Learning-based Optimal Control of Constrained Switched Linear Systems using Neural Networks

Lukas Markolf[a] and Olaf Stursberg[b]
Control and System Theory, Dept. of Electrical Engineering and Computer Science, University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany

Keywords: Neural Networks, Intelligent Control, Hybrid Systems, Approximate Dynamic Programming.

Abstract: This work considers (deep) artificial feed-forward neural networks as parametric approximators in optimal control of discrete-time switched linear systems with controlled switching. The proposed approach is based on approximate dynamic programming and allows the fast computation of (sub-)optimal discrete and continuous control inputs, either by approximating the optimal cost-to-go functions or by approximating the optimal discrete and continuous input policies. An important property of the approach is the satisfaction of polytopic state and input constraints, which is crucial for ensuring safety, as required in many control applications. A numeric example is provided for illustration and evaluation of the approaches.

1 INTRODUCTION

In many applications, continuous and discrete controls coexist as, e.g., in all production or processing systems which are equipped with continuous feedback controllers and supervisory controllers. Typically, the two types of controllers are considered and designed separately, not only to split the corresponding functions, but also to simplify the design task. The separate design, however, may lead to degraded performance if the two parts lead to opposing effects on the plant at the same time. This motivates to investigate techniques that optimize continuous and discrete controls simultaneously. This paper considers the design of optimizing feedback controllers for discrete-time switched linear systems (SLS). Such systems, which constitute a special class of hybrid systems (Branicky et al., 1998), allow to switch between linear dynamics by use of the discrete controls. Note that this externally triggered switching is different from the class of discrete-time piecewise affine systems (Sontag, 1981), in which switching occurs autonomously and is bound to the fact that the continuous state enters a new (polytopic) state region.

If optimization-based computation of control strategies for SLS is considered, typically mixed-integer programming problems are encountered, which are known to be NP hard problems, see e.g. (Bussieck and Pruessner, 2003). Nevertheless, for the optimal open-loop control of discrete-time SLS with quadratic performance measure and without state and input constraints relatively efficient techniques have been proposed, see e.g. in (Görges et al., 2011). The complexity there is reduced via pruning of the search tree and accepting sub-optimal solutions. An on-line open-loop control approach for the case with state and input constraints is described in (Liu and Stursberg, 2018), where a trade-off between performance and applicability is obtained by tree search with cost bounds and search heuristics.

In contrast, the present paper aims at determining optimal *closed-loop* control laws to select the continuous and discrete inputs for any state of the SLS. In principle, this task can be solved by dynamic programming (Bellman, 2010), but the complexity prevents the use for most systems. The concept of approximate dynamic programming (ADP) (Bertsekas, 2019) is more promising in this respect, but has not yet been used for controller synthesis of SLS with consideration of constraints – this is the very objective of this paper. The approach is to learn the control law from a dataset, which may originate from off-line solution of mixed-integer programming problems for selected initial states, or from approximate dynamic programming over short horizons. The use of (deep) neural networks (NN) are proposed as parametric architectures for either approximating cost-to-go functions, or the continuous-discrete control laws. NN as

[a] https://orcid.org/0000-0003-4910-8218
[b] https://orcid.org/0000-0002-9600-457X

Markolf, L. and Stursberg, O.
Learning-based Optimal Control of Constrained Switched Linear Systems using Neural Networks.
DOI: 10.5220/0010581600900098
In *Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2021)*, pages 90-98
ISBN: 978-989-758-522-7

parametric approximators are appealing due to their property of universal approximation (Cybenko, 1989; Hornik et al., 1989) and the recent success of deep learning (Goodfellow et al., 2016). For the different task of model predictive control of purely continuous-valued systems, recent work has shown that the use of NN (determined off-line) can lead to solutions of problems in which full on-line computation takes prohibitively long, see e.g. (Chen et al., 2018; Hertneck et al., 2018; Karg and Lucia, 2020; Paulson and Mesbah, 2020; Markolf and Stursberg, 2021). This motivates for the present paper to train NN also for optimal control of SLS, despite the complexity arising from the mixed inputs. While the analysis of neural networks is known to be challenging due to their nonlinear and often large-scale structure, this paper makes use of the methods developed in (Chen et al., 2018) and (Markolf and Stursberg, 2021) to ensure the satisfaction of state and input constraints in addition.

The considered problem is stated in Sec. 2, while Sec. 3 shows how an ADP approach can be formulated for SLS in principle. The specific choice of NN as function approximators and solutions for considering the constraints, as well as the main algorithms are detailed in Sec. 4. A numerical example is provided in Sec. 5, before the paper is concluded in Sec. 6.

2 PROBLEM STATEMENT

This paper considers discrete-time and constrained switched systems of the form:

$$x_{k+1} = f_{v_k}(x_k, u_k), \quad k \in \{0,\dots,N-1\}, \tag{1}$$

where k is the time index, N a finite time horizon, $x_k \in \mathbb{R}^{n_x}$ the continuous state vector, $u_k \in \mathbb{R}^{n_u}$ the vector of continuous control inputs, and $v_k \in V = \{v_{[1]},\dots,v_{[n_v]}\}$ the discrete control input determining the subsystem $f_{v_k} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ selected at time k. The focus in this paper is on switched linear systems with matrices A_{v_k} and B_{v_k} of appropriate dimensions:

$$f_{v_k}(x_k, u_k) = A_{v_k} x_k + B_{v_k} u_k, \quad k \in \{0,\dots,N-1\}. \tag{2}$$

The states and inputs are constrained to polytopes X and U:

$$x_k \in X = \left\{ x \in \mathbb{R}^{n_x} \,|\, H^X x \le h^X \right\}, \tag{3}$$

$$u_k \in U = \left\{ u \in \mathbb{R}^{n_u} \,|\, H^U u \le h^U \right\}, \tag{4}$$

with matrices H^X, H^U, and vectors h^X, h^U. For a given state x_k, $k \in \{0,\dots,N-1\}$ and input sequences over a time span $\{k,\dots,N-1\}$:

$$\phi_k^u := \{u_k,\dots,u_{N-1}\}, \tag{5}$$

$$\phi_k^v := \{v_k,\dots,v_{N-1}\}, \tag{6}$$

the corresponding unique state sequence is denoted by:

$$\phi_k^x := \{x_k,\dots,x_N\} \tag{7}$$

and obtained from (1).

Let $X_N \subseteq X$ be a target set specified as polytope:

$$X_N = \left\{ x \in \mathbb{R}^{n_x} \,|\, H^{X_N} x \le h^{X_N} \right\} \subseteq X. \tag{8}$$

Furthermore, let a sequence of state sets $\{X_0,\dots,X_{N-1}\}$ be defined, satisfying:

$$X_k = \{x \in X \,|\, \text{for each } v \in V : \exists u \in U \text{ such that } f_v(x,u) \in X_{k+1}\}, \quad k \in \{0,\dots,N-1\}, \tag{9}$$

i.e., for any state $x_k \in X_k$, $k \in \{0,\dots,N-1\}$ and an arbitrary choice of the discrete input $v_k \in V$, at least one admissible continuous input $u_k \in U$ exists such that $f_{v_k}(x_k,u_k) \in X_{k+1}$. The actual computation of these sets will be addressed later in Sec. 4.

The following fact then obviously holds:

Proposition 1. *If X_0 is nonempty, then for each initialization $x_0 \in X_0$ and each discrete input sequence ϕ_0^v at least one admissible continuous input sequence ϕ_0^u exists that transfers x_0 into the target set X_N, while satisfying $x_i \in X_i$, $i \in \{0,\dots,N\}$ and $u_i \in U$, $i \in \{0,\dots,N-1\}$.*

For introducing costs, assume that ϕ_k^x has been determined for a given state x_k at time $k \in \{0,\dots,N-1\}$, and for given input sequences ϕ_k^u and ϕ_k^v. Then let $J_k(\phi_k^x, \phi_k^u, \phi_k^v)$ denote the total cost associated with this evolution:

$$J_k(\phi_k^x, \phi_k^u, \phi_k^v) = g_N(x_N) + \sum_{i=k}^{N-1} g_{i,v_i}(x_i, u_i), \tag{10}$$

where $g_N : X \to \mathbb{R}^{\ge 0}$ denotes a terminal cost, and $g_{k,v_k} : X \times U \to \mathbb{R}^{\ge 0}$, $k \in \{0,\dots,N-1\}$ a stage cost for $v_k \in V$ at stage k. Furthermore, let $J_k^*(x)$ be the optimal cost-to-go for steering a state x at time $k \in \{0,\dots,N-1\}$ into the target set X_N within $N-k$ steps, while satisfying $x_i \in X_i$, $i \in \{k,\dots,N\}$ and $u_i \in U$, $i \in \{k,\dots,N-1\}$, i.e.:

$$J_k^*(x) = \min_{\phi_k^u, \phi_k^v} \quad J_k(\phi_k^x, \phi_k^u, \phi_k^v) \tag{11a}$$

subject to:

$$x_k = x, \tag{11b}$$

$$x_{i+1} = A_{v_i} x_i + B_{v_i} u_i, \quad i \in \{k,\dots,N-1\}, \tag{11c}$$

$$u_i \in U, \quad i \in \{k,\dots,N-1\}, \tag{11d}$$

$$x_i \in X_i, \quad i \in \{k,\dots,N\}. \tag{11e}$$

By convention, J_k^* is set to infinity for the case that (11) is infeasible for a specific state $x \in \mathbb{R}^{n_x}$. Subsequently, the constraints (11d) and (11e) are replaced by $u_i \in U_i(x_i, v_i)$, $i \in \{k, \dots, N-1\}$. Here, $U_k(x_k, v_k)$, $k \in \{0, \dots, N-1\}$ denotes a set that depends on the state $x_k \in X$ and the discrete input $v_k \in V$, and contains all the continuous inputs $u_k \in U$ for which $f_{v_k}(x_k, u_k) \in X_{k+1}$:

$$U_k(x, v) = \{u \in U \mid f_v(x, u) \in X_{k+1}\}. \tag{12}$$

Again, the actual computation of these sets will be addressed later in Sec. 4

The following problem is considered in this work:

Problem 1 (Finite-Horizon Control Problem)**.** *Find an optimal control law which assigns to each state $x_k \in X_k$ at time instant $k \in \{0, \dots, N-1\}$ a pair of optimal admissible inputs $v_k^* \in V$ and $u_k^* \in U_k(x_k, v_k^*)$, such that for any $x_0 \in X_0$, the cost $J_0\left(\phi_0^{x^*}, \phi_0^{u^*}, \phi_0^{v^*}\right)$ obtained for the resulting sequences $\phi_0^{x^*}$, $\phi_0^{u^*}$, $\phi_0^{v^*}$ is optimal, i.e.: $J_0\left(\phi_0^{x^*}, \phi_0^{u^*}, \phi_0^{v^*}\right) = J_0^*(x_0)$.*

In order to allow for the use of gradient methods to tackle this problem, as proposed in (Markolf and Stursberg, 2021), the following assumption is made, which is not practically restrictive.

Assumption 1. *Suppose that the functions $g_{k,v}(x, u)$, $k \in \{0, \dots, N-1\}$, $v \in V$ in (10) are continuously differentiable with respect to $u \in U$.*

3 APPROXIMATE DYNAMIC PROGRAMMING FOR SLS

In theory, dynamic programming (DP) provides a scheme to solve the Problem 1: Starting from:

$$J_N^*(x_N) := g_N(x_N), \tag{13}$$

the DP algorithm proceeds backward in time from $N-1$ to 0 to compute the optimal cost-to-go functions:

$$J_k^*(x_k) = \min_{\substack{v_k \in V \\ u_k \in U_k(x_k, v_k)}} \left[g_{k,v_k}(x_k, u_k) + J_{k+1}^*(f_{v_k}(x_k, u_k)) \right]. \tag{14}$$

Such a version of the DP-algorithm is similar to the standard one without discrete inputs, as can be found e.g. in (Bertsekas, 2005). Provided that the optimal cost-to-go values J_k^* are known for all relevant x_k and k, the optimal discrete and continuous input sequences $\phi_0^{v^*}$ and $\phi_0^{u^*}$ for $x_0 \in X_0$ can be constructed in a forward manner by:

$$(v_k^*, u_k^*) \in \arg \min_{\substack{v_k \in V \\ u_k \in U_k(x_k^*, v_k)}} \Big[g_{k,v_k}(x_k^*, u_k) + J_{k+1}^*\left(f_{v_k}(x_k^*, u_k)\right) \Big], \tag{15}$$

with $x_0^* = x_0$ and $x_{k+1}^* = f_{v_k^*}(x_k^*, u_k^*)$.

For the general setup considered in this work, the DP algorithm does not lead to closed-form expressions for J_k^* and for the respective optimal policies denoted by:

$$\pi^{u^*} := \{\mu_0^{u^*}(\cdot), \dots, \mu_{N-1}^{u^*}(\cdot)\},\ \mu_k^{u^*} : X \to U, \tag{16}$$

$$\pi^{v^*} := \{\mu_0^{v^*}(\cdot), \dots, \mu_{N-1}^{v^*}(\cdot)\},\ \mu_k^{v^*} : X \to V. \tag{17}$$

Hence, numeric solution is necessary, but is known to suffer from the *curse of dimensions*, thus limiting practical applicability.

However, the optimal cost-to-go functions J_k^* can be approximated by parametric functions $\tilde{J}_k$ with real-valued parameter vectors r_k^J, constituting a so-called *approximation in value space* (Bertsekas, 2019). The prediction of the optimal cost-to-go J_k^* by $\tilde{J}_k$ given some state x_k is a typical regression task. Suppose for a moment that a parametric function $\tilde{J}_k$ and a data set consisting of state-cost pairs (x_k^s, J_k^s), $s \in \{1, \dots, q_k^J\}$ are available, where each J_k^s is a regression target providing the desired value for the corresponding example state x_k^s. On this basis, the parameter vector r_k^J of the parametric function $\tilde{J}_k$ can be adapted with the objective to improve the performance on the considered regression task by learning from the data set. The mean squared error is here (as usual) considered as performance measure. Such an adaption procedure for r_k^J, typically called *training*, is an example for supervised learning. A challenge hereby is to perform also well on previously not explored states x_k, which distinguishes the training procedure from pure optimization. For a more detailed treatment, see e.g. (Goodfellow et al., 2016).

The following Algorithm 1, which extends the sequential DP procedure from (Bertsekas, 2019) to SLS, provides an approach for training the parametric approximators $\tilde{J}_k$ in a recursive manner (similar to the typical DP procedure). Once the parametric approximators $\tilde{J}_k$ are trained, approximations of the optimal discrete and continuous input sequences, denoted as $\phi_0^{\tilde{v}}$ and $\phi_0^{\tilde{u}}$, can be constructed for $x_0 \in X_0$ in a forward manner, similar to (15):

$$(\tilde{v}_k, \tilde{u}_k) \in \arg \min_{\substack{v_k \in V \\ u_k \in U_k(\tilde{x}_k, v_k)}} \Big[g_{k,v_k}(\tilde{x}_k, u_k) + \tilde{J}_{k+1}\left(f_{v_k}(\tilde{x}_k, u_k), r_{k+1}^J\right) \Big], \tag{18}$$

with $\tilde{x}_0 = x_0$ and $\tilde{x}_{k+1} = f_{\tilde{v}_k}(\tilde{x}_k, \tilde{u}_k)$.

Another way to obtain approximations of the optimal discrete and continuous input sequences is the approximation of the optimal policies π^{v^*} and π^{u^*} by so-called *parametric policies*:

$$\pi^{\tilde{v}} = \{\mu_1^{\tilde{v}}(\cdot, r_1^v), \ldots, \mu_{N-1}^{\tilde{v}}(\cdot, r_{N-1}^v)\}, \quad \text{with } \mu_k^{\tilde{v}}(\cdot, r_k^v) : X \to V, \tag{19}$$

$$\pi^{\tilde{u}} = \{\mu_1^{\tilde{u}}(\cdot, r_1^u), \ldots, \mu_{N-1}^{\tilde{u}}(\cdot, r_{N-1}^u)\}, \quad \text{with } \mu_k^{\tilde{u}}(\cdot, r_k^u) : X \to U_k(\cdot, \mu_k^{\tilde{v}}(\cdot, r_k^v)). \tag{20}$$

This approach is an example of an *approximation in policy space* (Bertsekas, 2019). Again, the parameter vectors r_k^u and r_k^v can be adapted by standard supervised learning techniques on the basis of available data sets (x_k^s, u_k^s), $s \in \{1, \ldots, q_k^u\}$ and (x_k^s, v_k^s), $s \in \{1, \ldots, q_k^v\}$, respectively. The data may originate from solutions of (18), constituting an example of *approximation in policy space on top of approximation in value space*.

4 OPTIMAL CONTROL OF SLS WITH CONSTRAINTS USING NN

Based on the rather conceptual derivations in the preceding chapter, the specific approach for synthesizing optimal control laws based on NN for SLS with input and state constraints is now described. Two procedures of using (deep) neural networks as parametric architectures for the approximation of the optimal continuous and discrete input sequences are proposed, one by approximation in value space, and the other by approximation in policy space. In both cases,

Algorithm 1: Sequential Dynamic Programming.

1: $\tilde{J}_N(x_N, r_N^J) := g_N(x_N)$
2: **for** $k = N-1$ to 0 **do**
3: Generate a large number of states x_k^s, $s \in \{1, \ldots, q_k^J\}$ by sampling the state space X_k
4: **for** $s = 1$ to q_k^J **do**
5:
$$J_k^s = \min_{\substack{v \in V \\ u \in U_k(x_k^s, v)}} \Big[g_{k,v}(x_k^s, u) + \tilde{J}_{k+1}\left(f_v(x_k^s, u), r_{k+1}^J\right) \Big]$$
6: **end for**
7: Determine r_k^J by training with (x_k^s, J_k^s), $s \in \{1, \ldots, q_k^J\}$
8: **end for**

neural networks $\hat{v}_k$, $k \in \{0, \ldots, N-1\}$ with softmax output units and parameter vectors r_k^v are employed to obtain for each $x_k \in X_k$ an n_v-dimensional output vector with $\hat{v}_{k,i}(x_k, r_k^v) = P(v_k^* = v_{[i]} \,|\, x_k)$. Thus, the outputs of the neural network $\hat{v}_k$ determine for x_k a probability distribution over the discrete control inputs, where $\hat{v}_{k,i}(x_k, r_k^v)$ denotes the probability that the discrete control input $v_{[i]}$ is optimal for x_k at stage k.

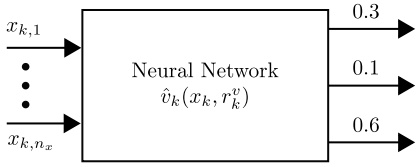
4.1 Approximation in Value Space

Neural networks with continuous and continuously differentiable activation functions are proposed as parametric approximators $\tilde{J}_k$ for the optimal cost-to-go functions J_k^*, $k \in \{0, \ldots, N-1\}$. Furthermore, a set $V_{\text{priority}}(\hat{v}_k, n_{\text{priority}}) \subseteq V$ is introduced, which is used in approximating the optimal discrete and optimal input sequences in a forward manner for given $x_0 \in X_0$ and $n_{\text{priority}} \in \{1, \ldots, n_v\}$:

$$(\tilde{v}_k, \tilde{u}_k) \in \arg \min_{\substack{v_k \in V_{\text{priority}}\left(\hat{v}_k\left(\tilde{x}_k, r_k^v\right), n_{\text{priority}}\right) \\ u_k \in U_k(\tilde{x}_k, v_k)}} \Big[g_{k,v_k}(\tilde{x}_k, u_k) + \tilde{J}_{k+1}\left(f_{v_k}(\tilde{x}_k, u_k), r_{k+1}^J\right) \Big], \tag{21}$$

with $\tilde{x}_0 = x_0$ and $\tilde{x}_{k+1} = f_{\tilde{v}_k}(\tilde{x}_k, \tilde{u}_k)$. Here, $n_{\text{priority}} \in \{1, \ldots, n_v\}$ is a user-defined number used to specify the number of elements in $V_{\text{priority}}(\hat{v}_k, n_{\text{priority}})$, where these elements are selected to be the discrete inputs with the highest probabilities according to $\hat{v}_k$. This allows to establish a trade-off in (21) between the consideration of only a single discrete input (with the highest probability to be the optimal one) or all discrete inputs contained in V. Fig. 1 provides an example of the concept.

It will be addressed in detail in Sec. 4.3 how to compute $U_k(x, v)$, $k \in \{0, \ldots, N-1\}$, and to show that



$$P\left(v_k^* = v_{[1]} \mid x_k\right) = \hat{v}_{k,1} = 0.3$$
$$P\left(v_k^* = v_{[2]} \mid x_k\right) = \hat{v}_{k,2} = 0.1$$
$$P\left(v_k^* = v_{[3]} \mid x_k\right) = \hat{v}_{k,3} = 0.6$$
$$V_{\text{priority}}\left(\hat{v}_k(x_k, r_k^v), 1\right) = \left\{v_{[3]}\right\}$$
$$V_{\text{priority}}\left(\hat{v}_k(x_k, r_k^v), 2\right) = \left\{v_{[1]}, v_{[3]}\right\}$$
$$V_{\text{priority}}\left(\hat{v}_k(x_k, r_k^v), 3\right) = \left\{v_{[1]}, v_{[2]}, v_{[3]}\right\}$$
$$\mu_k^{\tilde{v}}\left(x_k, r_k^v\right) = v_{[3]}$$

Figure 1: Example for illustrating the use of the neural network $\hat{v}_k$.

$U_k(x,v)$ in (12) is a polytope for all $x \in X_k$ and $v \in V$. The architecture of the neural networks and a closed-form expression for the partial derivative of $\tilde{J}_k(x, r_k^J)$ with respect to x will be described in Sec. 4.4. The property that $U_k(x,v)$, $k \in \{0,\dots,N-1\}$ is a polytope and the availability of closed-form expressions for $[\partial \tilde{J}_k/\partial x](x, r_k^J)$ open the door for addressing the minimization problems in (18), (21), and Algorithm 1 by applying well-established gradient methods for each considered discrete input $v \in V$. The satisfaction of the convex constraints $u \in U_k(x,v)$ in state-of-the-art methods of this type is not a problem. Hence, the satisfaction of the considered state and input constraints in (11) is guaranteed, even in case of imperfect approximations of the optimal cost-to-go functions, or if the iterative procedure of the gradient method is stopped before finding a local minimum. This approach can be based on existing work on gradient-methods for systems without switching (Markolf and Stursberg, 2021).

4.2 Approximation in Policy Space

Alternatively, NN can be used directly as parametric approximators $\mu_k^{\tilde{u}}$ of the optimal continuous policies $\mu_k^{u^*}$, $k \in \{0,\dots,N-1\}$. The optimal discrete input policies $\mu_k^{v^*}$ are approximated by:

$$\mu_k^{\tilde{v}}(x_k, r_k^v) \in \big\{ v_{[i]} \in V \,|\, \text{for all } j \in \{1,\dots,n_v\} : P\big(v_k^* = v_{[i]} \,|\, x_k\big) \geq P\big(v_k^* = v_{[j]} \,|\, x_k\big) \big\}. \tag{22}$$

For a given initial state $x_0 \in X_0$, the scheme is then to approximate the optimal continuous and discrete input sequences in forward manner by computing:

$$\tilde{v}_k = \mu_k^{\tilde{v}}(\tilde{x}_k, r_k^v) \in V, \tag{23}$$

$$\tilde{u}_k = \mu_k^{\tilde{u}}(\tilde{x}_k, \tilde{v}_k, r_k^u) \in U_k(\tilde{x}_k, \tilde{v}_k), \tag{24}$$

with $\tilde{x}_0 = x_0$ and $\tilde{x}_{k+1} = f_{\tilde{v}_k}(\tilde{x}_k, \tilde{u}_k)$.

Provided that $\mu_k^{\tilde{u}}(x_k, v_k, r_k^u) \in U_k(x_k, v_k)$ for each $x_k \in X_k$ and $v_k \in V$, the satisfaction of the state and input constraints considered in (11) is guaranteed. This can be achieved by projecting the output of the neural network onto the polytope $U_k(x_k, v_k)$. An approach to projecting the output of a neural network onto a polytope can be found in (Chen et al., 2018).

4.3 Controllable Sets

For a given $v \in V$, let $\mathrm{Pre}_v(\mathcal{X})$ be the set of state predecessors to $\mathcal{X}$, i.e. containing all the states $x \in \mathbb{R}^{n_x}$ for which at least one input $u \in U$ exists such that $f_v(x,u) \in \mathcal{X}$:

$$\mathrm{Pre}_v(\mathcal{X}) = \{ x \in \mathbb{R}^{n_x} \,|\, \exists u \in U \text{ such that } f_v(x,u) \in \mathcal{X} \}. \tag{25}$$

If $\mathcal{X}$ is a polytope, then $\mathrm{Pre}_v(\mathcal{X})$ results from a linear transformation of $\mathcal{X}$, and is thus also a polytope. Details about the computation of $\mathrm{Pre}_v(\mathcal{X})$ for a polytope $\mathcal{X}$ can be found e.g. in (Borrelli et al., 2017).

Starting from a target set $X_N \subseteq X$, the sequence of state sets $\{X_0,\dots,X_{N-1}\}$ can be computed recursively as shown in Algorithm 2. Since X_N is specified in (8) as polytope, each X_k, $k \in \{0,\dots,N-1\}$ defined in (9) is again a polytope:

$$X_k = \big\{ x \in \mathbb{R}^{n_x} \,|\, H^{X_k} x \leq h^{X_k} \big\}. \tag{26}$$

For polytopic sets X_k, $k \in \{0,\dots,N-1\}$, also the sets $U_k(x,v)$ defined by (12) are polytopes for all $x \in X_k$ and $v \in V$, and given by:

$$U_k(x,v) = \{ u \in \mathbb{R}^{n_u} \,|\, H^{U_k}(v) u \leq h^{U_k}(x,v) \}, \tag{27}$$

with:

$$H^{U_k}(v) = \begin{bmatrix} H^{X_{k+1}} B_v \\ H^U \end{bmatrix}, \tag{28}$$

$$h^{U_k}(x,v) = \begin{bmatrix} h^{X_{k+1}} - H^{X_{k+1}} A_v x \\ h^U \end{bmatrix}. \tag{29}$$

4.4 Neural Networks

Feed-forward NN characterized by a chain structure:

$$h(x) = (h^{(L)} \circ \cdots \circ h^{(2)} \circ h^{(1)})(x) \tag{30}$$

are considered, with final layer $h^{(L)}$ and hidden layers $h^{(\ell)}$, $\ell \in \{1,\dots,L-1\}$. Such structures are commonly used and detailed information can be found in several textbooks, see e.g. (Goodfellow et al., 2016). The output of layer ℓ is denoted as $\eta^{(\ell)}$ in the following, while $\eta^{(0)}$ is defined to be the input of the overall network:

$$\eta^{(0)}(x) = x, \tag{31}$$

$$\eta^{(\ell)}(x) = (h^{(\ell)} \circ \cdots \circ h^{(1)})(x). \tag{32}$$

Here, the hidden layers are (as usual) considered to be vector-to-vector functions of the form:

$$h^{(\ell)}(\eta^{(\ell-1)}) = (\phi^{(\ell)} \circ \psi^{(\ell)})(\eta^{(\ell-1)}), \tag{33}$$

with affine and nonlinear transformations $\psi^{(\ell)}$ and $\phi^{(\ell)}$, respectively. The affine transformation can be

Algorithm 2: Controllable Set Computation.

Input: n_v, N, X, X_N
Output: $X_1,\dots,X_{N-1}$

```
for k = N − 1 to 0 do
    X_k = X
    for i = 1 to n_v do
        X_k ← Pre_{v[i]}(X_{k+1}) ∩ X_k
    end for
end for
```

affected by the choice of the weight matrix $W^{(\ell)}$ and the bias vector $b^{(\ell)}$:

$$\psi^{(\ell)}(\eta^{(\ell-1)}) = W^{(\ell)}\eta^{(\ell-1)} + b^{(\ell)}. \tag{34}$$

Each layer can be interpreted to consist of parallel acting units, where a positive integer $S^{(\ell)}$ is used here to describe the number of units in layer ℓ. Each unit i in layer ℓ defines a vector-to-scalar function, which is the i-th component of $h^{(\ell)}$. In the case of hidden layers, $h_i^{(\ell)}(\eta^{(\ell-1)}) = \phi_i^{(\ell)}(W^{(\ell)}\eta^{(\ell-1)} + b^{(\ell)})$, where $\phi_i^{(\ell)}$ is known as activation function and often chosen as a rectified linear unit or a sigmoid function. For the purposes of this work, linear and softmax output units are considered. For a neural network with linear output units, the function $h^{(L)}$ is an affine transformation:

$$\psi^{(L)}(\eta^{(L-1)}) = W^{(L)}\eta^{(L-1)} + b^{(L)}. \tag{35}$$

Such an affine transformation arises also in softmax output units, in which $h_i^{(L)}$ is set to:

$$\text{softmax}_i\left(\psi^{(L)}\left(\eta^{(L-1)}\right)\right) = \frac{\exp\left(\psi_i^{(L)}\left(\eta^{(L-1)}\right)\right)}{\sum_{j=1}^{S^{(L)}}\exp\left(\psi_j^{(\ell)}\left(\eta^{(L-1)}\right)\right)}. \tag{36}$$

The neural network (30) belongs to the family of parametric functions, whose shape is formed by the parameter vector consisting of the weights and biases:

$$r = \begin{bmatrix} W_{1,1}^{(1)} & \dots & W_{S^{(L)},S^{(L-1)}}^{(L)} & b_1^{(1)} & \dots & b_{S^{(L)}}^{(L)} \end{bmatrix}^T. \tag{37}$$

4.4.1 Approximating the Optimal Cost-to-Go Functions

For approximating the optimal cost-to-go functions J_k^* by $\tilde{J}_k$, $k \in \{0,\dots,N-1\}$, the NN structure (30) is used with continuous and continuously differentiable activation functions (such as sigmoid functions) and linear output units. This allows for deriving closed-form expressions (Markolf and Stursberg, 2021) for the partial derivatives of h with respect to its arguments:

$$\frac{\partial h(x)}{\partial x} = \prod_{i=0}^{L-1}\frac{\partial h^{(L-i)}(\eta^{(L-(i+1))}(x))}{\partial \eta^{(L-(i+1))}}, \tag{38}$$

with:

$$\frac{\partial h^{(\ell)}(\eta^{(\ell-1)}(x))}{\partial \eta^{(\ell-1)}} = \frac{\partial \phi^{(\ell)}(\psi^{(\ell)}(\eta^{(\ell-1)}(x)))}{\partial \psi^{(\ell)}}\cdot W^{(\ell)} \tag{39}$$

for $\ell \in \{1,\dots,L-1\}$, and:

$$\frac{\partial h^{(L)}(\eta^{(L-1)}(x))}{\partial \eta^{(L-1)}} = W^{(L)}. \tag{40}$$

4.4.2 Approximating the Optimal Discrete Input Policies

As described above, the optimal discrete policies $\mu_k^{v^*}$ can be approximated by parametric policies $\mu_k^{\tilde{v}}$ based on the probability distributions defined by the neural networks $\hat{v}_k$, $k \in \{0,\dots,N-1\}$. For this, the NN structure (30) with softmax output units (36) is used as architecture for $\hat{v}_k$. Softmax units as output units are common, e.g. in classification tasks (Goodfellow et al., 2016), to represent probability distributions over different classes. According to (36), each output of the NN with softmax output units is in between 0 and 1, and all outputs sum up to 1, leading to a valid probability distribution.

4.4.3 Approximating the Optimal Continuous Input Policies

For the approximation of the optimal continuous input policies $\mu_k^{u^*}$ by $\mu_k^{\tilde{u}}$, $k \in \{0,\dots,N-1\}$, the use of the NN structure (30) with common activation functions and linear output units is proposed, following (Chen et al., 2018), where Dykstra's projection algorithm is used to project a potentially infeasible output onto the admissible polytope. This is exploited here to ensure that each $\tilde{u}_k$, as computed for $x_k \in X_k$ and $v_k \in V$ by (24), is an element of the polytope (27).

4.5 Main Algorithms

In order to summarize and combine the concepts introduced above, this subsection contains the overall algorithms to compute approximations of the optimal discrete and continuous input sequences as solution to Problem 1. While Algorithm 3 contains the procedure for approximation in value space, Algorithm 4 establishes the solution by approximation in policy space.

Algorithm 3: Finite-Horizon Control by Approximation in Value Space.

Input: $\tilde{x}_0 \in X_0$, $n_{\text{priority}} \in \{1,\dots,n_v\}$
Output: $\phi_0^{\tilde{x}} = \{\tilde{x}_0,\dots,\tilde{x}_N\}$, $\phi_0^{\tilde{u}} = \{\tilde{u}_0,\dots,\tilde{u}_{N-1}\}$, $\phi_0^{\tilde{v}} = \{\tilde{v}_0,\dots,\tilde{v}_{N-1}\}$

1: **for** $k = 0$ to $N-1$ **do**
2: determine $V_{\text{priority}}(\hat{v}_k(\tilde{x}_k, r_k^v), n_{\text{priority}})$
3: obtain $(\tilde{v}_k, \tilde{u}_k)$ from (21) for $\tilde{x}_k$
4: compute $\tilde{x}_{k+1} = f_{\tilde{v}_k}(\tilde{x}_k, \tilde{u}_k)$
5: **end for**

Algorithm 4: Finite-Horizon Control by Approximation in Policy Space.

Input: $\tilde{x}_0 \in X_0$
Output: $\phi_0^{\tilde{x}} = \{\tilde{x}_0, \ldots, \tilde{x}_N\}$, $\phi_0^{\tilde{u}} = \{\tilde{u}_0, \ldots, \tilde{u}_{N-1}\}$, $\phi_0^{\tilde{v}} = \{\tilde{v}_0, \ldots, \tilde{v}_{N-1}\}$
1: **for** $k = 0$ to $N-1$ **do**
2: determine $\tilde{v}_k = \mu_k^{\tilde{v}}(\tilde{x}_k, r_k^v)$
3: compute $\tilde{u}_k = \mu_k^{\tilde{u}}(\tilde{x}_k, \tilde{v}_k, r_k^u)$
4: evaluate $\tilde{x}_{k+1} = f_{\tilde{v}_k}(\tilde{x}_k, \tilde{u}_k)$
5: **end for**

5 NUMERICAL EXAMPLE

This section provides a numerical example for the illustration and evaluation of the proposed approaches. Hereto, a switched system (2) with matrices:

$$A_1 = \begin{bmatrix} 0 & 1 \\ -0.8 & 2.4 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 1 \\ -1.8 & 3.6 \end{bmatrix},$$
$$A_3 = \begin{bmatrix} 0 & 1 \\ -0.56 & 1.8 \end{bmatrix}, \quad A_4 = \begin{bmatrix} 0 & 1 \\ -8 & 6 \end{bmatrix}, \tag{41}$$
$$B_1 = B_2 = B_3 = B_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

is considered. This simple example, which is taken from (Görges, 2012), is chosen with the intention to ease the illustration of the procedures (not to demonstrate computational efficiency). The polytopes $X = \{x \in \mathbb{R}^2 \,|\, |x_i| \leq 1\}$ and $U = \{u \in \mathbb{R} \,|\, |u| \leq 4\}$ are specified as state and input constraints, and a quadratic cost function (10) is chosen:

$$g_N(x) = x^T x, \tag{42}$$
$$g_{k,v}(x,u) = x^T x + u^2 \text{ for all } k \in \{0, \ldots, N-1\}, v \in V. \tag{43}$$

The target set is specified to be identical to the origin of the state space $X_N = \{0\}$. If for this simple system, a low number $N = 6$ is chosen, the optimal solution of the corresponding instance of Problem 1 can be computed by enumerating over the 4^N possible discrete input sequences and solving one quadratic program (QP) each – this optimal solution serves to compare it with the approximating solutions obtained from the two proposed approaches based on approximation in value space, or in policy space respectively.

For all NN required in the proposed approaches, structures with one hidden layer and 50 units have been chosen. In each hidden unit, the hyperbolic tangent has been selected as activation function. The neural networks $\tilde{J}_k$ used for approximating the optimal cost-to-go values have been trained with state-cost pairs (x_k^s, J_k^s), $s \in \{1, \ldots, q_k^J\}$ generated on the basis of the sequential dynamic programming procedure described in Algorithm 1, where $q_k^J = 1000$ states x_k^s have been obtained for each $k \in \{0, \ldots, N-1\}$ by gridding the state space X_k obtained from Algorithm 2.

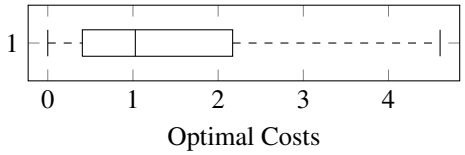


Figure 2: Box plot diagram with showing the distribution of the optimal costs $J_0^*(x_0^p)$ for 1000 initial states x_0^p obtained by gridding X_0.

For the same states x_k^s, the NN for approximating the optimal discrete and continuous input policies have been trained with state-input pairs (x_k^s, u_k^s) and (x_k^s, v_k^s), respectively, generated by addressing a minimization problem of type (18) with the previously determined $\tilde{J}_{k+1}$.

Let $\phi_0^{\tilde{u}}$ and $\phi_0^{\tilde{v}}$ denote approximated input sequences obtained for a specific initial state $\tilde{x}_0 \in X_0$ by either applying Algorithm 3 or Algorithm 4. Moreover, let $\phi_0^{\tilde{x}}$ be the resulting state sequence, and $\hat{J}_k(\tilde{x}_0)$ the cost obtained for $\phi_0^{\tilde{x}}$, $\phi_0^{\tilde{u}}$, and $\phi_0^{\tilde{v}}$ according to (10). For the evaluation of the approximation quality, 1000 initial states x_0^p, $p \in \{1, \ldots, n_p = 1000\}$ have been determined by gridding the set X_0, which is the backward reachable set from X_N for the selected N. The distribution of the optimal costs $J_0^*(x_0^p)$ for the initial states x_0^p, $p \in \{1, \ldots, n_p\}$ is illustrated in the box plot shown in Fig. 2. The average computation time for the determination of the optimal costs was 19.8 s on a common notebook (Intel® Core™ i5 – 7200U Processor), where the CPLEXQP solver from the IBM® ILOG® CPLEX® Optimization Studio has been used for the solution of the quadratic programs. On the other hand, the costs $\hat{J}_0(x_0^p)$ for the initial states x_0^p, $p \in \{1, \ldots, n_p\}$ were determined for the approximated solutions obtained from the approaches for approximation in value space, or approximation in policy space, where for the former all possible values for $n_{\text{Priority}} \in \{1, \ldots, 4\}$ were considered. The corresponding mean-squared errors can be found in the third column of Table 1, using:

$$\text{MSE} = \frac{1}{n_p} \sum_{p=1}^{n_p} \left(J_0^*(x_0^p) - \hat{J}_0(x_0^p) \right)^2. \tag{44}$$

The average computation times are listed in the fourth column of the same table.

As documented in Table 1, the average computation time for the optimal results is significantly higher than those for the approximated results. Moreover, the average computation time for the approach based

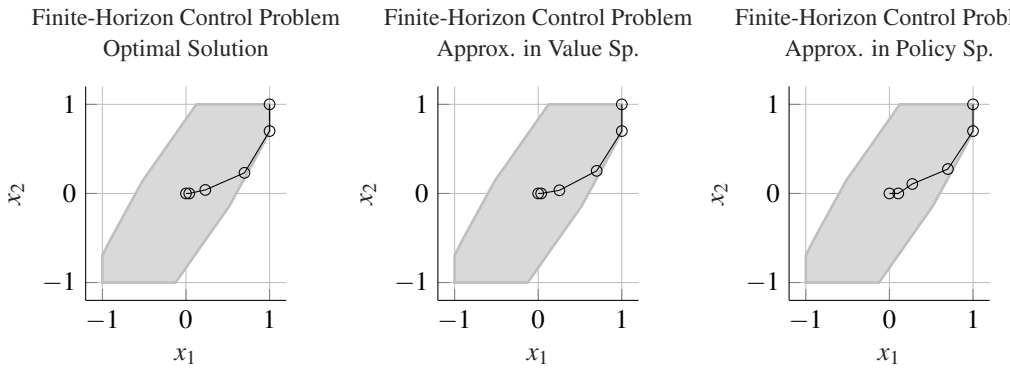


Figure 3: Solutions for the initial state $x_0 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$: optimal one, and for the two proposed techniques. The shaded polytope marks X_0.

Table 1: Mean squared errors according to (44) and average computation times to determine $\hat{J}_0(x_0^p)$ for 1000 initial states x_0^p obtained by gridding X_0.

Type of Approx.	n_{Priority}	MSE	Average Comp. Time
Value Space	4	3.51×10^{-4}	1.73 s
	3	3.51×10^{-4}	1.38 s
	2	3.50×10^{-4}	1.03 s
	1	6.56×10^{-4}	0.61 s
Policy Space	–	5.27×10^{-2}	0.27 s

on approximation in policy space was smaller than those for the approximating in value space. For the latter, the average computation times obviously grow with increasing n_{Priority}. Not surprisingly, the relatively high computation time for the optimal solutions are due to the large number of possible discrete input sequences. The use of an NN, as required for approximation in policy space, is in general faster than applying the gradient method n_{Priority}-times in the approach based on approximation in value space. Interestingly, the MSE for $n_{\text{Priority}} = 2$ to $n_{\text{Priority}} = 4$ are almost the same and very small. The observation that the MSE for the approximation in policy space is the largest depends (among other factors) on the fact that the training data for the NN $\tilde{\mu}_k$ has been generated on top of approximation in value space.

The state trajectories obtained from the optimal and approximated solutions of Problem 1 for the initial state $x_0 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$, as well as the polytope X_0 are illustrated in Fig. 3. The optimal state trajectory and the one approximated in value space are almost identical. For the state trajectory obtained from approximation in policy space, a slight difference is visible.

It is worth to stress that also for the approximated solutions, the sets defined in (27) ensure the satisfaction of the state and input constraints in (11).

6 CONCLUSION

This paper has proposed two solution techniques to synthesize optimal closed-loop controllers in form of NN for finite-horizon optimal control problems for discrete-time and constrained switched linear systems. Two general types of ADP, namely approximation in value space and approximation in policy space, were considered for fast approximation of the optimal solutions. For both ADP types, (deep) neural networks were chosen as parametric approximators. Established methods for projection or for constraint handling in nonlinear programming have been exploited to ensure the satisfaction of polytopic state and input constraints.

Properties of the optimal cost-to-go functions and optimal policies for the considered problem class have not been investigated in this work. Gaining a deeper insight by future work may help to specify the architectures of the neural networks. An approach to ensure the satisfaction of polytopic (continuous) input constraints by a policy based on neural networks without a subsequent projection has been recently proposed in (Markolf and Stursberg, 2021). A point of work future is to investigate if also that approach can be extended to guarantee the satisfaction of the considered state constraints.

REFERENCES

Bellman, R. (2010). *Dynamic Programming*. Princeton University Press.

Bertsekas, D. P. (2005). *Dynamic programming and optimal control*. Athena Scientific.

Bertsekas, D. P. (2019). *Reinforcement learning and optimal control*. Athena Scientific.

Borrelli, F., Bemporad, A., and Morari, M. (2017). *Predictive control for linear and hybrid systems*. Cambridge University Press.

Branicky, M. S., Borkar, V. S., and Mitter, S. K. (1998). A unified framework for hybrid control: model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45.

Bussieck, M. and Pruessner, A. (2003). Mixed-integer nonlinear programming. *SIAG/OPT Newsletter: Views & News*, 14(1):19–22.

Chen, S., Saulnier, K., Atanasov, N., Lee, D. D., Kumar, V., Pappas, G. J., and Morari, M. (2018). Approximating explicit model predictive control using constrained neural networks. In *Proc. American Control Conference*, pages 1520–1527.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Görges, D. (2012). *Optimal control of switched systems: With application to networked embedded control systems*. Logos-Verlag.

Görges, D., Izak, M., and Liu, S. (2011). Optimal control and scheduling of switched systems. *IEEE Transactions on Automatic Control*, 56(1):135–140.

Hertneck, M., Kohler, J., Trimpe, S., and Allgower, F. (2018). Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, 2(3):543–548.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Karg, B. and Lucia, S. (2020). Efficient representation and approximation of model predictive control laws via deep learning. *IEEE transactions on cybernetics*, 50(9):3866–3878.

Liu, Z. and Stursberg, O. (2018). Optimizing online control of constrained systems with switched dynamics. In *Proc. European Control Conference*, pages 788–794.

Markolf, L. and Stursberg, O. (2021). Polytopic input constraints in learning-based optimal control using neural networks. *arXiv e-print:2105.03376*.

Paulson, J. A. and Mesbah, A. (2020). Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction. *IEEE Control Systems Letters*, 4(3):719–724.

Sontag, E. (1981). Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2):346–358.