3.3 Model-based reinforcement learning

In model-based reinforcement learning, we assume to know the dynamic and reward function p_x and p_r, respectively.

3.3.1 Infinite horizon Dynamic Programming

Next we discuss *Value and Policy Iteration* used in reinforcement learning. Therefore, we will restrict our self to deterministic policies, the infinite horizon setting ($N = \infty$), and the value-discrete case. In this case, the value-discrete Bellman Expectation equation (3.59) with (3.19) can be written as

$$\begin{aligned} Q^\pi(\mathbf{x}, \mathbf{u}) &= \bar{r}(\mathbf{x}, \mathbf{u}) + \gamma \sum_{\mathbf{x}' \in \mathcal{X}} \mathsf{p}_\mathsf{x}(\mathbf{x}' \mid \mathbf{x}, \mathbf{u}) V^\pi(\mathbf{x}') \\ Q^\pi(\mathbf{x}_i, \mathbf{u}_l) &= \bar{r}(\mathbf{x}_i, \mathbf{u}_l) + \gamma \sum_{\mathbf{x}_j \in \mathcal{X}} \mathsf{p}_\mathsf{x}(\mathbf{x}_j \mid \mathbf{x}_i, \mathbf{u}_l) V^\pi(\mathbf{x}_j) \\ &= \bar{r}(\mathbf{x}_i, \mathbf{u}_l) + \gamma \sum_{j=1}^{|\mathcal{X}|} p_{ij}^{\mathbf{u}_l} V^\pi(\mathbf{x}_j) \ . \end{aligned} \tag{3.67}$$

and

$$\begin{aligned} V^\pi(\mathbf{x}) &= \bar{r}^\pi(\mathbf{x}) + \gamma \sum_{\mathbf{x}' \in \mathcal{X}} \sum_{\mathbf{u} \in \mathcal{U}} \mathsf{p}_\mathsf{x}(\mathbf{x}' \mid \mathbf{x}, \mathbf{u}) \pi(\mathbf{u} \mid \mathbf{x}) V^\pi(\mathbf{x}') \\ V^\pi(\mathbf{x}_i) &= \bar{r}^\pi(\mathbf{x}_i) + \gamma \sum_{\mathbf{x}_j \in \mathcal{X}} \sum_{\mathbf{u} \in \mathcal{U}} \mathsf{p}_\mathsf{x}(\mathbf{x}_j \mid \mathbf{x}_i, \mathbf{u}) \pi(\mathbf{u} \mid \mathbf{x}_i) V^\pi(\mathbf{x}_j) \\ &= \bar{r}^\pi(\mathbf{x}_i) + \gamma \sum_{j=1}^{|\mathcal{X}|} p_{ij}^\pi V^\pi(\mathbf{x}_j) \ . \end{aligned} \tag{3.68}$$

Policy Iteration

When the model is fully known, following Bellman Expectation Equation, we can use Dynamic Programming (DP) to iteratively evaluate value functions and improve the policy.

- *Policy Evaluation* (value update) is to compute the state-value functions $V^\pi(\mathbf{x})$ for a given policy π according to the Bellman Expectation Equation (3.58).
- Based on the state-value functions, *Policy Improvement* (policy update) generates a better policy $V^{\pi'}(\mathbf{x}) \geq V^\pi(\mathbf{x})$ by acting greedily.

Once again, we show Policy Iteration for the infinite horizon ($N = \infty$) and value-discrete case ($\mathcal{X}$ and $\mathcal{U}$ finite) only.

Policy Evaluation

Define the *state-value vector* $\mathbf{v}^\pi \in \mathbb{R}^{|\mathcal{X}|+1}$ via

$$\mathbf{v}^\pi = \begin{bmatrix} V^\pi(\mathbf{x}_0) & \dots & V^\pi\left(\mathbf{x}_{|\mathcal{X}|}\right) \end{bmatrix}^\top, \tag{3.69}$$

and the *reward vector* $\mathbf{r}^\pi \in \mathbb{R}^{|\mathcal{X}|+1}$ via

$$\mathbf{r}^\pi = \begin{bmatrix} \bar{r}_0^\pi & \cdots & \bar{r}_{|\mathcal{X}|}^\pi \end{bmatrix}^\top. \tag{3.70}$$

Then, the value-discrete infinite horizon Bellman Expectation equation (3.58) reads in vector notation as

$$\mathbf{v}^\pi = \mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{v}^\pi \ . \tag{3.71}$$

Policy Evaluation involves computing the state-value vector $\mathbf{v}^\pi$ for a given policy π solving the Bellman Expectation Equation. We also refer to it as the prediction problem. In this context, it is important to note that the induced matrix norm $\|\mathbf{P}^\pi\|_\infty$ is equal to 1, and since $\gamma < 1$, the matrix $\mathbf{I} - \gamma \mathbf{P}^\pi$ is nonsingular. Consequently, for any reward vector $\mathbf{r}^\pi$, there exists a unique $\mathbf{v}^\pi$ that satisfies (3.58). In principle it is possible to deduce from (3.58)

$$\mathbf{v}^\pi = (\mathbf{I} - \gamma \mathbf{P}^\pi)^{-1} \mathbf{r}^\pi \ . \tag{3.72}$$

However, applying this formula can be challenging in practical applications of Markov Decision Problems, mainly due to the large size of the state space $\mathcal{X}$, represented by the integer $|\mathcal{X}|$. Additionally, matrix inversion has a cubic complexity, making it impractical to invert the matrix $\mathbf{I} - \gamma \mathbf{P}^\pi$. Consequently, alternative approaches need to be explored. The Contraction Mapping Theorem provides a feasible solution in such cases.

Theorem 3.3. *(Iterative policy evaluation, Theorem 2 in [3.11]) The map* $\mathbf{y} \mapsto \mathrm{T}(\mathbf{y}) = \mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{y}$ *is monotone and is a contraction with respect to the* ℓ_∞*-norm, with contraction constant* γ*. Therefore, we can choose some vector* $\mathbf{y}^{(0)}$ *arbitrarily, and then define*

$$\mathbf{y}^{(i+1)} = \mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{y}^{(i)} \ . \tag{3.73}$$

Then $\mathbf{y}^{(i)}$ *converges to the state value vector* $\mathbf{v}^\pi$*.*

See [3.11] for a proof. Similar results can be obtained for the value-continuous case, see [3.5]. Therefore, the transition tensor $\mathbf{P}^\pi$ is replaced by a transition operator.

Policy Improvement

Given a state-value function $V^\pi(\mathbf{x})$ of a policy π, it is possible to perform a *policy improvement step*, cf. [3.7, pp. 84–85]. The result is a potentially better strategy π', where $V^{\pi'}(\mathbf{x}) \geq V^\pi(\mathbf{x})$ holds. To obtain π', the previous strategy π is no longer followed, but instead the *greedy policy* is chosen, which maximizes the expected reward:

$$\pi' \leftarrow \arg\max_{\mathbf{u}\in\mathcal{U}} Q^{\pi}(\mathbf{x},\mathbf{u}) = \arg\max_{\mathbf{u}\in\mathcal{U}} \left[\bar{r}^{\pi} + \gamma \mathbb{E}_{\mathbf{x}'\sim \mathsf{p}_{\mathsf{x}}(\mathbf{x}'|\mathbf{x},\mathbf{u})}\{V^{\pi}(\mathbf{x}')\}\right] . \tag{3.74}$$

This can be proven by the Policy Improvement Theorem 3.4:

Theorem 3.4. *(Policy Improvement Theorem) We consider two policies $\pi(\mathbf{u} \mid \mathbf{x})$ and $\pi'(\mathbf{u} \mid \mathbf{x})$. Let us define*

$$Q^{\pi}(\mathbf{x},\pi') = \mathbb{E}_{\mathbf{u}\sim\pi'(\mathbf{u}|\mathbf{x})}\{Q^{\pi}(\mathbf{x},\mathbf{u})\} . \tag{3.75}$$

If $\forall \mathbf{x} \in \mathcal{X}$, we have that $Q^{\pi}(\mathbf{x},\pi') \geq V^{\pi}(\mathbf{x})$, then it holds that

$$V^{\pi}(\mathbf{x}) \leq Q^{\pi}(\mathbf{x},\pi') \leq V^{\pi'}(\mathbf{x}), \forall x . \tag{3.76}$$

This means that π' is at least as good a policy as π.

Proof. By expanding Q^{π}, we can get that $\forall \mathbf{x} \in \mathcal{X}$,

$$\begin{aligned}
V^{\pi}(\mathbf{x}) \leq Q^{\pi}(\mathbf{x},\pi') &= \mathbb{E}_{\mathbf{u}\sim\pi'(\mathbf{u}|\mathbf{x})}\{Q^{\pi}(\mathbf{x},\mathbf{u})\} \\
&= \mathbb{E}_{\mathbf{u}\sim\pi'(\mathbf{u}|\mathbf{x})}\left\{\mathsf{r}_k + \gamma \underbrace{V^{\pi}(\mathbf{x}')}_{\leq Q^{\pi}(\mathbf{x}',\pi')}\right\} \\
&\leq \mathbb{E}_{\mathbf{u}\sim\pi'(\mathbf{u}|\mathbf{x})}\{\mathsf{r}_k + \gamma Q^{\pi}(\mathbf{x}',\pi')\} \\
&= \mathbb{E}_{\mathbf{u},\mathbf{u}'\sim\pi'}\left\{\mathsf{r}_k + \gamma \mathsf{r}_{k+1} + \gamma^2 V^{\pi}(\mathbf{x}'')\right\} \\
&\leq \dots \\
&\leq \mathbb{E}_{\mathbf{u},\mathbf{u}',\mathbf{u}''\dots\sim\pi'}\left\{\mathsf{r}_k + \gamma \mathsf{r}_{k+1} + \gamma^2 \mathsf{r}_{k+2} + \dots\right\} \\
&= V^{\pi'}(\mathbf{x}) .
\end{aligned} \tag{3.77}$$

□

The *Infinite Horizon Policy Iteration Algorithm* 1 shows how to use policy evaluation and policy improvement to find an optimal policy π^*.

Algorithm 1: Infinite Horizon Policy Iteration Algorithm

Data: θ is a small number
/* Initialization */
Initialize $V^\pi(\mathbf{x})$ arbitrarily;
Randomly initialize policy $\boldsymbol{\mu}$;
/* Policy Evaluation */
$\Delta \leftarrow 0$;
while $\Delta < \theta$ **do**
  for *each* $\mathbf{x} \in \mathcal{X}$ **do**
    $v^\pi \leftarrow V^\pi(\mathbf{x})$;
    $V^\pi(\mathbf{x}) \leftarrow \bar{r}^\pi + \gamma \sum_{\mathbf{x}' \in \mathcal{X}} \mathsf{p}_{\mathsf{x}}(\mathbf{x}' \mid \mathbf{x}, \mathbf{u}) V^\pi(\mathbf{x}')$;
    $\Delta \leftarrow \max(\Delta, |v^\pi - V^\pi(\mathbf{x})|)$;
  end
end
/* Policy Improvement */
policy is stable $\leftarrow$ true;
for *each* $\mathbf{x} \in \mathcal{X}$ **do**
  old-policy $\leftarrow \pi$;
  $Q^\pi(\mathbf{x}, \mathbf{u}) \leftarrow \bar{r}(\mathbf{x}, \mathbf{u}) + \gamma \sum_{\mathbf{x}' \in \mathcal{X}} \mathsf{p}_{\mathsf{x}}(\mathbf{x}' \mid \mathbf{x}, \mathbf{u}) V^\pi(\mathbf{x}')$;
  $\boldsymbol{\mu}(\mathbf{x}) \leftarrow \arg\max_{\mathbf{u} \in \mathcal{U}} Q^\pi(\mathbf{x}, \mathbf{u})$;
  if *old-policy* $\neq \pi$ **then**
    policy is stable $\leftarrow$ false;
  end
end
if *policy is stable* **then**
  return $V^\pi \approx V^*$ and $\pi \approx \pi^*$;
else
  go to Policy Evaluation;
end

Remark 3.9. *A Jupyter Notebook of the* Infinite Horizon Policy Iteration Algorithm *1 for the Frozen Lake Example can be found here.*

Value Iteration

Value iteration is the computation of the state-value function V^π by the Dynamic Programming described in Theorem 3.2. The iteration is performed for the entire state space starting from arbitrarily initialized residual reward [3.7, p. 83]. The iteration rule in Theorem 3.2 represents a fixed point iteration for V^π and converges against V^*, see [3.9]. Therefore, we introduce the *state-value vector*

$$\mathbf{v}^\pi = \begin{bmatrix} V^\pi(\mathbf{x}_0) & \dots & V^\pi\left(\mathbf{x}_{|\mathcal{X}|}\right) \end{bmatrix}^\top \in \mathbb{R}^{|\mathcal{X}|+1} \tag{3.78}$$

and define the *Bellman Iteration Map*[6] $\mathrm{B} : \mathbb{R}^{|\mathcal{X}|+1} \to \mathbb{R}^{|\mathcal{X}|+1}$ via

$$\mathbf{y} \mapsto (\mathrm{B}(\mathbf{y}))[i] = \max_{\mathbf{u}_l \in \mathcal{U}} \left[\bar{r}(\mathbf{x}_i, \mathbf{u}_l) + \gamma \sum_{j=1}^{|\mathcal{X}|} p_{ij}^{\mathbf{u}_l} \mathbf{y}[j] \right]. \tag{3.79}$$

Theorem 3.5. *(Theorem 3 in [3.11]) The map* B *is monotone and a contraction with respect to the* ℓ_∞*-norm. Therefore, the fixed point* $\mathbf{v}^{(\infty)}$ *of the map* B *satisfies the relation*

$$\mathbf{v}^{(\infty)}[i] = \max_{\mathbf{u}_l \in \mathcal{U}} \left[\bar{r}(\mathbf{x}_i, \mathbf{u}_l) + \gamma \sum_{j=0}^{|\mathcal{X}|-1} p_{ij}^{\mathbf{u}_l} \mathbf{v}^{(\infty)}[j] \right]. \tag{3.80}$$

See [3.11] for a proof. Given an initial guess $\mathbf{v}^{(0)} \in \mathbb{R}^{|\mathcal{X}|+1}$, one can iteratively employ the Bellman iteration. These iterations will converge to the unique fixed point, denoted as $\mathbf{v}^{(\infty)}$, of the operator B. The importance of this iterative process is elaborated in the subsequent theorem.

Theorem 3.6. *(Theorem 4 in [3.11]) Define* $\mathbf{v}^{(\infty)} \in \mathbb{R}^{|\mathcal{X}|+1}$ *to be the unique fixed point of* B*, and define* $\mathbf{v}^* \in \mathbb{R}^{|\mathcal{X}|+1}$ *to equal* $V^*(\mathbf{x}), \mathbf{x} \in \mathcal{X}$*, where* $V^*(\mathbf{x})$ *is defined in* (3.47)*. Then* $\mathbf{v}^{(\infty)} = \mathbf{v}^*$*.*

See [3.11] for a proof. Therefore, the optimal value vector can be computed using the Bellman iteration. However, knowing the optimal value vector does not, by itself, give us an optimal policy. Therefore, after the convergence Bellman iteration, a policy determination according to (3.64) is performed.

The *Infinite Horizon Value Iteration Algorithm* 2 show how to use Value Iteration to find an optimal policy π^*.

[6] Called Backup operator in the reinforcement learning literature.

Algorithm 2: Infinite Horizon Value Iteration Algorithm

Data: θ is a small number
Result: Find $V^*(\mathbf{x})$ and $\boldsymbol{\mu}^*$,$\forall \mathbf{x} \in \mathcal{X}$
`/* Initialization */`
Initialize $V(\mathbf{x})$ arbitrarily;
$V^\pi(\mathbf{x}) \leftarrow 0$,$\forall \mathbf{x} \in \mathcal{X}$;
`/* Loop until convergence */`
$\Delta \leftarrow 0$;
`/* Optimal Value Determination */`
while $\Delta < \theta$ **do**
  for *each* $\mathbf{x} \in \mathcal{X}$ **do**
    $v \leftarrow V^\pi(\mathbf{x})$;
    $Q^\pi(\mathbf{x}) \leftarrow \bar{r}(\mathbf{x}, \mathbf{u}) + \gamma \sum_{\mathbf{x}' \in \mathcal{X}} \mathsf{p}_{\mathsf{x}}(\mathbf{x}' \mid \mathbf{x}, \mathbf{u}) V^\pi(\mathbf{x}')$;
    $V^\pi(\mathbf{x}) \leftarrow \max_{\mathbf{u} \in \mathcal{U}} [Q^\pi(\mathbf{x})]$;
    $\Delta \leftarrow \max(\Delta, |v - V^\pi(\mathbf{x})|)$;
  end
end
`/* Optimal Policy Determination */`
$Q^*(\mathbf{x}, \mathbf{u}) \leftarrow \bar{r}(\mathbf{x}, \mathbf{u}) + \gamma \sum_{\mathbf{x}' \in \mathcal{X}} \mathsf{p}_{\mathsf{x}}(\mathbf{x}' \mid \mathbf{x}, \mathbf{u}) V^\pi(\mathbf{x}')$;
$\boldsymbol{\mu}^*(\mathbf{x}) \leftarrow \arg\max_{\mathbf{u} \in \mathcal{U}} Q^*(\mathbf{x}, \mathbf{u})$;
return μ^* ,$\forall \mathbf{x} \in \mathcal{X}$

Remark 3.10. *A Jupyter Notebook of the* Infinite Horizon Value Iteration Algorithm *2 for the Frozen Lake Example can be found here.*

3.3.2 Classical Optimal Control as MDP

Classical optimal control aims to solving

$$\begin{aligned} \min_{\substack{\mathbf{x}_0,\dots,\mathbf{x}_N \in \mathbb{R}^n \\ \mathbf{u}_0,\dots,\mathbf{u}_{N-1} \in \mathbb{R}^m}} \quad & l_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} \bar{l}_k(\mathbf{x}_k, \mathbf{u}_k) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k) \ , \quad \mathbf{x}_0 = \bar{\mathbf{x}}_0 \ . \end{aligned} \tag{3.81}$$

This Section demonstrates that (3.81) is a special case of a classical MDP.

In OCP, we typically aim for minimization rather than maximization, i.e. we introduce the loss $\mathsf{l}_k = -\mathsf{r}_k$.

Markov Decision Process

Since both, the loss and the dynamic functions are deterministic, we can express them by a combined Dirac delta distribution

$$\mathbf{x}_{k+1}, \mathsf{l}_k \sim \mathsf{p}(\mathbf{x}_{k+1}, l_k | \mathbf{x}_k, \mathbf{u}_k) = \delta\Big(\mathbf{x}_{k+1} - \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k)\Big)\delta\Big(l_k - \bar{l}_k(\mathbf{x}_k, \mathbf{u}_k)\Big) \ . \tag{3.82}$$

Similarly, the initial condition distribution is given by

$$\mathbf{x}_0 \sim \mathsf{p}_{\mathsf{x}_0}(\mathbf{x}_0) = \delta(\mathbf{x}_0 - \bar{\mathbf{x}}_0) \ , \tag{3.83}$$

with the deterministic initial state $\bar{\mathbf{x}}_0 \in \mathcal{X}$. We can compute the conditional state transition dynamics

$$\mathsf{p}_{\mathsf{x}}(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k) = \int_{\mathcal{L}} \mathsf{p}(\mathbf{x}_{k+1}, l_k|\mathbf{x}_k, \mathbf{u}_k)\, \mathrm{d}l_k = \delta\Big(\mathbf{x}_{k+1} - \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k)\Big) \tag{3.84}$$

and the conditional loss density

$$\mathsf{p}_{\mathsf{l}}(l_k|\mathbf{x}_k, \mathbf{u}_k) = \int_{\mathcal{X}} p(\mathbf{x}_{k+1}, l_k|\mathbf{x}_k, \mathbf{u}_k)\, \mathrm{d}\mathbf{x}_{k+1} = \delta\Big(l_k - \bar{l}_k(\mathbf{x}_k, \mathbf{u}_k)\Big) \ . \tag{3.85}$$

Note that (3.84) and (3.85) yields $\mathsf{p}(\mathbf{x}_{k+1}, l_k|\mathbf{x}_k, \mathbf{u}_k) = \mathsf{p}_{\mathsf{x}}(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k)\mathsf{p}_{\mathsf{l}}(l_k|\mathbf{x}_k, \mathbf{u}_k)$, i.e. the state $\mathbf{x}_{k+1}$ and the loss l_k are conditionally independent given $\mathbf{x}_k$ and $\mathbf{u}_k$.

The expected next state is computed as

$$\mathbb{E}[\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k] = \int_{\mathcal{X}} \mathbf{x}_{k+1}\mathsf{p}_{\mathsf{x}}(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k)\, \mathrm{d}\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \ , \tag{3.86}$$

as well as the expected loss

$$\mathbb{E}[\mathsf{l}_k|\mathbf{x}_k, \mathbf{u}_k] = \int_{\mathcal{L}} l_k \mathsf{p}_{\mathsf{l}}(l_k|\mathbf{x}_k, \mathbf{u}_k) = \bar{l}_k(\mathbf{x}_k, \mathbf{u}_k) \ . \tag{3.87}$$

Policy and Rollout

Typically, the result of solving an OCP is a deterministic feedforward policy

$$\pi(\mathbf{u}_k|\mathbf{x}_k) = \delta(\mathbf{u}_k - \bar{\mathbf{u}}_k) \ , \tag{3.88}$$

or $\boldsymbol{\mu}_k(\mathbf{x}_k) = \bar{\mathbf{u}}_k$. This yields the distribution of a rollout (3.42) when applying the deterministic policy (3.88)

$$\mathsf{p}^{\pi}(\boldsymbol{\tau}) = \delta(\mathbf{x}_0 - \bar{\mathbf{x}}_0) \prod_{k=0}^{N} \delta\Big(\mathbf{x}_{k+1} - \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)\Big)\delta(\mathbf{u}_k - \bar{\mathbf{u}}_k) \tag{3.89}$$

Note that (3.89) is a distrubution of a deterministic trajectory, i.e. it is non-zero if and only if $\mathbf{x}_0 = \bar{\mathbf{x}}_0$, $\mathbf{u}_k = \bar{\mathbf{u}}_k$ and $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$. Thus we can rewrite (3.89) as

$$\mathsf{p}^{\pi}(\boldsymbol{\tau}) = \delta(\mathbf{x}_0 - \bar{\mathbf{x}}_0) \prod_{k=0}^{N} \delta(\mathbf{x}_{k+1} - \bar{\mathbf{x}}_{k+1})\delta(\mathbf{u}_k - \bar{\mathbf{u}}_k) = \delta(\boldsymbol{\tau} - \bar{\boldsymbol{\tau}}) \ , \tag{3.90}$$

with $\bar{\boldsymbol{\tau}} = (\bar{\mathbf{x}}_0, \bar{\mathbf{u}}_0, \ldots, \bar{\mathbf{u}}_{N-1}, \bar{\mathbf{x}}_N)$ and the condition $\bar{\mathbf{x}}_{k+1} = \mathbf{f}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)$. Due to the properties of the dirac delta function, the expectation of any function $f(\boldsymbol{\tau})$ gives

$$\mathbb{E}_{\pi}[f(\boldsymbol{\tau})] = \int_{\boldsymbol{\tau}} f(\boldsymbol{\tau})\mathsf{p}^{\pi}(\boldsymbol{\tau})\, \mathrm{d}\boldsymbol{\tau} = f(\bar{\boldsymbol{\tau}}) \ , \tag{3.91}$$

with $\bar{\mathbf{x}}_{k+1} = \mathbf{f}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)$ and initial condition $\bar{\mathbf{x}}_0$ (as we would expect for a deterministic rollout). Thus, all concepts from the previous sections can by applied to the deterministic case, by just replacing expectations with the corresponding function evaluated at the deterministic values.

Overall Expected Loss and OCP

From (3.90) the expected loss under a deterministic policy can be computed as (3.88)

$$\begin{aligned}\mathbb{E}_\pi[\mathsf{l}_k] &= \int_{\boldsymbol{\tau}} \mathbb{E}_\pi[\mathsf{l}_k|\boldsymbol{\tau}]\mathsf{p}^\pi(\boldsymbol{\tau})\,\mathrm{d}\boldsymbol{\tau} \\ &= \int_{\boldsymbol{\tau}} \mathbb{E}_\pi[\mathsf{l}_k|\mathbf{x}_k,\mathbf{u}_k]\mathsf{p}^\pi(\boldsymbol{\tau})\,\mathrm{d}\boldsymbol{\tau} \\ &\overset{(3.87)}{=} \int_{\boldsymbol{\tau}} \bar{l}_k(\mathbf{x}_k,\mathbf{u}_k)\mathsf{p}^\pi(\boldsymbol{\tau})\,\mathrm{d}\boldsymbol{\tau} \\ &\overset{(3.91)}{=} \bar{l}_k(\bar{\mathbf{x}}_k,\bar{\mathbf{u}}_k)\ ,\end{aligned} \tag{3.92}$$

where the second equality follows from the independence of l_k and $\mathbf{x}_i, \mathbf{u}_i$, $i \neq k$. With $\mathsf{L}_0 = -\mathsf{R}_0$ we get the overall expected loss by

$$\begin{aligned}\mathbb{E}_\pi[\mathsf{L}_0] &= \mathbb{E}_\pi\left[\mathsf{l}_N + \sum_{k=0}^{N-1} \mathsf{l}_k\right] \\ &= \mathbb{E}_\pi[\mathsf{l}_N] + \sum_{k=0}^{N-1} \mathbb{E}_\pi[\mathsf{l}_k] \\ &= \bar{l}_N(\bar{\mathbf{x}}_N) + \sum_{k=0}^{N-1} \bar{l}_k(\bar{\mathbf{x}}_k,\bar{\mathbf{u}}_k)\ , \\ &\text{with } \bar{\mathbf{x}}_{k+1} = \mathbf{f}(\bar{\mathbf{x}}_k,\bar{\mathbf{u}}_k)\ .\end{aligned} \tag{3.93}$$

Hence, minimizing $\mathbb{E}_\pi[\mathsf{L}_0]$ directly yields the OCP (3.81) with the corresponding dynamics constraint. This optimal control can be considered as solving a deterministic Markov decision process.

Value Function

The action-value function (3.44a) for the OCP (3.81) evaluates to

$$Q_k^\pi(\mathbf{x}_k,\mathbf{u}_k) = \mathbb{E}_\pi[\mathsf{L}_k|\mathbf{x}_k,\mathbf{u}_k] = \underbrace{\bar{l}_k(\mathbf{x}_k,\mathbf{u}_k)}_{\text{Apply } \mathbf{u}_k \text{ at } \mathbf{x}_k} + \underbrace{\bar{l}_N(\bar{\mathbf{x}}_N) + \sum_{i=k+1}^{N-1} \bar{l}(\bar{\mathbf{x}}_i,\bar{\mathbf{u}}_i)}_{\text{Continue with } \pi = (\bar{\mathbf{u}}_{k+1},\dots,\bar{\mathbf{u}}_{N-1})}\ , \tag{3.94}$$

with the conditions $\bar{\mathbf{x}}_{k+1} = \mathbf{f}(\mathbf{x}_k,\mathbf{u}_k)$ and $\bar{\mathbf{x}}_{i+1} = \mathbf{f}(\bar{\mathbf{x}}_i,\bar{\mathbf{u}}_i)$ for $i > k$. Similarly, the state-value function (3.45a) for the OCP (3.81) yields

$$V_k^\pi(\mathbf{x}_k) = \mathbb{E}_\pi[\mathsf{L}_k|\mathbf{x}_k] = \underbrace{\bar{l}_k(\mathbf{x}_k,\bar{\mathbf{u}}_k) + \bar{l}_N(\bar{\mathbf{x}}_N) + \sum_{i=k+1}^{N-1} \bar{l}_i(\bar{\mathbf{x}}_i,\bar{\mathbf{u}}_i)}_{\text{Apply policy } \pi = (\bar{\mathbf{u}}_k,\dots,\bar{\mathbf{u}}_{N-1}) \text{ at state } \mathbf{x}_k} \tag{3.95}$$

with the conditions $\bar{\mathbf{x}}_{k+1} = \mathbf{f}(\mathbf{x}_k,\bar{\mathbf{u}}_k)$ and $\bar{\mathbf{x}}_{i+1} = \mathbf{f}(\bar{\mathbf{x}}_i,\bar{\mathbf{u}}_i)$ for $i > k$. Note that a direct consequence of (3.95) is the boundary condition $V_N^\pi(\mathbf{x}_N) = \bar{l}_N(\mathbf{x}_N)$.

Comparing (3.94) and (3.95) directly yields the relations

$$V_k^\pi(\mathbf{x}_k) = Q_k^\pi(\mathbf{x}_k, \bar{\mathbf{u}}_k) \tag{3.96a}$$

$$Q_k^\pi(\mathbf{x}_k, \mathbf{u}_k) = \bar{l}_k(\mathbf{x}_k, \mathbf{u}_k) + V_{k+1}^\pi\Big(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)\Big) \ , \tag{3.96b}$$

which corresponds from the result of (3.46) and the Bellman Expectation equation (3.59).

LQR - the finite time horizon case

In the following we assume a special case of (3.81) with linear dynamics

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{\Phi}\mathbf{x}_k + \mathbf{\Gamma}\mathbf{u}_k \tag{3.97}$$

and quadratic costs

$$\begin{aligned} \bar{l}_N(\mathbf{x}_N) &= \frac{1}{2}\mathbf{x}_N^\mathrm{T}\mathbf{L}_{\mathbf{xx},N}\mathbf{x}_N \\ \bar{l}_k(\mathbf{x}_k, \mathbf{u}_k) &= \frac{1}{2}\mathbf{x}_k^\mathrm{T}\mathbf{L}_{\mathbf{xx},k}\mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^\mathrm{T}\mathbf{L}_{\mathbf{uu},k}\mathbf{u}_k + \mathbf{x}_k^\mathrm{T}\mathbf{L}_{\mathbf{xu},k}\mathbf{u}_k \ . \end{aligned} \tag{3.98}$$

Assume that the optimal value function is quadratic at time $k+1$, i.e. $V_{k+1}^*(\mathbf{x}_{k+1}) = \frac{1}{2}\mathbf{x}_{k+1}^\mathrm{T}\mathbf{V}_{\mathbf{xx},k+1}\mathbf{x}_{k+1}$, which is obviously true for $k+1 = N$ with $\mathbf{V}_{\mathbf{xx},N} = \mathbf{L}_{\mathbf{xx},N}$. Using Theorem 3.2, the optimal action-value function then evaluates to

$$\begin{aligned} Q_k^*(\mathbf{x}_k, \mathbf{u}_k) = &\underbrace{\frac{1}{2}\mathbf{x}_k^\mathrm{T}\mathbf{L}_{\mathbf{xx},k}\mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^\mathrm{T}\mathbf{L}_{\mathbf{uu},k}\mathbf{u}_k + \mathbf{x}_k^\mathrm{T}\mathbf{L}_{\mathbf{xu},k}\mathbf{u}_k}_{\bar{l}_k(\mathbf{x}_k,\mathbf{u}_k)} + \\ &\underbrace{\frac{1}{2}(\mathbf{\Phi}\mathbf{x}_k + \mathbf{\Gamma}\mathbf{u}_k)^\mathrm{T}\mathbf{V}_{\mathbf{xx},k+1}(\mathbf{\Phi}\mathbf{x}_k + \mathbf{\Gamma}\mathbf{u}_k)}_{V_{k+1}^*\left(\mathbf{f}(\mathbf{x}_k,\mathbf{u}_k)\right)} \ . \end{aligned} \tag{3.99}$$

Hence, the optimal action-value function (3.99) can be rewritten as a quadratic form

$$Q_k^*(\mathbf{x}_k, \mathbf{u}_k) = \frac{1}{2}\mathbf{x}_k^\mathrm{T}\mathbf{Q}_{\mathbf{xx},k}\mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^\mathrm{T}\mathbf{Q}_{\mathbf{uu},k}\mathbf{u}_k + \mathbf{x}_k^\mathrm{T}\mathbf{Q}_{\mathbf{xu},k}\mathbf{u}_k \ , \tag{3.100a}$$

with

$$\mathbf{Q}_{\mathbf{xx},k} = \mathbf{L}_{\mathbf{xx},k} + \mathbf{\Phi}^\mathrm{T}\mathbf{V}_{\mathbf{xx},k+1}\mathbf{\Phi} \tag{3.100b}$$

$$\mathbf{Q}_{\mathbf{uu},k} = \mathbf{L}_{\mathbf{uu},k} + \mathbf{\Gamma}^\mathrm{T}\mathbf{V}_{\mathbf{xx},k+1}\mathbf{\Gamma} \tag{3.100c}$$

$$\mathbf{Q}_{\mathbf{xu},k} = \mathbf{L}_{\mathbf{xu},k} + \mathbf{\Phi}^\mathrm{T}\mathbf{V}_{\mathbf{xx},k+1}\mathbf{\Gamma} \ . \tag{3.100d}$$

From Theorem 3.2, the optimal policy can be obtained by minimizing (3.100), which yields

$$\mathbf{u}_k^* = \boldsymbol{\mu}_k^*(\mathbf{x}_k) = \underbrace{-\mathbf{Q}_{\mathbf{uu},k}^{-1}\mathbf{Q}_{\mathbf{xu},k}}_{=\mathbf{K}_k}\mathbf{x}_k \ . \tag{3.101}$$

Theorem 3.2 allows to compute the optimal state-value function as

$$V_k^*(\mathbf{x}_k) = \mathbf{Q}_k^*(\mathbf{x}_k, \boldsymbol{\mu}_k^*(\mathbf{x}_k)) = \frac{1}{2}\mathbf{x}_k^{\mathrm{T}}\mathbf{V}_{\mathbf{xx},k}\mathbf{x}_k \ , \tag{3.102a}$$

with

$$\mathbf{V}_{\mathbf{xx},k} = \mathbf{Q}_{\mathbf{xx},k} + \mathbf{K}_k^{\mathrm{T}}\mathbf{Q}_{\mathbf{uu},k}\mathbf{K}_k + \mathbf{Q}_{\mathbf{xu},k}\mathbf{K}_k + \mathbf{K}_k^{\mathrm{T}}\mathbf{Q}_{\mathbf{xu},k}^{\mathrm{T}} \ . \tag{3.102b}$$

Hence, $V_k^*(\mathbf{x}_k)$ is again a quadratic function of $\mathbf{x}_k$ and we can repeat until $k = 0$.

LQR - the finite time horizon case with affine dynamics

Consider the LQR Problem with the affine dynamics

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \boldsymbol{\Phi}\mathbf{x}_k + \boldsymbol{\Gamma}\mathbf{u}_k + \mathbf{d} \tag{3.103}$$

and the cost

$$\begin{aligned} \bar{l}_N(\mathbf{x}_N) &= l_N + \mathbf{l}_{\mathbf{x},N}^{\mathrm{T}}\mathbf{x}_N + \frac{1}{2}\mathbf{x}_N^{\mathrm{T}}\mathbf{L}_{\mathbf{xx},N}\mathbf{x}_N \\ \bar{l}_k(\mathbf{x}_k, \mathbf{u}_k) &= l_k + \mathbf{l}_{\mathbf{x},k}^{\mathrm{T}}\mathbf{x}_k + \mathbf{l}_{\mathbf{u},k}^{\mathrm{T}}\mathbf{u}_k + \frac{1}{2}\mathbf{x}_k^{\mathrm{T}}\mathbf{L}_{\mathbf{xx},k}\mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^{\mathrm{T}}\mathbf{L}_{\mathbf{uu},k}\mathbf{u}_k + \mathbf{x}_k^{\mathrm{T}}\mathbf{L}_{\mathbf{xu},k}\mathbf{u}_k \ . \end{aligned} \tag{3.104}$$

Assume that the optimal value function at time $k+1$ has the form $V_{k+1}^*(\mathbf{x}_{k+1}) = v_{k+1} + \mathbf{v}_{\mathbf{x},k+1}^{\mathrm{T}}\mathbf{x}_{k+1} + \frac{1}{2}\mathbf{x}_k^{\mathrm{T}}\mathbf{V}_{\mathbf{xx},k+1}\mathbf{x}_k$, which is obviously true for $k+1 = N$ with $v_N = l_N$, $\mathbf{v}_{\mathbf{x},N} = \mathbf{l}_{\mathbf{x},N}$ and $\mathbf{V}_{\mathbf{xx},N} = \mathbf{L}_{\mathbf{xu},N}$. Using Theorem 3.2, the optimal action-value function then evaluates to

$$\begin{aligned} Q_k^*(\mathbf{x}_k,\mathbf{u}_k) = &\underbrace{l_k + \mathbf{l}_{\mathbf{x},k}^{\mathrm{T}}\mathbf{x}_k + \mathbf{l}_{\mathbf{u},k}^{\mathrm{T}}\mathbf{u}_k + \frac{1}{2}\mathbf{x}_k^{\mathrm{T}}\mathbf{L}_{\mathbf{xx},k}\mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^{\mathrm{T}}\mathbf{L}_{\mathbf{uu},k}\mathbf{u}_k + \mathbf{x}_k^{\mathrm{T}}\mathbf{L}_{\mathbf{xu},k}\mathbf{u}_k}_{\bar{l}_k(\mathbf{x}_k,\mathbf{u}_k)} + \\ &\underbrace{V_{k+1} + \mathbf{v}_{\mathbf{x},k+1}^{\mathrm{T}}(\boldsymbol{\Phi}\mathbf{x}_k + \boldsymbol{\Gamma}\mathbf{u}_k + \mathbf{d}) + \frac{1}{2}(\boldsymbol{\Phi}\mathbf{x}_k + \boldsymbol{\Gamma}\mathbf{u}_k + \mathbf{d})^{\mathrm{T}}\mathbf{V}_{\mathbf{xx},k+1}(\boldsymbol{\Phi}\mathbf{x}_k + \boldsymbol{\Gamma}\mathbf{u}_k + \mathbf{d})}_{V_{k+1}^*\left(\mathbf{f}(\mathbf{x}_k,\mathbf{u}_k)\right)} \ . \end{aligned} \tag{3.105}$$

Hence, the optimal action-value function (3.105) can be rewritten as a quadratic form

$$Q_k^*(\mathbf{x}_k, \mathbf{u}_k) = q_k + \mathbf{q}_{\mathbf{x},k}^{\mathrm{T}}\mathbf{x}_k + \mathbf{q}_{\mathbf{u},k}^{\mathrm{T}}\mathbf{u}_k + \frac{1}{2}\mathbf{x}_k^{\mathrm{T}}\mathbf{Q}_{\mathbf{xx},k}\mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^{\mathrm{T}}\mathbf{Q}_{\mathbf{uu},k}\mathbf{u}_k + \mathbf{x}_k^{\mathrm{T}}\mathbf{Q}_{\mathbf{xu},k}\mathbf{u}_k \ , \tag{3.106a}$$

with

$$q_k = l_k + V_{k+1} + \mathbf{d}^{\mathrm{T}}\mathbf{v}_{\mathbf{x},k+1} + \frac{1}{2}\mathbf{d}^{\mathrm{T}}\mathbf{V}_{\mathbf{xx},k+1}\mathbf{d} \tag{3.106b}$$

$$\mathbf{q}_{\mathbf{x},k} = \mathbf{l}_{\mathbf{x},k} + \mathbf{\Phi}^{\mathrm{T}}\mathbf{v}_{\mathbf{x},k+1} + \mathbf{\Phi}^{\mathrm{T}}\mathbf{V}_{\mathbf{xx},k+1}\mathbf{d} \tag{3.106c}$$

$$\mathbf{q}_{\mathbf{u},k} = \mathbf{l}_{\mathbf{u},k} + \mathbf{\Gamma}^{\mathrm{T}}\mathbf{v}_{\mathbf{x},k+1} + \mathbf{\Gamma}^{\mathrm{T}}\mathbf{V}_{\mathbf{xx},k+1}\mathbf{d} \tag{3.106d}$$

$$\mathbf{Q}_{\mathbf{xx},k} = \mathbf{L}_{\mathbf{xx},k} + \mathbf{\Phi}^{\mathrm{T}}\mathbf{V}_{\mathbf{xx},k+1}\mathbf{\Phi} \tag{3.106e}$$

$$\mathbf{Q}_{\mathbf{uu},k} = \mathbf{L}_{\mathbf{uu},k} + \mathbf{\Gamma}^{\mathrm{T}}\mathbf{V}_{\mathbf{xx},k+1}\mathbf{\Gamma} \tag{3.106f}$$

$$\mathbf{Q}_{\mathbf{xu},k} = \mathbf{L}_{\mathbf{xu},k} + \mathbf{\Phi}^{\mathrm{T}}\mathbf{V}_{\mathbf{xx},k+1}\mathbf{\Gamma}\ . \tag{3.106g}$$

From Theorem 3.2, the optimal policy can be obtained by minimizing (3.106), which yields

$$\mathbf{u}_k^* = \boldsymbol{\mu}_k^*(\mathbf{x}_k) = \underbrace{-\mathbf{Q}_{\mathbf{uu},k}^{-1}\mathbf{Q}_{\mathbf{xu},k}^{\mathrm{T}}}_{=\mathbf{K}_k}\mathbf{x}_k \underbrace{-\mathbf{Q}_{\mathbf{uu},k}^{-1}\mathbf{q}_{\mathbf{u},k}}_{=\mathbf{k}_k} = \mathbf{K}_k\mathbf{x}_k + \mathbf{k}_k\ . \tag{3.107}$$

Theorem 3.2 allow to compute the optimal state-value function as

$$V_k^*(\mathbf{x}_k) = \mathbf{Q}_k^*(\mathbf{x}_k, \boldsymbol{\mu}_k^*(\mathbf{x}_k)) = v_k + \mathbf{v}_{\mathbf{x},k}^{\mathrm{T}}\mathbf{x}_k + \frac{1}{2}\mathbf{x}_k^{\mathrm{T}}\mathbf{V}_{\mathbf{xx},k}\mathbf{x}_k\ , \tag{3.108a}$$

with

$$v_k = q_k + \mathbf{k}_k^{\mathrm{T}}\mathbf{q}_{\mathbf{u},k} + \frac{1}{2}\mathbf{k}_k^{\mathrm{T}}\mathbf{Q}_{\mathbf{uu},k}\mathbf{k}_k \tag{3.108b}$$

$$\mathbf{v}_{\mathbf{x},k} = \mathbf{q}_{\mathbf{x},k} + \mathbf{K}_k^{\mathrm{T}}\mathbf{q}_{\mathbf{u},k} + \mathbf{K}_k^{\mathrm{T}}\mathbf{Q}_{\mathbf{uu},k}\mathbf{k}_k + \mathbf{Q}_{\mathbf{xu},k}\mathbf{k}_k \tag{3.108c}$$

$$\mathbf{V}_{\mathbf{xx},k} = \mathbf{Q}_{\mathbf{xx},k} + \mathbf{K}_k^{\mathrm{T}}\mathbf{Q}_{\mathbf{uu},k}\mathbf{K}_k + \mathbf{Q}_{\mathbf{xu},k}\mathbf{K}_k + \mathbf{K}_k^{\mathrm{T}}\mathbf{Q}_{\mathbf{xu},k}^{\mathrm{T}}\ . \tag{3.108d}$$

Hence, $V_k^*(\mathbf{x}_k)$ is again a quadratic function of $\mathbf{x}_k$ and we can repeat until $k = 0$.

iLQR - the finite horizon nonlinear case

The *iterative linear-quadratic regulator (iLQR)* is an algorithm that uses the idea of the LQR algorithm to solve general nonlinear optimal control problems. Given a general nonlinear dynamics $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ with an initial rollout $\bar{\boldsymbol{\tau}} = (\bar{\mathbf{x}}_0, \bar{\mathbf{u}}_0, \ldots, \bar{\mathbf{u}}_{N-1}, \bar{\mathbf{x}}_N$ we can linearize the dynamics around small increments $\mathbf{x}_k = \bar{\mathbf{x}}_k + \Delta\mathbf{x}_k$ and $\mathbf{u}_k = \bar{\mathbf{u}}_k + \Delta\mathbf{u}_k$, which yields

$$\Delta\mathbf{x}_{k+1} = \underbrace{\mathbf{f}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k) - \bar{\mathbf{x}}_{k+1}}_{\mathbf{d}_k} + \mathbf{\Phi}_k\Delta\mathbf{x}_k + \mathbf{\Gamma}_k\Delta\mathbf{u}_k \tag{3.109}$$

with $\mathbf{\Phi}_k = \frac{\partial \mathbf{f}}{\partial \mathbf{x}_k}\big|_{\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k}$ and $\mathbf{\Gamma}_k = \frac{\partial \mathbf{f}}{\partial \mathbf{u}_k}\big|_{\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k}$.

Similarly, we can use a quadratic expansions of the nonlinear costs $\bar{l}_N(\mathbf{x}_N)$ and $\bar{l}_k(\mathbf{x}_k, \mathbf{u}_k)$, yielding

$$\begin{aligned}
\bar{l}_N(\mathbf{x}_N) &\approx \bar{l}_N(\bar{\mathbf{x}}_N) + \mathbf{l}_{\mathbf{x},N}^{\mathrm{T}}\Delta\mathbf{x}_N + \frac{1}{2}\Delta\mathbf{x}_N^{\mathrm{T}}\mathbf{L}_{\mathbf{xx},N}\Delta\mathbf{x}_N \\
\bar{l}_k(\mathbf{x}_k, \mathbf{u}_k) &\approx \bar{l}_k(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k) + \mathbf{l}_{\mathbf{x},k}^{\mathrm{T}}\Delta\mathbf{x}_k + \mathbf{l}_{\mathbf{u},k}^{\mathrm{T}}\Delta\mathbf{u}_k \\
&\quad + \frac{1}{2}\Delta\mathbf{x}_k^{\mathrm{T}}\mathbf{L}_{\mathbf{xx},k}\Delta\mathbf{x}_k + \frac{1}{2}\Delta\mathbf{u}_k^{\mathrm{T}}\mathbf{L}_{\mathbf{uu},k}\Delta\mathbf{u}_k + \Delta\mathbf{x}_k^{\mathrm{T}}\mathbf{L}_{\mathbf{xu},k}\Delta\mathbf{u}_k\ ,
\end{aligned} \tag{3.110}$$

where $\mathbf{l}_{(\cdot),k} = \nabla_{(\cdot)}\bar{l}_k\big|_{\bar{\mathbf{x}}_k,\bar{\mathbf{u}}_k}$, $\mathbf{l}_{\mathbf{x},N} = \nabla_{\mathbf{x}_N}\bar{l}_N\big|_{\bar{\mathbf{x}}_N}$ are the gradients and $\mathbf{L}_{(\cdot),(\cdot),k} = \nabla_{(\cdot)(\cdot)}\bar{l}_k\big|_{\bar{\mathbf{x}}_k,\bar{\mathbf{u}}_k}$, $\mathbf{L}_{\mathbf{xx},N} = \nabla^2_{\mathbf{x}_N}\bar{l}_N\big|_{\bar{\mathbf{x}}_N}$ are the Hessians, all evaluated at the initial solution $\bar{\boldsymbol{\tau}}$. Since (3.109) and (3.110) share the same structure in $\Delta\mathbf{x}_k$ and $\Delta\mathbf{u}_k$ as (3.104), we can use the algorithm to compute an optimal increment

$$\Delta\mathbf{u}_k^* = \mathbf{K}_k\Delta\mathbf{x}_k + \mathbf{k}_k \tag{3.111}$$

with $\mathbf{K}_k$ and $\mathbf{k}_k$ as defined in (3.107).

Thus the algorithm contains two steps

- **Backward pass:** Here we use our linear system approximation and the quadratic cost approximation to compute quadratic approximations of the state- and action-value functions (3.108) and (3.106) This is performed in a backward recursion, i.e. we stat at $k+1 = N$ and iterate backward in time, see (3.108) and (3.106). Note, that this step can be interpreted as *policy evaluation*.

- **Forward pass:** Here we compute the new rollout $\boldsymbol{\tau}$, based on the updated controls (3.111), as well as the new cost, gradients and Hessians (3.109) and (3.110). Thus, this step can be interpreted as *policy improvement*.

  Two different strategies to compute the forward pass can be found in the literature:

  1. We can assume that the initial states $(\bar{\mathbf{x}}_0, \dots, \bar{\mathbf{x}}_N)$ are obtained by integrating the dynamics $\mathbf{f}$ along the initial control sequence $(\bar{\mathbf{u}}_0, \dots, \bar{\mathbf{u}}_{N-1})$. Beginning at $\mathbf{x}_0 = \bar{\mathbf{x}}_0$, we can use $\Delta\mathbf{x}_k = \mathbf{x}_k - \bar{\mathbf{x}}_k$ to compute $\mathbf{u}_k = \mathbf{K}_k\Delta\mathbf{x}_k + \mathbf{k}_k$ to further compute the new state $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ and then proceed with $\Delta\mathbf{x}_{k+1} = \mathbf{x}_{k+1} - \bar{\mathbf{x}}_{k+1}$. This corresponds to a *single shooting* version, which is what is commonly referred as the iLQR algorithm in the literature. Note that $\mathbf{d}_k = \mathbf{0}$ always holds in this case, since the dynamics $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ is always satisfied.

  2. Starting at $\Delta\mathbf{x}_0 = \mathbf{0}$, we can compute for a given $\Delta\mathbf{x}_k$ the control increment $\Delta\mathbf{u}_k = \mathbf{K}_k\Delta\mathbf{x}_k + \mathbf{k}_k$ and then use the linearized dynamics $\Delta\mathbf{x}_{k+1} = \mathbf{d}_k + \boldsymbol{\Phi}_k\Delta\mathbf{x}_k + \boldsymbol{\Gamma}_k\Delta\mathbf{u}_k$ to compute the next state increment which is further used to compute $\Delta\mathbf{u}_{k+1}$. The new state sequence is then obtained by $\mathbf{x}_k = \bar{\mathbf{x}}_k + \Delta\mathbf{x}_k$. This corresponds to a *multiple shooting* version of the iLQR algorithm, which is typically called Gauss-Newton multiple shooting (GNMS) in the literature. Note, that $\mathbf{d}_k \neq \mathbf{0}$ in general (until convergence).

  In both cases, we use the initial guess $\bar{\boldsymbol{\tau}}$ and the control increments $\Delta\mathbf{u}_k$, $k = 0, \dots, N-1$ to obtain a new solution $\boldsymbol{\tau} = (\mathbf{x}_0, \mathbf{u}_0, \dots, \mathbf{u}_{N-1}, \mathbf{x}_N)$. We then set $\bar{\boldsymbol{\tau}} \leftarrow \boldsymbol{\tau}$ and repeat until convergence. The two strategies can be combines too, which is then referred to as iLQR-GNMS.

Algorithm 3: Unconstrained iLQR and GNMS

while $\Delta J > \varepsilon$ **do**
  /* Terminal conditions */
  $\mathbf{v_x} \leftarrow \mathbf{l_{x},}_N$;
  $\mathbf{V_{xx}} \leftarrow \mathbf{L_{xx},}_N$;
  /* Backward Pass */
  for $k = N-1, \ldots, 0$ **do**
    $\mathbf{q_x} \leftarrow \mathbf{l_{x},}_k + \mathbf{\Phi}_k^\top \mathbf{v_x} + \mathbf{\Phi}_k^\top \mathbf{V_{xx}} \mathbf{d}_k$;
    $\mathbf{q_u} \leftarrow \mathbf{l_{u},}_k + \mathbf{\Gamma}_k^\top \mathbf{v_x} + \mathbf{\Phi}_k^\top \mathbf{V_{xx}} \mathbf{d}_k$;
    $\mathbf{Q_{xx}} \leftarrow \mathbf{L_{xx}} + \mathbf{\Phi}_k^\top \mathbf{V_{xx}} \mathbf{\Phi}_k$;
    $\mathbf{Q_{uu}} \leftarrow \mathbf{L_{uu}} + \mathbf{\Gamma}_k^\top \mathbf{V_{xx}} \mathbf{\Gamma}_k$;
    $\mathbf{Q_{xu}} \leftarrow \mathbf{L_{xu}} + \mathbf{\Phi}_k^\top \mathbf{P}_{k+1} \mathbf{\Gamma}_k$;
    $\mathbf{k}_k \leftarrow -\mathbf{Q_{uu}}^{-1} \mathbf{q_u}$;
    $\mathbf{K}_k \leftarrow -\mathbf{Q_{uu}}^{-1} \mathbf{Q_{ux}}$;
    $\mathbf{v_x} \leftarrow \mathbf{q_x} + \mathbf{K}_k^\top \mathbf{q_u} + \mathbf{K}_k^\top \mathbf{Q_{uu}} \mathbf{k}_k + \mathbf{Q_{xu}} \mathbf{k}_k$;
    $\mathbf{V_{xx}} \leftarrow \mathbf{Q_{xx}} + \mathbf{K}_k^\top \mathbf{Q_{uu}} \mathbf{K}_k + \mathbf{Q_{xu}} \mathbf{K}_k + \mathbf{K}_k^\top \mathbf{Q_{xu}}^\top$;
  end
  /* Forward Pass (iLQR case) */
  $\mathbf{x}_0 = \bar{\mathbf{x}}_0$ **for** $k = 0, \ldots, N-1$ **do**
    $\Delta \mathbf{u}_k \leftarrow \mathbf{k}_k + \mathbf{K}_k \Delta \mathbf{x}_k$;
    $\mathbf{u}_k \leftarrow \bar{\mathbf{u}}_k + \Delta \mathbf{u}_k$;
    $\mathbf{x}_{k+1} \leftarrow \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$;
    $\Delta \mathbf{x}_{k+1} \leftarrow \mathbf{x}_{k+1} - \bar{\mathbf{x}}_{k+1}$;
    $\mathbf{d}_k = \mathbf{0}$
  end
  /* Forward Pass (GNMS case) */
  $\Delta \mathbf{x}_0 \leftarrow \mathbf{0}$;
  for $k = 0, \ldots, N-1$ **do**
    $\Delta \mathbf{u}_k \leftarrow \mathbf{k}_k + \mathbf{K}_k \Delta \mathbf{x}_k$;
    $\Delta \mathbf{x}_{k+1} \leftarrow \mathbf{d}_k + \mathbf{\Phi}_k \Delta \mathbf{x}_k + \mathbf{\Gamma}_k \Delta \mathbf{u}_k$;
    $\mathbf{u}_k \leftarrow \bar{\mathbf{u}}_k + \Delta \mathbf{u}_k$;
    $\mathbf{x}_{k+1} = \bar{\mathbf{x}}_{k+1} + \Delta \mathbf{x}_{k+1}$;
    $\mathbf{d}_k = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}_{k+1}$
  end
  /* Update sequences */
  $(\bar{\mathbf{x}}_k)_{k=0}^{N} \leftarrow (\mathbf{x}_k)_{k=0}^{N}$;
  $(\bar{\mathbf{u}}_k)_{k=0}^{N-1} \leftarrow (\mathbf{u}_k)_{k=0}^{N-1}$;
end
return $(\mathbf{u}_k^*)_{k=0}^{N-1}$, $(\mathbf{x}_k^*)_{k=0}^{N}$

3.3.3 Sampling Based Model Predictive Control

While model predictive control (MPC) methods based on gradient-based OCP solvers are great tools for a variety of applications, they show two major drawbacks:

1. They suffer from the potential of getting stuck in local minima. In robotics this is often the case in collision avoidance applications.
2. The system dynamics as well as the costs have to be differentiable.

Sampling based MPC methods rely on sampling instead of gradients for policy updates, which lossens these limitations to some extend. While there is a huge amount of work on sampling based MPC methods, the basic idea is very similar for all of them. This section deals with the variant presented in [3.12].

In the following, we consider a deterministic dynamic function

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{3.112}$$

and the goal is to find an optimal stochastic open-loop policy of the form

$$\mathbf{u}_k \sim \pi_{\boldsymbol{\theta}_k}(\mathbf{u}_k) = \mathcal{N}(\mathbf{u}_k; \boldsymbol{\theta}_k, \boldsymbol{\Sigma}) = \eta^{-1} \exp\left((\mathbf{u}_k - \boldsymbol{\theta}_k)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{u}_k - \boldsymbol{\theta}_k)\right) , \tag{3.113}$$

with the normalization $\eta = \left((2\pi)^m |\boldsymbol{\Sigma}|\right)^{\frac{1}{2}}$. Given a sequence of inputs $\mathbf{U} = (\mathbf{u}_0, \dots, \mathbf{u}_{N-1})$ with the corresponding mean values $\boldsymbol{\Theta} = (\boldsymbol{\theta}_0, \dots, \boldsymbol{\theta}_{N-1})$ we can define the joint distribution as

$$\Pi_{\boldsymbol{\Theta}}(\mathbf{U}) = \prod_{k=0}^{N-1} \pi_{\boldsymbol{\theta}_k}(\mathbf{u}_k) \ . \tag{3.114}$$

The goal is to solve a stochastic optimal control problem of the form

$$\begin{aligned} &\min_{\boldsymbol{\Theta} \in \mathcal{U}^N} \mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}\{\mathsf{L}_0\} \\ &\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k) \ , \quad \mathbf{x}_0 \sim \mathsf{p}_{\mathsf{x}_0}(\mathbf{x}_0) \ . \end{aligned} \tag{3.115}$$

with

$$\mathsf{L}_0 = l_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} l_k(\mathbf{x}_k) + \frac{\lambda}{2} (\boldsymbol{\theta}_k - \bar{\boldsymbol{\theta}}_k)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{\theta}_k - \bar{\boldsymbol{\theta}}_k) \ . \tag{3.116}$$

If we define a *base distribution* $\Pi_{\bar{\boldsymbol{\Theta}}}(\mathbf{U})$ with $\bar{\boldsymbol{\Theta}} = (\bar{\boldsymbol{\theta}}_0, \dots, \bar{\boldsymbol{\theta}}_{N-1})$ and the state dependend costs $\mathsf{S} = S(\mathbf{U}) = l_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} l_k(\mathbf{x}_k)$, we can rewrite the costs of (3.81)

$$\mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}[\mathsf{L}_0] = \mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}[\mathsf{S}] + \lambda D_{\mathrm{KL}}(\Pi_{\boldsymbol{\Theta}} \parallel \Pi_{\bar{\boldsymbol{\Theta}}}) \ , \tag{3.117}$$

where D_{KL} is the Kullback-Leibler divergence, a similarity measure for two distributions. The result (3.117) suggests that we can interpret the costs as one part $\mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}[\mathsf{S}]$ that reduces the state costs and one part $D_{\mathrm{KL}}(\Pi_{\boldsymbol{\Theta}} \parallel \Pi_{\bar{\boldsymbol{\Theta}}})$ that tries to keep the policy close to the base policy $\Pi_{\bar{\boldsymbol{\Theta}}}$ (often $\bar{\boldsymbol{\Theta}} = \mathbf{0}$).

Free Energy Lower Bound

The *free energy* of a control system under the base policy $\Pi_{\bar{\boldsymbol{\Theta}}}$ is defined as

$$\mathcal{F} = \log\left(\mathbb{E}_{\Pi_{\bar{\boldsymbol{\Theta}}}}\left[\exp\left(-\frac{1}{\lambda}\mathsf{S}\right)\right]\right), \tag{3.118}$$

with the inverse temperatur parameter $\lambda > 0$ and some base distribution Π_b. With importance sampling, we can reformulate (3.118) as

$$\mathcal{F} = \log\left(\mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}\left[\exp\left(-\frac{1}{\lambda}\mathsf{S}\right)\frac{\Pi_{\bar{\boldsymbol{\Theta}}}(\mathbf{U})}{\Pi_{\boldsymbol{\Theta}}(\mathbf{U})}\right]\right), \tag{3.119}$$

and the concavity of the logarithm allows to apply Jensen's inequality

$$\begin{aligned}\mathcal{F} &\geq \mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}\left[\log\left(\exp\left(-\frac{1}{\lambda}\mathsf{S}\right)\frac{\Pi_{\bar{\boldsymbol{\Theta}}}(\mathbf{U})}{\Pi_{\boldsymbol{\Theta}}(\mathbf{U})}\right)\right] \\ &= -\frac{1}{\lambda}\mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}\left[\mathsf{S} + \lambda\log\left(\frac{\Pi_{\boldsymbol{\Theta}}(\mathbf{U})}{\Pi_{\bar{\boldsymbol{\Theta}}}(\mathbf{U})}\right)\right] \\ &= -\frac{1}{\lambda}\left(\mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}[\mathsf{S}] + \lambda D_{\mathrm{KL}}(\Pi_{\boldsymbol{\Theta}} \parallel \Pi_{\bar{\boldsymbol{\Theta}}})\right).\end{aligned} \tag{3.120}$$

Finally, this yields the free energy lower bound

$$-\lambda\mathcal{F} \leq \mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}[\mathsf{S}] + \lambda D_{\mathrm{KL}}(\Pi_{\boldsymbol{\Theta}} \parallel \Pi_{\bar{\boldsymbol{\Theta}}}) \tag{3.121}$$

for the stochastic OCP (3.115).

Optimal Policy

Consider the policy

$$\begin{aligned}\Pi^*(\mathbf{U}) &= \frac{1}{\eta}\exp\left(-\frac{1}{\lambda}\mathsf{S}\right)\Pi_{\bar{\boldsymbol{\Theta}}}(\mathbf{U}) \\ \eta &= \int_{\mathbb{R}^{m\times N}}\exp\left(-\frac{1}{\lambda}\mathsf{S}\right)\Pi_{\bar{\boldsymbol{\Theta}}}(\mathbf{U})\,\mathrm{d}\mathbf{U} = \mathbb{E}_{\Pi_{\bar{\boldsymbol{\Theta}}}}\left[\exp\left(-\frac{1}{\lambda}\mathsf{S}\right)\right].\end{aligned} \tag{3.122}$$

The KL divergence w.r.t the base policy yields

$$D_{\mathrm{KL}}(\Pi^* \parallel \Pi_{\bar{\boldsymbol{\Theta}}}) = \mathbb{E}_{\Pi^*}\left[\log\left(\frac{\Pi^*}{\Pi_{\bar{\boldsymbol{\Theta}}}}\right)\right] = -\log(\eta) - \frac{1}{\lambda}\mathbb{E}_{\Pi^*}[\mathsf{S}]\,. \tag{3.123}$$

Substituting into the right hand side of (3.121) and using the definition (3.122) of η yields

$$\mathbb{E}_{\Pi^*}[\mathsf{S}] - \mathbb{E}_{\Pi^*}[\mathsf{S}] - \lambda\log(\eta) = -\lambda\log\left(\mathbb{E}_{\Pi_{\bar{\boldsymbol{\Theta}}}}\left[\exp\left(-\frac{1}{\lambda}\mathsf{S}\right)\right]\right) = -\lambda\mathcal{F}\,. \tag{3.124}$$

This illustrates, that (3.122) is indeed optimal, since the overall costs are equal to the lower limit.

Minimization

The goal is now to align the policy $\Pi_{\boldsymbol{\Theta}}$ with the optimal distribution Π^*. This can be achieved my minimizing the KL-Divergence

$$\boldsymbol{\Theta}^* = \arg\min_{\Theta \in \mathcal{U}^N} [D_{\mathrm{KL}}(\Pi^* \parallel \Pi_{\boldsymbol{\Theta}})] \,, \tag{3.125}$$

where the KL-Divergence evaluates to

$$\begin{aligned} D_{\mathrm{KL}}(\Pi^* \parallel \Pi_{\boldsymbol{\Theta}}) &= \mathbb{E}_{\Pi^*}\left[\log\left(\frac{\Pi^*(\mathbf{U})}{\Pi_{\boldsymbol{\Theta}}(\mathbf{U})}\right)\right] \\ &= \mathbb{E}_{\Pi^*}\Big[\log\left(\Pi^*(\mathbf{U})\right)\Big] - \mathbb{E}_{\Pi^*}\Big[\log\left(\Pi_{\boldsymbol{\Theta}}(\mathbf{U})\right)\Big] \\ &= \mathbb{E}_{\Pi^*}\Big[\log\left(\Pi^*(\mathbf{U})\right)\Big] - \mathbb{E}_{\Pi^*}\left[\log(\eta^{-N}) - \frac{1}{2}\sum_{k=0}^{N-1}(\mathbf{u}_k - \boldsymbol{\theta}_k)^{\mathrm{T}}\Sigma^{-1}(\mathbf{u}_k - \boldsymbol{\theta}_k)\right] . \end{aligned} \tag{3.126}$$

Since Π^* is independent of $\boldsymbol{\Theta}$, this results in

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta} \in \mathcal{U}^N} [D_{\mathrm{KL}}(\Pi^* \parallel \Pi_{\boldsymbol{\Theta}})] = \arg\min_{\boldsymbol{\Theta} \in \mathcal{U}^N} \mathbb{E}_{\Pi^*}\left[\frac{1}{2}\sum_{k=0}^{N-1}(\mathbf{u}_k - \boldsymbol{\theta}_k)^{\mathrm{T}}\Sigma^{-1}(\mathbf{u}_k - \boldsymbol{\theta}_k)\right] , \tag{3.127}$$

which yields the optimal solution

$$\boldsymbol{\theta}_k^* = \mathbb{E}_{\Pi^*}[\mathbf{u}_k] \,. \tag{3.128}$$

Importance sampling

The problem with the solution (3.128) is that it is neither possible to analytically compute the expectation over the optimal policy, nor can we directly sample from it. However, importance sampling allows us to compute a set of samples. This yields

$$\mathbb{E}_{\Pi^*} = \mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}[w(\mathbf{U})\mathbf{u}_k] \qquad \text{with} \qquad w(\mathbf{U}) = \frac{\Pi^*(\mathbf{U})}{\Pi_{\boldsymbol{\Theta}}(\mathbf{U})} \,. \tag{3.129}$$

We can expand the expression $w(\mathbf{U})$ as

$$w(\mathbf{U}) = \frac{\Pi^*(\mathbf{U})}{\Pi_{\bar{\boldsymbol{\Theta}}}(\mathbf{U})}\frac{\Pi_{\bar{\boldsymbol{\Theta}}}(\mathbf{U})}{\Pi_{\boldsymbol{\Theta}}(\mathbf{U})} = \frac{1}{\eta}\exp\left(-\frac{1}{\lambda}\mathsf{S}\right)\frac{\Pi_{\bar{\boldsymbol{\Theta}}}(\mathbf{U})}{\Pi_{\boldsymbol{\Theta}}(\mathbf{U})} \tag{3.130}$$

and applying importance sampling to η yields

$$\eta = \mathbb{E}_{\Pi_{\bar{\boldsymbol{\Theta}}}}\left[\exp\left(-\frac{1}{\lambda}\mathsf{S}\right)\right] = \mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}\left[\exp\left(-\frac{1}{\lambda}\mathsf{S}\right)\frac{\Pi_{\bar{\boldsymbol{\Theta}}}}{\Pi_{\boldsymbol{\Theta}}}\right] \tag{3.131}$$

with the fraction

$$\begin{aligned}\frac{\Pi_{\bar{\boldsymbol{\Theta}}}}{\Pi_{\boldsymbol{\Theta}}} &= \exp\left(-\frac{1}{2}\sum_{k=0}^{N-1} d_k + 2(\boldsymbol{\theta}_k - \bar{\boldsymbol{\theta}}_k)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{u}_k\right) \\ &= \underbrace{\exp\left(-\frac{1}{2}\sum_{k=0}^{N-1} d_k\right)}_{D_k} \exp\left(-\frac{1}{2}\sum_{k=0}^{N-1}(\boldsymbol{\theta}_k - \bar{\boldsymbol{\theta}}_k)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{u}_k\right) \\ d_k &= \bar{\boldsymbol{\theta}}_k^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\theta}_k\end{aligned} \tag{3.132}$$

occuring at both, the nominator and the denominator in (3.130). Since D_k does not depend on $\mathbf{U}$, we can factor it outside of the expectation and it cancels out.

In summary, this yields

$$\begin{aligned}w(\mathbf{U}) &= \frac{\exp\left(-\frac{1}{\lambda}J(\mathbf{U})\right)}{\mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}\left[\exp\left(-\frac{1}{\lambda}J(\mathbf{U})\right)\right]} \\ J(\mathbf{U}) &= \mathsf{S} + \lambda\sum_{k=0}^{N-1}(\boldsymbol{\theta}_k - \bar{\boldsymbol{\theta}}_k)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{u}_k\ .\end{aligned} \tag{3.133}$$

Implementation

Algorithm 4 summarizes the main algorithm. To obtain M realizations $\mathbf{U}^i$, $i = 0, \ldots, M-1$ of the actual policy $\Pi_{\boldsymbol{\Theta}}$, we sample $\mathcal{E}^i = [\boldsymbol{\varepsilon}_0^i, \ldots, \boldsymbol{\varepsilon}_{N-1}^i]^{\mathrm{T}}$ with $\boldsymbol{\varepsilon}_k^i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ and compute $\mathbf{U}^i = \boldsymbol{\Theta} + \mathcal{E}$. Then we can compute the costs $J^i = J(\mathbf{U}^i)$ and approximate

$$\eta = \mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}\left[\exp\left(-\frac{1}{\lambda}J(\mathbf{U})\right)\right] \approx \frac{1}{M}\sum_{i=0}^{M-1}\exp\left(-\frac{1}{\lambda}J^i\right) \tag{3.134}$$

This further allows to compute the weights $w_i = \frac{\exp(-\frac{1}{\lambda}J^i)}{\eta}$, which are then used to compute approximate the expectation

$$\boldsymbol{\theta}_k = \mathbb{E}_{\Pi_{\boldsymbol{\Theta}}}[w(\mathbf{U})\mathbf{u}_k] \approx \frac{1}{M}\sum_{i=0}^{M-1} w^i\mathbf{u}_k^i\ . \tag{3.135}$$

Algorithm 4: Sampling Based MPC

while *Not done* **do**
  for $i = 0, \dots, M-1$ **do**
    /* Sampling */
    Sample $\mathcal{E}^i = [\boldsymbol{\varepsilon}_0^i, \dots, \boldsymbol{\varepsilon}_{N-1}^i]^{\mathrm{T}}$
    /* Compute realizations of control sequences */
    $\mathbf{U}^i = \boldsymbol{\Theta} + \mathcal{E}^i$ /* Compute costs */
    Compute $J^i = S(\mathbf{U}^i) + \lambda \sum_{k=0}^{N-1} (\boldsymbol{\theta}_k - \bar{\boldsymbol{\theta}}_k)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \mathbf{u}_k^i$
  end
  $\eta = \frac{1}{M} \sum_{i=0}^{M-1} \exp\Big(-\frac{1}{\lambda} J^i \Big)$
  /* Compute weights */
  for $i = 0, \dots, M-1$ **do**
    $w^i = \frac{1}{\eta} \exp\Big(-\frac{1}{\lambda} J^i \Big)$
  end
  /* Update control mean */
  $\boldsymbol{\Theta} = \frac{1}{M} \sum_{i=0}^{M-1} w^i \mathbf{U}^i$
  /* Apply control */
  Apply($\boldsymbol{\theta}_0$) /* Shift control */
  for $k = 1, \dots, N-1$ **do**
    $\boldsymbol{\theta}_{k-1} = \boldsymbol{\theta}_k$
  end
end

3.4 Literatur

[3.1] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.

[3.2] M. P. Deisenroth, A. Faisal, and C. S. Ong, *Mathematics for Machine Learning*. Cambridge University Press, 2020. [Online]. Available: https://mml-book.github.io/.

[3.3] E. T. Jaynes, *Probability Theory: The Logic of Science*. Cambridge University Press, 2013, vol. 1.

[3.4] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2005, vol. 1.

[3.5] M. L. Puterman, *Markov Decision Processes*. John Wiley & Sons, 2005.

[3.6] C. Szepesvari, *Algorithms for Reinforcement Learning*. Morgan & Claypool, 2009.

[3.7] R. S. Sutton and A. G. Barto, *Reinforcement Learning*. MIT Press, 2018.

[3.8] M. Sugiyama, *Statistical Reinforcment Learning*. CRC Press, 2015.

[3.9] D. Silver, *Reinforcement learning: An introduction*, 2015. [Online]. Available: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html.

[3.10] F. L. Lewis, D. L. Varbie, and V. Syrmos, *Optimal Control*. John Wiley & Sons, 2012.

[3.11] M. Vidyasagar, "A tutorial introduction to reinforcement learning," 2023. [Online]. Available: https://arxiv.org/abs/2304.00803v1.

[3.12] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[3.13] C. M. Bischop, *Pattern Recognition and Machine Learning*. Springer, 2006.