ Q_\star(\mathbf{s},\mathbf{a})=\boldsymbol{\pi}_\star(\mathbf{s})$$

- **Policy gradient methods** adjust $\boldsymbol{\theta}$ to get

$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}}) \qquad \Leftrightarrow \qquad \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}})=0$$

  yields policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s})\approx\boldsymbol{\pi}_\star(\mathbf{s})$ directly

All approaches hinge on building either $Q_{\boldsymbol{\theta}}$ or $\{\boldsymbol{\pi}_{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}}\}$

Most approaches are **derivative-based**: we need $\nabla_{\boldsymbol{\theta}}\boldsymbol{\pi}_{\boldsymbol{\theta}}, \nabla_{\boldsymbol{\theta}} V_{\boldsymbol{\theta}}, \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}$

Nowadays RL typically relies on DNNs as function approximators:
difficult to understand, no strong guarantees

Alternative: use MPC as function approximator
this provides explainability and makes it possible to guarantee safety and stability

Reinforcement Learning - Algorithms

Form function approximators:

$$Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}), \quad V_{\boldsymbol{\theta}}(\mathbf{s}), \quad \boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s})$$

via ad-hoc parametrization

$$\begin{aligned}
\min_{\mathbf{u},\mathbf{x}} \quad & V^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N)+\sum_{k=0}^{N-1}\ell_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k) \\
\text{s.t.} \quad & \mathbf{x}_0=\mathbf{s}, \\
& \mathbf{x}_{k+1}=\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k), \\
& \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k)\leq 0, \\
& \mathbf{h}^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N)\leq 0,
\end{aligned}$$

- ***Q*-learning methods** adjust $\boldsymbol{\theta}$ to get

$$\min_{\boldsymbol{\theta}} \mathbb{E}\left[\left(Q_\star(\mathbf{s},\mathbf{a})-Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)^2\right]$$

  Yields policy:

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s})=\mathrm{a}\min_{\mathbf{a}}\ Q_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\approx \mathrm{a}\min_{\mathbf{a}}\ Q_\star(\mathbf{s},\mathbf{a})=\boldsymbol{\pi}_\star(\mathbf{s})$$

- **Policy gradient methods** adjust $\boldsymbol{\theta}$ to get

$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}}) \qquad \Leftrightarrow \qquad \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\pi}_{\boldsymbol{\theta}})=0$$

  yields policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s})\approx\boldsymbol{\pi}_\star(\mathbf{s})$ directly

All approaches hinge on building either $Q_{\boldsymbol{\theta}}$ or $\{\boldsymbol{\pi}_{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}}\}$

Most approaches are **derivative-based**: we need $\nabla_{\boldsymbol{\theta}}\boldsymbol{\pi}_{\boldsymbol{\theta}}, \nabla_{\boldsymbol{\theta}} V_{\boldsymbol{\theta}}, \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}$

Nowadays RL typically relies on DNNs as function approximators:
difficult to understand, no strong guarantees

Alternative: use MPC as function approximator
this provides explainability and makes it possible to guarantee safety and stability
we need to differentiate MPC!

MPC as a Function Approximator

MPC Tuning parameter $\boldsymbol{\theta}$, initial state s

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} \ell_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{u}) + V_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \\
\text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s} \\
& \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \\
& \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \leq 0 \\
& \mathbf{h}_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \leq 0
\end{aligned}$$

MPC as a Function Approximator

MPC Tuning parameter $\boldsymbol{\theta}$, initial state s

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s}) = \mathbf{u}_0^\star \qquad \begin{aligned} \mathbf{x}^\star, \mathbf{u}^\star = \arg\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} \ell_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) + V_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \\ \text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s} \\ & \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \\ & \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \leq 0 \\ & \mathbf{h}_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \leq 0 \end{aligned}$$

- MPC delivers a parametric deterministic policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$

MPC as a Function Approximator

MPC Tuning parameter $\boldsymbol{\theta}$, initial state $\mathbf{s}$

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s}) = \mathbf{u}_0^\star$$

$$\begin{aligned} V_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\mathbf{s}) = \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} \ell_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{u}) + V_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \\ \text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s} \\ & \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \\ & \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \leq 0 \\ & \mathbf{h}_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \leq 0 \end{aligned}$$

- MPC delivers a parametric deterministic policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$
- MPC delivers a parametric value function $V_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}$

MPC as a Function Approximator

MPC Tuning parameter $\boldsymbol{\theta}$, initial state s, fixed policy $\mathbf{a}$

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s}) = \mathbf{u}_0^\star \qquad Q_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\mathrm{s},\mathbf{a}) = \begin{aligned}[t] \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} \ell_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{u}) + V_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \\ \mathrm{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \quad \mathbf{u}_0 = \mathbf{a} \\ & \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k) \\ & \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k) \leq 0 \\ & \mathbf{h}_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \leq 0 \end{aligned}$$

- MPC delivers a parametric deterministic policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$
- MPC delivers a parametric value function $V_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}$
- MPC delivers a parametric action value function $Q_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}$

MPC as a Function Approximator

MPC Tuning parameter $\boldsymbol{\theta}$, initial state s, fixed policy $\mathbf{a}$

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathrm{s}) = \mathbf{u}_0^\star \qquad\qquad \begin{aligned} Q_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\mathrm{s}, \mathbf{a}) = \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} \ell_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) + V_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \\ \mathrm{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \quad \mathbf{u}_0 = \mathbf{a} \\ & \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \\ & \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \leq 0 \\ & \mathbf{h}_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \leq 0 \end{aligned}$$

- MPC delivers a parametric deterministic policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$
- MPC delivers a parametric value function $V_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}$
- MPC delivers a parametric action value function $Q_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}$
    - In practice be careful with $\mathbf{u}_0 = \mathbf{a}$: LICQ issues

MPC as a Function Approximator

MPC Tuning parameter $\boldsymbol{\theta}$, initial state $\mathbf{s}$, fixed policy $\mathbf{a}$

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s}) = \mathbf{u}_0^\star \qquad\qquad Q_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\mathbf{s},\mathbf{a}) = \min_{\mathbf{x},\mathbf{u}} \ \sum_{k=0}^{N-1} \ell_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{u}) + V_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N)$$

$$\begin{aligned} \text{s.t.}\quad & \mathbf{x}_0 = \mathbf{s}, \quad \mathbf{u}_0 = \mathbf{a} \\ & \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \\ & \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \leq 0 \\ & \mathbf{h}_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \leq 0 \end{aligned}$$

- MPC delivers a parametric deterministic policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$
- MPC delivers a parametric value function $V_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}$
- MPC delivers a parametric action value function $Q_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}$
  - In practice be careful with $\mathbf{u}_0 = \mathbf{a}$: LICQ issues
- **All these functions can be differentiated wrt $\boldsymbol{\theta}$**
  - derivatives necessary in RL algorithms
  - need LICQ + few other conditions: mild requirements
  - if we can solve MPC we can differentiate it

MPC as a Function Approximator

MPC Tuning parameter $\boldsymbol{\theta}$, initial state $\mathbf{s}$, fixed policy $\mathbf{a}$

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s}) = \mathbf{u}_0^\star \qquad Q_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\mathbf{s},\mathbf{a}) = \min_{\mathbf{x},\mathbf{u}} \sum_{k=0}^{N-1} \ell_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{u}) + V_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N)$$

$$\begin{aligned} \text{s.t. } & \mathbf{x}_0 = \mathbf{s}, \quad \mathbf{u}_0 = \mathbf{a} \\ & \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k) \\ & \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k) \leq 0 \\ & \mathbf{h}_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \leq 0 \end{aligned}$$

- MPC delivers a parametric deterministic policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$
- MPC delivers a parametric value function $V_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}$
- MPC delivers a parametric action value function $Q_{\boldsymbol{\theta}}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}$
  - In practice be careful with $\mathbf{u}_0 = \mathbf{a}$: LICQ issues
- **All these functions can be differentiated wrt $\boldsymbol{\theta}$**
  - derivatives necessary in RL algorithms
  - need LICQ + few other conditions: mild requirements
  - if we can solve MPC we can differentiate it

MPC has all the properties required from a function approximator

- We can let RL learn the best $\boldsymbol{\theta}$
- We can let MPC enforce stability and safety guarantees in RL

MPC is a parametric NLP

MPC:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} \ell_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) + V_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \\
\text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s} \\
& \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \\
& \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \leq 0 \\
& \mathbf{h}_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \leq 0
\end{aligned}
$$

MPC is a parametric NLP

MPC:

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} \ell_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) + V_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \\
\text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s} \\
& \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \\
& \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \leq 0 \\
& \mathbf{h}_{\boldsymbol{\theta}}^{\mathrm{f}}(\mathbf{x}_N) \leq 0
\end{aligned}$$

Parametric NLP:

$$\begin{aligned}
\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad & \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \\
\text{s.t.} \quad & \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0 \\
& \bar{\mathbf{h}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \leq 0
\end{aligned}$$

where

$$\begin{aligned}
\bar{\mathbf{x}} &= (\mathbf{x}, \mathbf{u}) \\
\bar{\mathbf{p}} &= \boldsymbol{\theta}
\end{aligned}$$

MPC is a parametric NLP

MPC:

$$\begin{aligned}\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} \ell_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) + V^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N) \\ \text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s} \\ & \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \\ & \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \leq 0 \\ & \mathbf{h}^{\mathrm{f}}_{\boldsymbol{\theta}}(\mathbf{x}_N) \leq 0\end{aligned}$$

Parametric NLP:

$$\begin{aligned}\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad & \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \\ \text{s.t.} \quad & \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0 \\ & \bar{\mathbf{h}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \leq 0\end{aligned}$$

where

$$\begin{aligned}\bar{\mathbf{x}} &= (\mathbf{x}, \mathbf{u}) \\ \bar{\mathbf{p}} &= \boldsymbol{\theta}\end{aligned}$$

Two questions require a solid answer:

- How do we solve NLPs?
- How do we differentiate NLPs?

How Do We Solve Parametric NLPs?

NLP

$$\begin{aligned} \min_{\bar{\mathbf{x}}} \quad & \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \\ \text{s.t.} \quad & \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0 \\ & \bar{\mathbf{h}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \leq 0 \end{aligned}$$

Quadratic approximation

$$\begin{aligned} \min_{\Delta\bar{\mathbf{x}}} \quad & \Delta\bar{\mathbf{x}}^\top \boldsymbol{M} \Delta\bar{\mathbf{x}} + \boldsymbol{m}^\top \Delta\bar{\mathbf{x}} \\ \text{s.t.} \quad & \boldsymbol{G}\Delta\bar{\mathbf{x}} + \bar{\boldsymbol{g}} = 0 \\ & \boldsymbol{H}\Delta\bar{\mathbf{x}} + \bar{\boldsymbol{h}} \leq 0 \end{aligned}$$

Iterative procedure:

- Given $\bar{\mathbf{x}}$
- Compute quadratic approximation
- Enforce $\boldsymbol{M} \succ 0$
- Solve QP / linear system
- Ensure progress by computing α
- Take a step $\bar{\mathbf{x}} = \bar{\mathbf{x}} + \alpha\Delta\bar{\mathbf{x}}$

How Do We Solve Parametric NLPs?

NLP

$$\begin{aligned}\min_{\bar{\mathbf{x}}} \quad & \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \\ \text{s.t.} \quad & \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0 \\ & \bar{\mathbf{h}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \leq 0\end{aligned}$$

Quadratic approximation

$$\begin{aligned}\min_{\Delta\bar{\mathbf{x}}} \quad & \Delta\bar{\mathbf{x}}^\top \boldsymbol{M} \Delta\bar{\mathbf{x}} + \boldsymbol{m}^\top \Delta\bar{\mathbf{x}} \\ \text{s.t.} \quad & \boldsymbol{G}\Delta\bar{\mathbf{x}} + \bar{\boldsymbol{g}} = 0 \\ & \boldsymbol{H}\Delta\bar{\mathbf{x}} + \bar{\boldsymbol{h}} \leq 0\end{aligned}$$

Iterative procedure:

- Given $\bar{\mathbf{x}}$
- Compute quadratic approximation
- Enforce $\boldsymbol{M} \succ 0$
- Solve QP / linear system
- Ensure progress by computing α
- Take a step $\bar{\mathbf{x}} = \bar{\mathbf{x}} + \alpha\Delta\bar{\mathbf{x}}$

Primal-dual solution

$$\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}) := \left(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\boldsymbol{\lambda}}^\star(\bar{\mathbf{p}}), \bar{\boldsymbol{\mu}}^\star(\bar{\mathbf{p}})\right)$$

satisfies the KKT conditions

$$\mathbf{r}(\bar{\mathbf{z}}, \bar{\mathbf{p}}) := \begin{bmatrix} \nabla_{\bar{\mathbf{x}}}\mathcal{L}(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}}, \bar{\mathbf{p}}) \\ \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \\ \bar{\mathbf{h}}_{\mathbb{A}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \end{bmatrix} = 0$$

$$\bar{\boldsymbol{\mu}}^\star_{\mathbb{A}^c}(\bar{\mathbf{p}}) = 0$$

with active set $\mathbb{A}$ and

$$\mathcal{L}(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}}, \bar{\mathbf{p}}) := \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) + \bar{\boldsymbol{\lambda}}^\top \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) + \bar{\boldsymbol{\mu}}^\top \bar{\mathbf{h}}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$

Parametric NLP Sensitivities

Parametric NLP:

$$\begin{aligned} \bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad & \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \\ \text{s.t.} \quad & \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0 \end{aligned}$$

Parametric NLP Sensitivities

Parametric NLP:

$$\begin{aligned}\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad & \bar{f}(\bar{\mathbf{x}},\bar{\mathbf{p}}) \\ \text{s.t.} \quad & \bar{\mathbf{g}}(\bar{\mathbf{x}},\bar{\mathbf{p}}) = 0\end{aligned}$$

KKT Conditions

$$r(\bar{\mathbf{z}},\bar{\mathbf{p}}) := \begin{bmatrix} \nabla_{\bar{\mathbf{x}}}\mathcal{L}(\bar{\mathbf{x}},\bar{\boldsymbol{\lambda}},\bar{\mathbf{p}}) \\ \bar{\mathbf{g}}(\bar{\mathbf{x}},\bar{\mathbf{p}}) \end{bmatrix} = 0$$

Parametric NLP Sensitivities

Parametric NLP:

$$\begin{aligned} \bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad & \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \\ \text{s.t.} \quad & \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0 \end{aligned}$$

KKT Conditions

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) := \begin{bmatrix} \nabla_{\bar{\mathbf{x}}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}, \bar{\mathbf{p}}) \\ \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \end{bmatrix} = 0$$

Implicit Function Theorem

Let $\bar{\mathbf{z}}$ be implicitly given by the (at least) $\mathcal{C}^1$ function

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) = 0 \quad \text{with} \quad \nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right) \quad \text{full rank.}$$

Then $\exists$ a $\mathcal{C}^1$ function $\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})$ such that

$$r(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}) = 0$$

holds in a neighbourhood of $\bar{\mathbf{p}}$.
Moreover:

$$\nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)^\top \frac{\mathrm{d}\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})}{\mathrm{d}\bar{\mathbf{p}}} = -\frac{\partial}{\partial \bar{\mathbf{p}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)$$

Parametric NLP Sensitivities

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

KKT Conditions

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) := \begin{bmatrix} \nabla_{\bar{\mathbf{x}}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}, \bar{\mathbf{p}}) \\ \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \end{bmatrix} = 0$$

Implicit Function Theorem

Let $\bar{\mathbf{z}}$ be implicitly given by the (at least) $\mathcal{C}^1$ function

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) = 0 \quad \text{with} \quad \nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right) \quad \text{full rank.}$$

Then $\exists$ a $\mathcal{C}^1$ function $\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})$ such that

$$r(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}) = 0$$

holds in a neighbourhood of $\bar{\mathbf{p}}$.
Moreover:

$$\nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)^\top \frac{\mathrm{d}\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})}{\mathrm{d}\bar{\mathbf{p}}} = -\frac{\partial}{\partial \bar{\mathbf{p}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)$$

Is this enough?

Parametric NLP Sensitivities

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

KKT Conditions

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) := \begin{bmatrix} \nabla_{\bar{\mathbf{x}}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}, \bar{\mathbf{p}}) \\ \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \end{bmatrix} = 0$$

Implicit Function Theorem

Let $\bar{\mathbf{z}}$ be implicitly given by the (at least) $\mathcal{C}^1$ function

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) = 0 \quad \text{with} \quad \nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right) \quad \text{full rank.}$$

Then $\exists$ a $\mathcal{C}^1$ function $\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})$ such that

$$r(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}) = 0$$

holds in a neighbourhood of $\bar{\mathbf{p}}$.
Moreover:

$$\nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)^\top \frac{\mathrm{d}\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})}{\mathrm{d}\bar{\mathbf{p}}} = -\frac{\partial}{\partial \bar{\mathbf{p}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)$$

Is this enough?

Not all KKT points are minima!

Parametric NLP Sensitivities

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

KKT Conditions

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) := \begin{bmatrix} \nabla_{\bar{\mathbf{x}}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}, \bar{\mathbf{p}}) \\ \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \end{bmatrix} = 0$$

Implicit Function Theorem

Let $\bar{\mathbf{z}}$ be implicitly given by the (at least) $\mathcal{C}^1$ function

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) = 0 \quad \text{with} \quad \nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right) \quad \text{full rank.}$$

Then $\exists$ a $\mathcal{C}^1$ function $\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})$ such that

$$r(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}) = 0$$

holds in a neighbourhood of $\bar{\mathbf{p}}$.
Moreover:

$$\nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)^\top \frac{\mathrm{d}\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})}{\mathrm{d}\bar{\mathbf{p}}} = -\frac{\partial}{\partial \bar{\mathbf{p}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)$$

Is this enough?

Not all KKT points are minima!

We need:

- LICQ
- SOSC

Parametric NLP Sensitivities

Parametric NLP:

$$\begin{aligned}\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad & \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \\ \text{s.t.} \quad & \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0\end{aligned}$$

KKT Conditions

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) := \begin{bmatrix} \nabla_{\bar{\mathbf{x}}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}, \bar{\mathbf{p}}) \\ \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \end{bmatrix} = 0$$

Implicit Function Theorem

Let $\bar{\mathbf{z}}$ be implicitly given by the (at least) $\mathcal{C}^1$ function

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) = 0 \quad \text{with} \quad \nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right) \quad \text{full rank.}$$

Then $\exists$ a $\mathcal{C}^1$ function $\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})$ such that

$$r(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}) = 0$$

holds in a neighbourhood of $\bar{\mathbf{p}}$.
Moreover:

$$\nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)^\top \frac{\mathrm{d}\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})}{\mathrm{d}\bar{\mathbf{p}}} = -\frac{\partial}{\partial \bar{\mathbf{p}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)$$

Is this enough?

Not all KKT points are minima!

We need:

- LICQ
- SOSC

Then $\nabla_{\bar{\mathbf{z}}} r$ invertible!

Parametric NLP Sensitivities

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

KKT Conditions

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) := \begin{bmatrix} \nabla_{\bar{\mathbf{x}}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}, \bar{\mathbf{p}}) \\ \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \end{bmatrix} = 0$$

How to Compute the Derivative

$$\nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)^\top \frac{\mathrm{d}\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})}{\mathrm{d}\bar{\mathbf{p}}} = -\frac{\partial}{\partial \bar{\mathbf{p}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)$$

Is this enough?

Not all KKT points are minima!

We need:

- LICQ
- SOSC

Then $\nabla_{\bar{\mathbf{z}}} r$ invertible!

Parametric NLP Sensitivities

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

KKT Conditions

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) := \begin{bmatrix} \nabla_{\bar{\mathbf{x}}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}, \bar{\mathbf{p}}) \\ \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \end{bmatrix} = 0$$

How to Compute the Derivative

$$\nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)^\top \frac{\mathrm{d}\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})}{\mathrm{d}\bar{\mathbf{p}}} = -\frac{\partial}{\partial \bar{\mathbf{p}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)$$

Is this enough?

Not all KKT points are minima!

We need:

- LICQ
- SOSC

Then $\nabla_{\bar{\mathbf{z}}} r$ invertible!

Shorthand:

$$\mathcal{L}_{\bar{\mathbf{p}}} := \mathcal{L}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\boldsymbol{\lambda}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$
$$\bar{\mathbf{g}}_{\bar{\mathbf{p}}} := \bar{\mathbf{g}}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$

Parametric NLP Sensitivities

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

KKT Conditions

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) := \begin{bmatrix} \nabla_{\bar{\mathbf{x}}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}, \bar{\mathbf{p}}) \\ \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \end{bmatrix} = 0$$

How to Compute the Derivative

$$\nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)^\top \frac{\mathrm{d}\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})}{\mathrm{d}\bar{\mathbf{p}}} = -\frac{\partial}{\partial \bar{\mathbf{p}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)$$

Observe that

$$\nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)^\top = \begin{bmatrix} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}} \mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}} \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}} \bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{bmatrix}$$

Is this enough?

Not all KKT points are minima!

We need:

- LICQ
- SOSC

Then $\nabla_{\bar{\mathbf{z}}} r$ invertible!

Shorthand:

$$\mathcal{L}_{\bar{\mathbf{p}}} := \mathcal{L}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\boldsymbol{\lambda}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$
$$\bar{\mathbf{g}}_{\bar{\mathbf{p}}} := \bar{\mathbf{g}}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$

Parametric NLP Sensitivities

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \qquad \text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

KKT Conditions

$$r(\bar{\mathbf{z}}, \bar{\mathbf{p}}) := \begin{bmatrix} \nabla_{\bar{\mathbf{x}}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}, \bar{\mathbf{p}}) \\ \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \end{bmatrix} = 0$$

How to Compute the Derivative

$$\nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)^\top \frac{\mathrm{d}\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})}{\mathrm{d}\bar{\mathbf{p}}} = -\frac{\partial}{\partial \bar{\mathbf{p}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)$$

Observe that

$$\nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}}\right)^\top = \begin{bmatrix} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}} \mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}} \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}} \bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{bmatrix}$$

Then

$$\begin{bmatrix} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}} \mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}} \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}} \bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{bmatrix} \frac{\mathrm{d}\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})}{\mathrm{d}\bar{\mathbf{p}}} = -\frac{\partial}{\partial \bar{\mathbf{p}}} \begin{bmatrix} \nabla_{\bar{\mathbf{x}}} \mathcal{L}_{\bar{\mathbf{p}}} \\ \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \end{bmatrix}$$

Is this enough?

Not all KKT points are minima!

We need:

- LICQ
- SOSC

Then $\nabla_{\bar{\mathbf{z}}} r$ invertible!

Shorthand:

$$\mathcal{L}_{\bar{\mathbf{p}}} := \mathcal{L}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\boldsymbol{\lambda}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$

$$\bar{\mathbf{g}}_{\bar{\mathbf{p}}} := \bar{\mathbf{g}}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$

Parametric NLP Sensitivities

Parametric NLP:

$$\begin{aligned}\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad & \bar{f}(\bar{\mathbf{x}},\bar{\mathbf{p}}) \\ \text{s.t.} \quad & \bar{\mathbf{g}}(\bar{\mathbf{x}},\bar{\mathbf{p}}) = 0\end{aligned}$$

KKT Conditions

$$r(\bar{\mathbf{z}},\bar{\mathbf{p}}) := \left[\begin{array}{c} \nabla_{\bar{\mathbf{x}}}\mathcal{L}(\bar{\mathbf{x}},\bar{\boldsymbol{\lambda}},\bar{\mathbf{p}}) \\ \bar{\mathbf{g}}(\bar{\mathbf{x}},\bar{\mathbf{p}}) \end{array}\right] = 0$$

How to Compute the Derivative

$$\nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}),\bar{\mathbf{p}}\right)^\top \frac{\mathrm{d}\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})}{\mathrm{d}\bar{\mathbf{p}}} = -\frac{\partial}{\partial\bar{\mathbf{p}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}),\bar{\mathbf{p}}\right)$$

Observe that

$$\nabla_{\bar{\mathbf{z}}} r\left(\bar{\mathbf{z}}^\star(\bar{\mathbf{p}}),\bar{\mathbf{p}}\right)^\top = \left[\begin{array}{cc} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{array}\right]$$

Then

$$\left[\begin{array}{cc} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{array}\right] \frac{\mathrm{d}\bar{\mathbf{z}}^\star(\bar{\mathbf{p}})}{\mathrm{d}\bar{\mathbf{p}}} = -\frac{\partial}{\partial\bar{\mathbf{p}}}\left[\begin{array}{c} \nabla_{\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} \\ \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \end{array}\right]$$

Compare with the (last) Newton step!

$$\left[\begin{array}{cc} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{array}\right]\left[\begin{array}{c} \Delta\bar{\mathbf{x}} \\ \bar{\boldsymbol{\lambda}} \end{array}\right] = -\left[\begin{array}{c} \nabla_{\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} \\ \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \end{array}\right]$$

Is this enough?

Not all KKT points are minima!

We need:

- LICQ
- SOSC

Then $\nabla_{\bar{\mathbf{z}}} r$ invertible!

Shorthand:

$$\mathcal{L}_{\bar{\mathbf{p}}} := \mathcal{L}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}),\bar{\boldsymbol{\lambda}}^\star(\bar{\mathbf{p}}),\bar{\mathbf{p}})$$
$$\bar{\mathbf{g}}_{\bar{\mathbf{p}}} := \bar{\mathbf{g}}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}),\bar{\mathbf{p}})$$

Sensitivities and Implicit Functions

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}},\bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}},\bar{\mathbf{p}}) = 0$$

$$f^\star(\bar{\mathbf{p}}) = \bar{f}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}),\bar{\mathbf{p}})$$

Parametric Sensitivities:

$$\begin{bmatrix} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{bmatrix} \frac{\mathbf{d}}{\mathbf{d}\bar{\mathbf{p}}} \begin{bmatrix} \bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) \\ \bar{\lambda}^\star(\bar{\mathbf{p}}) \end{bmatrix} = -\frac{\partial}{\partial \bar{\mathbf{p}}} \begin{bmatrix} \nabla_{\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} \\ \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \end{bmatrix}$$

Sensitivities and Implicit Functions

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}},\bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}},\bar{\mathbf{p}}) = 0$$

$$f^\star(\bar{\mathbf{p}}) = \bar{f}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}),\bar{\mathbf{p}})$$

Important properties:

Parametric Sensitivities:

$$\begin{bmatrix} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{bmatrix} \frac{\mathbf{d}}{\mathbf{d}\bar{\mathbf{p}}} \begin{bmatrix} \bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) \\ \bar{\lambda}^\star(\bar{\mathbf{p}}) \end{bmatrix} = -\frac{\partial}{\partial\bar{\mathbf{p}}} \begin{bmatrix} \nabla_{\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} \\ \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \end{bmatrix}$$

Sensitivities and Implicit Functions

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

$$f^\star(\bar{\mathbf{p}}) = \bar{f}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$

Parametric Sensitivities:

$$\begin{bmatrix} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{bmatrix} \frac{\mathbf{d}}{\mathbf{d}\bar{\mathbf{p}}} \begin{bmatrix} \bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) \\ \bar{\lambda}^\star(\bar{\mathbf{p}}) \end{bmatrix} = -\frac{\partial}{\partial \bar{\mathbf{p}}} \begin{bmatrix} \nabla_{\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} \\ \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \end{bmatrix}$$

Important properties:

- Sensitivities are (almost) for free: KKT matrix **already factorized**!

Sensitivities and Implicit Functions

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

$$f^\star(\bar{\mathbf{p}}) = \bar{f}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$

Parametric Sensitivities:

$$\begin{bmatrix} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{bmatrix} \frac{\mathbf{d}}{\mathbf{d}\bar{\mathbf{p}}} \begin{bmatrix} \bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) \\ \bar{\lambda}^\star(\bar{\mathbf{p}}) \end{bmatrix} = -\frac{\partial}{\partial \bar{\mathbf{p}}} \begin{bmatrix} \nabla_{\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} \\ \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \end{bmatrix}$$

Important properties:

- Sensitivities are (almost) for free: KKT matrix **already factorized**!
- Sensitivity of the optimal value:

$$\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}} = \frac{\partial \mathcal{L}_{\bar{\mathbf{p}}}}{\partial \bar{\mathbf{p}}}$$

Sensitivities and Implicit Functions

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

$$f^\star(\bar{\mathbf{p}}) = \bar{f}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$

Parametric Sensitivities:

$$\begin{bmatrix} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{bmatrix} \frac{\mathbf{d}}{\mathbf{d}\bar{\mathbf{p}}} \begin{bmatrix} \bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) \\ \bar{\lambda}^\star(\bar{\mathbf{p}}) \end{bmatrix} = -\frac{\partial}{\partial \bar{\mathbf{p}}} \begin{bmatrix} \nabla_{\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} \\ \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \end{bmatrix}$$

Important properties:

- Sensitivities are (almost) for free: KKT matrix **already factorized**!
- Sensitivity of the optimal value:

$$\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}} = \frac{\partial \mathcal{L}_{\bar{\mathbf{p}}}}{\partial \bar{\mathbf{p}}}$$

What does this mean for MPC?

Sensitivities and Implicit Functions

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

$$f^\star(\bar{\mathbf{p}}) = \bar{f}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$

Parametric Sensitivities:

$$\begin{bmatrix} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}} \mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}} \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}} \bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{bmatrix} \frac{\mathbf{d}}{\mathbf{d}\bar{\mathbf{p}}} \begin{bmatrix} \bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) \\ \bar{\lambda}^\star(\bar{\mathbf{p}}) \end{bmatrix} = -\frac{\partial}{\partial \bar{\mathbf{p}}} \begin{bmatrix} \nabla_{\bar{\mathbf{x}}} \mathcal{L}_{\bar{\mathbf{p}}} \\ \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \end{bmatrix}$$

Important properties:

- Sensitivities are (almost) for free: KKT matrix **already factorized**!
- Sensitivity of the optimal value:

$$\frac{\mathbf{d} f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}} = \frac{\partial \mathcal{L}_{\bar{\mathbf{p}}}}{\partial \bar{\mathbf{p}}}$$

What does this mean for MPC?

- $\frac{\mathrm{d} V_\theta(s)}{\mathrm{d}\theta}$ from $\frac{\mathbf{d} f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}}$

Sensitivities and Implicit Functions

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) \quad \text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

$$f^\star(\bar{\mathbf{p}}) = \bar{f}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$

Parametric Sensitivities:

$$\begin{bmatrix} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{bmatrix} \frac{\mathbf{d}}{\mathbf{d}\bar{\mathbf{p}}} \begin{bmatrix} \bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) \\ \bar{\lambda}^\star(\bar{\mathbf{p}}) \end{bmatrix} = -\frac{\partial}{\partial \bar{\mathbf{p}}} \begin{bmatrix} \nabla_{\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} \\ \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \end{bmatrix}$$

Important properties:

- Sensitivities are (almost) for free: KKT matrix **already factorized**!
- Sensitivity of the optimal value:

$$\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}} = \frac{\partial \mathcal{L}_{\bar{\mathbf{p}}}}{\partial \bar{\mathbf{p}}}$$

What does this mean for MPC?

- $\frac{\mathrm{d}V_\theta(s)}{\mathrm{d}\theta}$ from $\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}}$
- $\frac{\mathrm{d}Q_\theta(s,a)}{\mathrm{d}\theta}$ from $\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}}$

Sensitivities and Implicit Functions

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

$$f^\star(\bar{\mathbf{p}}) = \bar{f}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$

Parametric Sensitivities:

$$\begin{bmatrix} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{bmatrix} \frac{\mathbf{d}}{\mathbf{d}\bar{\mathbf{p}}} \begin{bmatrix} \bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) \\ \bar{\lambda}^\star(\bar{\mathbf{p}}) \end{bmatrix} = -\frac{\partial}{\partial \bar{\mathbf{p}}} \begin{bmatrix} \nabla_{\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} \\ \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \end{bmatrix}$$

Important properties:

- Sensitivities are (almost) for free: KKT matrix **already factorized**!
- Sensitivity of the optimal value:

$$\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}} = \frac{\partial \mathcal{L}_{\bar{\mathbf{p}}}}{\partial \bar{\mathbf{p}}}$$

What does this mean for MPC?

- $\frac{\mathrm{d}V_\theta(s)}{\mathrm{d}\theta}$ from $\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}}$
- $\frac{\mathrm{d}Q_\theta(s,a)}{\mathrm{d}\theta}$ from $\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}}$
- $\frac{\mathrm{d}\boldsymbol{\pi}_\theta(s)}{\mathrm{d}\theta}$ from $\frac{\mathbf{d}\bar{\mathbf{x}}^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}}$

Sensitivities and Implicit Functions

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

$$f^\star(\bar{\mathbf{p}}) = \bar{f}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$

Parametric Sensitivities:

$$\begin{bmatrix} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{bmatrix} \frac{\mathbf{d}}{\mathbf{d}\bar{\mathbf{p}}} \begin{bmatrix} \bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) \\ \bar{\lambda}^\star(\bar{\mathbf{p}}) \end{bmatrix} = -\frac{\partial}{\partial \bar{\mathbf{p}}} \begin{bmatrix} \nabla_{\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} \\ \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \end{bmatrix}$$

Important properties:

- Sensitivities are (almost) for free: KKT matrix **already factorized**!
- Sensitivity of the optimal value:

$$\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}} = \frac{\partial \mathcal{L}_{\bar{\mathbf{p}}}}{\partial \bar{\mathbf{p}}}$$

Sensitivities are cheap and easy to compute

What does this mean for MPC?

- $\frac{\mathrm{d}V_\theta(s)}{\mathrm{d}\theta}$ from $\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}}$
- $\frac{\mathrm{d}Q_\theta(s,a)}{\mathrm{d}\theta}$ from $\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}}$
- $\frac{\mathrm{d}\boldsymbol{\pi}_\theta(s)}{\mathrm{d}\theta}$ from $\frac{\mathbf{d}\bar{\mathbf{x}}^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}}$

Sensitivities and Implicit Functions

Parametric NLP:

$$\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) := \arg\min_{\bar{\mathbf{x}}} \quad \bar{f}(\bar{\mathbf{x}}, \bar{\mathbf{p}})$$
$$\text{s.t.} \quad \bar{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{p}}) = 0$$

$$f^\star(\bar{\mathbf{p}}) = \bar{f}(\bar{\mathbf{x}}^\star(\bar{\mathbf{p}}), \bar{\mathbf{p}})$$

Parametric Sensitivities:

$$\begin{bmatrix} \nabla^2_{\bar{\mathbf{x}}\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} & \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}} \\ \nabla_{\bar{\mathbf{x}}}\bar{\mathbf{g}}_{\bar{\mathbf{p}}}^\top & 0 \end{bmatrix} \frac{\mathbf{d}}{\mathbf{d}\bar{\mathbf{p}}} \begin{bmatrix} \bar{\mathbf{x}}^\star(\bar{\mathbf{p}}) \\ \bar{\lambda}^\star(\bar{\mathbf{p}}) \end{bmatrix} = -\frac{\partial}{\partial \bar{\mathbf{p}}} \begin{bmatrix} \nabla_{\bar{\mathbf{x}}}\mathcal{L}_{\bar{\mathbf{p}}} \\ \bar{\mathbf{g}}_{\bar{\mathbf{p}}} \end{bmatrix}$$

Important properties:

- Sensitivities are (almost) for free: KKT matrix **already factorized**!
- Sensitivity of the optimal value:

$$\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}} = \frac{\partial \mathcal{L}_{\bar{\mathbf{p}}}}{\partial \bar{\mathbf{p}}}$$

Sensitivities are cheap and easy to compute

What does this mean for MPC?

- $\frac{\mathrm{d}V_\theta(s)}{\mathrm{d}\theta}$ from $\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}}$
- $\frac{\mathrm{d}Q_\theta(s,a)}{\mathrm{d}\theta}$ from $\frac{\mathbf{d}f^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}}$
- $\frac{\mathrm{d}\boldsymbol{\pi}_\theta(s)}{\mathrm{d}\theta}$ from $\frac{\mathbf{d}\bar{\mathbf{x}}^\star(\bar{\mathbf{p}})}{\mathbf{d}\bar{\mathbf{p}}}$

What about inequality constraints?

Not a real issue in practice,
see parametric NLP theory

Safe and Stabilizing RL Based on Linear Tube MPC

Tube-Based Robust MPC

$$
\begin{aligned}
\hat{Q}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}) :=& \\
\min_{\mathbf{z}} \quad & \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 \\
& + \left\| \mathbf{x}_0 \right\|_{\Lambda}^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l \\
\mathrm{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\
& \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathbf{b}, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathbf{t}(\boldsymbol{\theta}) \leq 0,
\end{aligned}
$$

Parameter vector: $\boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$

Safe and Stabilizing RL Based on Linear Tube MPC

Tube-Based Robust MPC

$$\hat{Q}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}) :=$$

$$\begin{aligned}
\min_{\mathbf{z}} \quad & \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 \\
& + \left\| \mathbf{x}_0 \right\|_{\Lambda}^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l \\
\text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\
& \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathbf{b}, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathbf{t}(\boldsymbol{\theta}) \leq 0,
\end{aligned}$$

Parameter vector: $\boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$

Conditions to enforce on $\boldsymbol{\theta}$:

Safe and Stabilizing RL Based on Linear Tube MPC

Tube-Based Robust MPC

$$\hat{Q}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}) :=$$

$$\begin{aligned}
\min_{\mathbf{z}} \quad & \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 \\
& + \left\| \mathbf{x}_0 \right\|_\Lambda^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l \\
\text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\
& \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathbf{b}, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathbf{t}(\boldsymbol{\theta}) \leq 0,
\end{aligned}$$

Parameter vector: $\boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$

Conditions to enforce on $\boldsymbol{\theta}$:

- Steady state

$$\mathbf{x}_\mathrm{r} = \boldsymbol{A}\mathbf{x}_\mathrm{r} + \boldsymbol{B}\mathbf{u}_\mathrm{r}$$

Safe and Stabilizing RL Based on Linear Tube MPC

Tube-Based Robust MPC

$$\hat{Q}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}) :=$$

$$\begin{aligned}
\min_{\mathbf{z}} \quad & \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 \\
& + \left\| \mathbf{x}_0 \right\|_{\Lambda}^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l \\
\text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\
& \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathbf{b}, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathbf{t}(\boldsymbol{\theta}) \leq 0,
\end{aligned}$$

Parameter vector: $\boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$

Conditions to enforce on $\boldsymbol{\theta}$:

- Steady state

$$\mathbf{x}_\mathrm{r} = \boldsymbol{A}\mathbf{x}_\mathrm{r} + \boldsymbol{B}\mathbf{u}_\mathrm{r}$$

- Positive definiteness

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{S} \\ \boldsymbol{S}^\top & \boldsymbol{R} \end{bmatrix} \succ 0$$

Safe and Stabilizing RL Based on Linear Tube MPC

Tube-Based Robust MPC

$$\hat{Q}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}) :=$$

$$\begin{aligned}
\min_{\mathbf{z}} \quad & \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 \\
& + \left\| \mathbf{x}_0 \right\|_{\Lambda}^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l \\
\text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\
& \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathbf{b}, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathbf{t}(\boldsymbol{\theta}) \leq 0,
\end{aligned}$$

Parameter vector: $\boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$

Conditions to enforce on $\boldsymbol{\theta}$:

- Steady state

$$\mathbf{x}_\mathrm{r} = \boldsymbol{A}\mathbf{x}_\mathrm{r} + \boldsymbol{B}\mathbf{u}_\mathrm{r}$$

- Positive definiteness

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{S} \\ \boldsymbol{S}^\top & \boldsymbol{R} \end{bmatrix} \succ 0$$

- Consistent uncertainty set

$$\boldsymbol{M}(\mathbf{s}_{i+1} - (\boldsymbol{A}\mathbf{s}_i + \boldsymbol{B}\mathbf{a}_i + \mathbf{b})) \leq \mathbf{m}$$

Safe and Stabilizing RL Based on Linear Tube MPC

Tube-Based Robust MPC

$$\hat{Q}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}) :=$$

$$\begin{aligned} \min_{\mathbf{z}} \quad & \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 \\ & + \left\| \mathbf{x}_0 \right\|_{\Lambda}^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l \\ \text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\ & \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathbf{b}, \qquad k \in \mathbb{I}_0^{N-1}, \\ & \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \\ & \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathbf{t}(\boldsymbol{\theta}) \leq 0, \end{aligned}$$

Parameter vector: $\quad \boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$

Conditions to enforce on $\boldsymbol{\theta}$:

- Steady state

$$\mathbf{x}_\mathrm{r} = \boldsymbol{A}\mathbf{x}_\mathrm{r} + \boldsymbol{B}\mathbf{u}_\mathrm{r}$$

- Positive definiteness

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{S} \\ \boldsymbol{S}^\top & \boldsymbol{R} \end{bmatrix} \succ 0$$

- Consistent uncertainty set

$$\boldsymbol{M}(\mathbf{s}_{i+1} - (\boldsymbol{A}\mathbf{s}_i + \boldsymbol{B}\mathbf{a}_i + \mathbf{b})) \leq \mathbf{m}$$

- Terminal set includes the reference

$$\boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_\mathrm{r} \leq \mathbf{t}(\boldsymbol{\theta})$$

Safe and Stabilizing RL Based on Linear Tube MPC

Tube-Based Robust MPC

$$\hat{Q}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}) :=$$

$$\begin{aligned}\min_{\mathbf{z}} \quad & \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 \\ & + \left\| \mathbf{x}_0 \right\|_\Lambda^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l \\ \text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\ & \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathbf{b}, \qquad k \in \mathbb{I}_0^{N-1}, \\ & \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \le 0, \qquad k \in \mathbb{I}_0^{N-1}, \\ & \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathbf{t}(\boldsymbol{\theta}) \le 0,\end{aligned}$$

Parameter vector: $\boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$

Conditions to enforce on $\boldsymbol{\theta}$:

- Steady state

$$\mathbf{x}_\mathrm{r} = \boldsymbol{A}\mathbf{x}_\mathrm{r} + \boldsymbol{B}\mathbf{u}_\mathrm{r}$$

- Positive definiteness

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{S} \\ \boldsymbol{S}^\top & \boldsymbol{R} \end{bmatrix} \succ 0$$

- Consistent uncertainty set

$$\boldsymbol{M}(\mathbf{s}_{i+1} - (\boldsymbol{A}\mathbf{s}_i + \boldsymbol{B}\mathbf{a}_i + \mathbf{b})) \le \mathbf{m}$$

- Terminal set includes the reference

$$\boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_\mathrm{r} \le \mathbf{t}(\boldsymbol{\theta})$$

Moreover:

- Riccati terminal cost and control law

$$\boldsymbol{K}(\boldsymbol{\theta}) = (\boldsymbol{R} + \boldsymbol{B}^\top \boldsymbol{P}(\boldsymbol{\theta})\boldsymbol{B})^{-1}(\boldsymbol{S}^\top + \boldsymbol{B}^\top \boldsymbol{P}(\boldsymbol{\theta})\boldsymbol{A})$$

$$\boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{Q} + \boldsymbol{A}^\top \boldsymbol{P}(\boldsymbol{\theta})\boldsymbol{A} - (\boldsymbol{S} + \boldsymbol{A}^\top \boldsymbol{P}(\boldsymbol{\theta})\boldsymbol{B})\boldsymbol{K}(\boldsymbol{\theta})$$

Safe and Stabilizing RL Based on Linear Tube MPC

Tube-Based Robust MPC

$$\hat{Q}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}) :=$$

$$\begin{aligned}
\min_{\mathbf{z}} \quad & \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 \\
& + \left\| \mathbf{x}_0 \right\|_\Lambda^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l \\
\text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\
& \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathbf{b}, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathbf{t}(\boldsymbol{\theta}) \leq 0,
\end{aligned}$$

Parameter vector: $\boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$

Conditions to enforce on $\boldsymbol{\theta}$:

- Steady state
$$\mathbf{x}_\mathrm{r} = \boldsymbol{A}\mathbf{x}_\mathrm{r} + \boldsymbol{B}\mathbf{u}_\mathrm{r}$$
- Positive definiteness
$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{S} \\ \boldsymbol{S}^\top & \boldsymbol{R} \end{bmatrix} \succ 0$$
- Consistent uncertainty set
$$\boldsymbol{M}(\mathbf{s}_{i+1} - (\boldsymbol{A}\mathbf{s}_i + \boldsymbol{B}\mathbf{a}_i + \mathbf{b})) \leq \mathbf{m}$$
- Terminal set includes the reference
$$\boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_\mathrm{r} \leq \mathbf{t}(\boldsymbol{\theta})$$

Moreover:

- Riccati terminal cost and control law
$$\boldsymbol{K}(\boldsymbol{\theta}) = (\boldsymbol{R} + \boldsymbol{B}^\top\boldsymbol{P}(\boldsymbol{\theta})\boldsymbol{B})^{-1}(\boldsymbol{S}^\top + \boldsymbol{B}^\top\boldsymbol{P}(\boldsymbol{\theta})\boldsymbol{A})$$
$$\boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{Q} + \boldsymbol{A}^\top\boldsymbol{P}(\boldsymbol{\theta})\boldsymbol{A} - (\boldsymbol{S} + \boldsymbol{A}^\top\boldsymbol{P}(\boldsymbol{\theta})\boldsymbol{B})\boldsymbol{K}(\boldsymbol{\theta})$$
- Constraint tightening; RPI terminal set
$$\mathbf{f}(\boldsymbol{\theta}) = (\mathbf{c}_k(\boldsymbol{\theta}), \boldsymbol{T}(\boldsymbol{\theta}), \mathbf{t}(\boldsymbol{\theta}))$$

Safe and Stabilizing RL Based on Linear Tube MPC

Tube-Based Robust MPC

$$\hat{Q}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}) :=$$

$$\begin{aligned}
\min_{\mathbf{z}} \quad & \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 \\
& + \left\| \mathbf{x}_0 \right\|_\Lambda^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l \\
\mathrm{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\
& \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathbf{b}, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathrm{t}(\boldsymbol{\theta}) \leq 0,
\end{aligned}$$

RL problem

$$\begin{aligned}
\min_{\boldsymbol{\theta}} \quad & \psi(\boldsymbol{\theta}) \\
\mathrm{s.t.} \quad & \mathbf{x}_\mathrm{r} = \boldsymbol{A}\mathbf{x}_\mathrm{r} + \boldsymbol{B}\mathbf{u}_\mathrm{r} \\
& \boldsymbol{H} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{S} \\ \boldsymbol{S}^\top & \boldsymbol{R} \end{bmatrix} \succ 0 \\
& \boldsymbol{M}(\mathbf{s}_{i+1} - (\boldsymbol{A}\mathbf{s}_i + \boldsymbol{B}\mathbf{a}_i + \mathbf{b})) \leq \mathbf{m} \\
& \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_\mathrm{r} \leq \mathrm{t}(\boldsymbol{\theta})
\end{aligned}$$

Parameter vector: $\boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$

Moreover:

- Riccati terminal cost and control law

$$\boldsymbol{K}(\theta) = (\boldsymbol{R} + \boldsymbol{B}^\top \boldsymbol{P}(\theta)\boldsymbol{B})^{-1}(\boldsymbol{S}^\top + \boldsymbol{B}^\top \boldsymbol{P}(\theta)\boldsymbol{A})$$

$$\boldsymbol{P}(\theta) = \boldsymbol{Q} + \boldsymbol{A}^\top \boldsymbol{P}(\theta)\boldsymbol{A} - (\boldsymbol{S} + \boldsymbol{A}^\top \boldsymbol{P}(\theta)\boldsymbol{B})\boldsymbol{K}(\theta)$$

- Constraint tightening; RPI terminal set

$$\mathbf{f}(\boldsymbol{\theta}) = (\mathbf{c}_k(\boldsymbol{\theta}), \boldsymbol{T}(\boldsymbol{\theta}), \mathrm{t}(\boldsymbol{\theta}))$$

Safe and Stabilizing RL Based on Linear Tube MPC

Tube-Based Robust MPC

$$\hat{Q}_{\boldsymbol{\theta}}(\mathrm{s},\mathrm{a}) :=$$

$$\begin{aligned}
\min_{\mathbf{z}} \quad & \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 \\
& + \left\| \mathbf{x}_0 \right\|_{\Lambda}^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l \\
\text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\
& \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathrm{b}, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathrm{t}(\boldsymbol{\theta}) \leq 0,
\end{aligned}$$

RL problem

$$\begin{aligned}
\min_{\boldsymbol{\theta}} \quad & \psi(\boldsymbol{\theta}) \\
\text{s.t.} \quad & \mathbf{x}_\mathrm{r} = \boldsymbol{A}\mathbf{x}_\mathrm{r} + \boldsymbol{B}\mathbf{u}_\mathrm{r} \\
& \boldsymbol{H} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{S} \\ \boldsymbol{S}^\top & \boldsymbol{R} \end{bmatrix} \succ 0 \\
& \boldsymbol{M}(\mathrm{s}_{i+1} - (\boldsymbol{A}\mathrm{s}_i + \boldsymbol{B}\mathbf{a}_i + \mathrm{b})) \leq \mathbf{m} \\
& \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_\mathrm{r} \leq \mathrm{t}(\boldsymbol{\theta})
\end{aligned}$$

Parameter vector: $\boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$

Moreover:

- Riccati terminal cost and control law

$$\boldsymbol{K}(\boldsymbol{\theta}) = (\boldsymbol{R} + \boldsymbol{B}^\top \boldsymbol{P}(\boldsymbol{\theta})\boldsymbol{B})^{-1}(\boldsymbol{S}^\top + \boldsymbol{B}^\top \boldsymbol{P}(\boldsymbol{\theta})\boldsymbol{A})$$

$$\boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{Q} + \boldsymbol{A}^\top \boldsymbol{P}(\boldsymbol{\theta})\boldsymbol{A} - (\boldsymbol{S} + \boldsymbol{A}^\top \boldsymbol{P}(\boldsymbol{\theta})\boldsymbol{B})\boldsymbol{K}(\boldsymbol{\theta})$$

- Constraint tightening; RPI terminal set

$$\mathrm{f}(\boldsymbol{\theta}) = (\mathbf{c}_k(\boldsymbol{\theta}), \boldsymbol{T}(\boldsymbol{\theta}), \mathrm{t}(\boldsymbol{\theta}))$$

Derivative computation:

- apply chain rule:

$$\frac{\mathrm{d}\psi(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial\psi(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}} + \frac{\partial\psi(\boldsymbol{\theta})}{\partial\mathrm{f}(\boldsymbol{\theta})}\frac{\mathrm{d}\mathrm{f}(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}}$$

- can be cumbersome but it's not rocket science

A Simple Example of Safe and Stabilizing RL

Tube MPC

$$\min_{\mathbf{z}} \quad \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 + \left\| \mathbf{x}_0 \right\|_\Lambda^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l$$

$$\begin{aligned} \text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\ & \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathbf{b}, && k \in \mathbb{I}_0^{N-1}, \\ & \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \leq 0, && k \in \mathbb{I}_0^{N-1}, \\ & \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathbf{t}(\boldsymbol{\theta}) \leq 0, \end{aligned}$$

Double integrator

$$\mathbf{s}_+ = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \mathbf{s} + \begin{bmatrix} 0.05 \\ 0.1 \end{bmatrix} a + \mathbf{w}$$

$$\mathbf{s} \in [-1, 1]^2 \qquad a \in [-1, 1]$$

- Unknown noise set $\mathbb{W}$
- RL parameter: $\boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$
- Reward

$$-R(\mathbf{s}, a) = (p-3)^2 + 0.01v^2 + 0.01a^2$$

A Simple Example of Safe and Stabilizing RL

Tube MPC

$$\begin{aligned}
\min_{\mathbf{z}} \quad & \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 \\
& + \left\| \mathbf{x}_0 \right\|_\Lambda^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l \\
\text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\
& \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathbf{b}, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \\
& \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathbf{t}(\boldsymbol{\theta}) \leq 0,
\end{aligned}$$

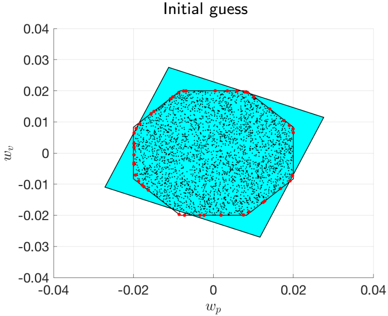
Double integrator

$$\mathbf{s}_+ = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \mathbf{s} + \begin{bmatrix} 0.05 \\ 0.1 \end{bmatrix} a + \mathbf{w}$$

$$\mathbf{s} \in [-1, 1]^2 \qquad a \in [-1, 1]$$

- Unknown noise set $\mathbb{W}$
- RL parameter: $\boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$
- Reward

$$-R(\mathbf{s}, a) = (p-3)^2 + 0.01v^2 + 0.01a^2$$



A Simple Example of Safe and Stabilizing RL

Tube MPC

$$\min_{\mathbf{z}} \quad \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 + \left\| \mathbf{x}_0 \right\|_\Lambda^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l$$

$$\begin{aligned} \text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\ & \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathbf{b}, \qquad k \in \mathbb{I}_0^{N-1}, \\ & \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \le 0, \qquad k \in \mathbb{I}_0^{N-1}, \\ & \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathbf{t}(\boldsymbol{\theta}) \le 0, \end{aligned}$$

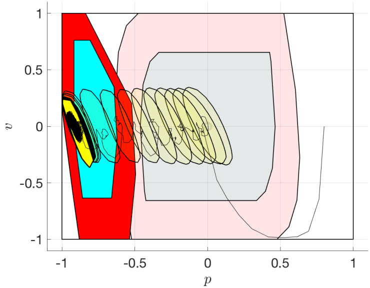
Double integrator

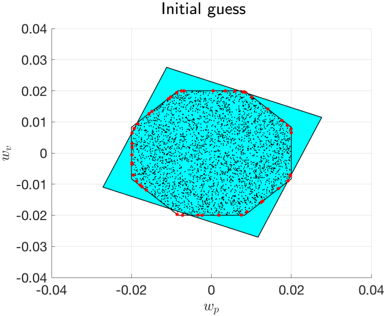
$$\mathbf{s}_+ = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \mathbf{s} + \begin{bmatrix} 0.05 \\ 0.1 \end{bmatrix} a + \mathbf{w}$$

$$\mathbf{s} \in [-1,1]^2 \qquad a \in [-1,1]$$

- Unknown noise set $\mathbb{W}$
- RL parameter: $\boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$
- Reward

$$-R(\mathbf{s}, a) = (p-3)^2 + 0.01v^2 + 0.01a^2$$





A Simple Example of Safe and Stabilizing RL

Tube MPC

$$\min_{\mathbf{z}} \sum_{k=0}^{N-1} \left\| \begin{matrix} \mathbf{x}_k - \mathbf{x}_\mathrm{r} \\ \mathbf{u}_k - \mathbf{u}_\mathrm{r} \end{matrix} \right\|_{\boldsymbol{H}}^2 + \left\| \mathbf{x}_N - \mathbf{x}_\mathrm{r} \right\|_{\boldsymbol{P}(\boldsymbol{\theta})}^2 + \left\| \mathbf{x}_0 \right\|_{\Lambda}^2 + \boldsymbol{\lambda}^\top \mathbf{x}_0 + l$$

$$\begin{aligned} \text{s.t.} \quad & \mathbf{x}_0 = \mathbf{s}, \qquad \mathbf{u}_0 = \mathbf{a}, \\ & \mathbf{x}_{k+1} = \boldsymbol{A}\mathbf{x}_k + \boldsymbol{B}\mathbf{u}_k + \mathbf{b}, && k \in \mathbb{I}_0^{N-1}, \\ & \boldsymbol{C}\mathbf{x}_k + \boldsymbol{D}\mathbf{u}_k + \mathbf{c}_k(\boldsymbol{\theta}) \leq 0, && k \in \mathbb{I}_0^{N-1}, \\ & \boldsymbol{T}(\boldsymbol{\theta})\mathbf{x}_N + \mathbf{t}(\boldsymbol{\theta}) \leq 0, \end{aligned}$$

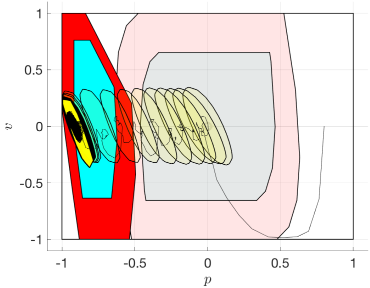
Double integrator

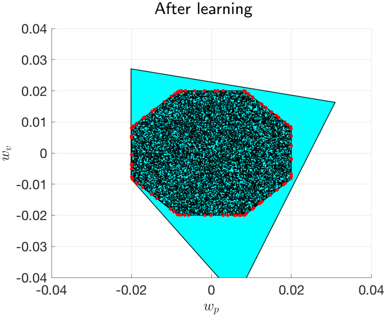
$$\mathbf{s}_+ = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \mathbf{s} + \begin{bmatrix} 0.05 \\ 0.1 \end{bmatrix} a + \mathbf{w}$$

$$\mathbf{s} \in [-1, 1]^2 \qquad a \in [-1, 1]$$

- Unknown noise set $\mathbb{W}$
- RL parameter: $\boldsymbol{\theta} = \{\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r}, \boldsymbol{H}, \Lambda, \boldsymbol{\lambda}, l, M\}$
- Reward

$$-R(\mathbf{s}, a) = (p-3)^2 + 0.01v^2 + 0.01a^2$$





Wrap-up

- MPC can be differentiated and that is
  - cheap
  - useful

Wrap-up

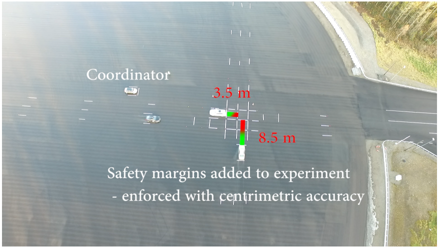
- MPC can be differentiated and that is
  - cheap
  - useful
- Differentiable MPC can be used as a function approximator within RL
  - RL can tune MPC to get the best possible performance
  - proxy to introduce safety and stability guarantees in RL
  - I have a talk on that on WeB17.2: Reinforcement Learning with Guarantees (A Tutorial on Policy Learning Methods for Advanced Controller Representations)

Wrap-up

- MPC can be differentiated and that is
  - cheap
  - useful

- Differentiable MPC can be used as a function approximator within RL
  - RL can tune MPC to get the best possible performance
  - proxy to introduce safety and stability guarantees in RL
  - I have a talk on that on WeB17.2: Reinforcement Learning with Guarantees (A Tutorial on Policy Learning Methods for Advanced Controller Representations)

- Differentiable MPC can also be exploited to build distributed algorithms, e.g.:



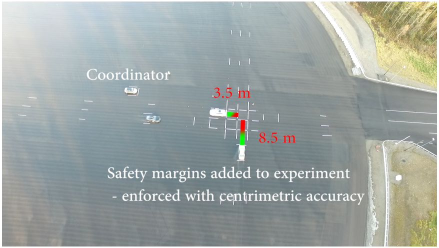
Joint work with R. Hult, S. Gros, P. Falcone

Wrap-up

- MPC can be differentiated and that is
  - cheap
  - useful

- Differentiable MPC can be used as a function approximator within RL
  - RL can tune MPC to get the best possible performance
  - proxy to introduce safety and stability guarantees in RL
  - I have a talk on that on WeB17.2: Reinforcement Learning with Guarantees (A Tutorial on Policy Learning Methods for Advanced Controller Representations)

- Differentiable MPC can also be exploited to build distributed algorithms, e.g.:



Joint work with R. Hult, S. Gros, P. Falcone

- Parametric sensitivities are also helpful to tune tracink MPC to behave similarly to economic MPC

Our work on the topic

1. **Stability-Constrained Markov Decision Processes Using MPC**, M. Zanon, M. Palladino, S. Gros, Automatica 2022
2. **Safe Reinforcement Learning with Stability & Safety Guarantees Using Robust MPCLearning for MPC with Stability & Safety Guarantees**, S. Gros, M. Zanon, Automatica 2022
3. **Bias Correction in Reinforcement Learning via the Deterministic Policy Gradient Method for MPC-Based Policies**, S. Gros, M. Zanon, ACC, 2021
4. **Reinforcement Learning based on MPC and the Stochastic Policy Gradient Method**, S. Gros, M. Zanon, ACC, 2021
5. **Safe Reinforcement Learning Using Robust MPC**, M. Zanon, S. Gros, IEEE TAC, 2021
6. **Safe Reinforcement Learning via Projection on a Safe Set: How to Achieve Optimality?** S. Gros, M. Zanon, A. Bemporad, IFAC World Congress 2020
7. **Reinforcement Learning for Mixed-Integer Problems Based on MPC**, S. Gros, M. Zanon, IFAC World Congress 2020
8. **Reinforcement Learning Based on Real-Time Iteration NMPC**, M. Zanon, V. Kungurstev, S. Gros, IFAC World Congress 2020
9. **Data-driven Economic NMPC using Reinforcement Learning**, S. Gros, M. Zanon, IEEE TAC, 2020
10. **Practical Reinforcement Learning of Stabilizing Economic MPC**, M. Zanon, S. Gros, A. Bemporad, European Control Conference 2019