Abstract—Optimal switching between different topologies in step-down DC-DC voltage converters, with non-ideal inductors and capacitors, is investigated in this paper. Challenges including constraint on the inductor current and voltage leakages across the capacitor (due to switching) are incorporated. The objective is generating the desired voltage with low ripples and high robustness towards line and load disturbances. A previously developed tool which is based on approximate dynamic programming is adapted for this application. The scheme leads to tuning a parametric function approximator to provide optimal switching in a feedback form. No fixed cycle time is assumed, as the cycle time and the duty ratio will be adjusted on the fly in an optimal fashion. The controller demonstrates good capabilities in controlling the system even under parameter uncertainties. Finally, some modifications on the scheme are conducted to handle optimal switching problems with state jumps at the switching times.

Index Terms—Step-down dc-dc voltage converter, approximate dynamic programming, optimal switching with state jump, neural networks.

I. INTRODUCTION

The idea in the widely used DC-DC buck converters is switching between different topologies involving an inductor-capacitor filter such that a desired voltage is generated at the output. The problem, however, is finding the right switching time, such that the output has a desired steady state behavior (e.g., in terms of low harmonic distortions) as well as a suitable dynamic performance (e.g., low sensitivity towards disturbances) [1]–[3].

To have a short literature survey on this rich field, one may consider state space averaging scheme [4], linear quadratic regulator (LQR) control [5], [6], fuzzy logic control [7], [8], adaptive control [9], nonlinear control schemes [10], including sliding mode control [11], [12], and optimization based schemes, including model predictive control (MPC) [3], [13]–[16], and dynamic programming [17]–[19], used for control of voltage converters. The common practice for control of the converters is Pulse Width Modulation (PWM), where a fixed cycle time is selected and the duty ratio (the ratio of the cycle that a switch is closed, hence, a specific topology is selected) is adjusted, [3]–[7], [10], [13]. Another practice, however, is finding switching times without a fixed cycle, [14], [17]–[20].

The challenges in control of power converters are numerous, including desired low sensitivity towards variations in the source voltage (line disturbances) and in the load current (load disturbances), existing parameter uncertainties in modeling the system, enforcing constraints on the inductor current, and incorporation of voltage leaks across the capacitors after each switching. While each of the available method in the literature tackles some of these challenges, this study is aimed at developing a scheme which has the capability of addressing all of these concerns.

The idea is utilizing approximate dynamic programming (ADP), sometimes referred to by reinforcement learning (RL) or neuro-dynamic programming (NDP), for solving this problem. Interested readers are referred to the chapters in Ref. [21] for reviews of the ADP/RL field from perspectives of different authors of the chapters. ADP/RL has shown to have many advantages over other competitors in different applications, including power systems, both in voltage converters [17]–[19] and other problems of the discipline [22]–[24].

This study is mainly focused on investigation of an interesting “application” of a general purpose “tool” developed previously by the author for switching problems, namely, [25], [26]. The application is the voltage converter problem presented in [27] and [20]. Using the tool, the dynamic and steady state performance of the switching controller is shown to be promising in the voltage converter under line disturbances and parameter uncertainties. However, in investigating the sensitivity to the load current, the controller is observed to lead to an undesired steady state error. An idea is proposed for measuring and feeding the load current into the controller, to find the optimal switching schedule as a function of the current, i.e., having both the load current and the states as the feedback signals for the controller. After demonstrating the superior performance of this controller in handling the disturbance, another challenge is investigated, namely, voltage leakage across the capacitor after each switching. In other words, the problem is changed to a switching problem with state jump at the switches. Incorporating the state jump is shown to be a non-trivial task as it leads to fundamental changes in the controller. As a theoretical contribution of this study, a new learning scheme is proposed to accommodate the state jumps and the results are applied to the problem with desired outcomes. Finally, comparing this work with [25] and [26], the main difference is this study focuses on application while the cited two references are theoretical work. Also, the two papers do not address switching problems with state jump, which is addressed in here.

The closest studies to this work are [17] and [18], where relaxed dynamic programming was used for finding the approximate optimal switching schedule. The differences of this work compared with them are 1) providing detailed analyses of capabilities of the ADP-based controller in facing different existing challenges in voltage converters, 2) developing an idea for reducing the sensitivity of the result to load disturbances, and 3) incorporating voltage leakages through developing a
novel controller, i.e., a controller for optimal switching with state jumps.

The rest of this study is organized as follows. The switching problem of a power converter is briefly discussed in Section II. Afterwards, the theory of the method selected for finding the controller for the switching problem is detailed in III, followed by its implementation and numerical analyses in Section IV. The issue of sensitivity to load disturbances is addressed in Section V. The new scheme for incorporating switching problems with state jumps is theoretically developed and numerically analyzed in Section VI. Comparison of the results with two other methods is presented in Section VII. Finally, some concluding remarks are given in Section VIII.

II. DC-DC CONVERTER

Consider the DC-DC buck converter investigated in [27] and [20] shown in Fig. 1. The switch on the left of the inductor is subject to be shifted between the zero and one positions, each of which leads to a different topology for the circuit. The inductor (with the inductance of \(L\)) is assumed to be non-ideal, hence, has the resistance of \(r_i\), and the capacitor (with the capacitance of \(C\)) also has the internal resistance of \(r_c\), the current source \(i_o\) represents the load on the converter, and finally, the voltage input/source is given by voltage source \(V_s\). Denoting the voltage across the capacitor with \(V_c\) and the current through the inductor with \(i_l\), the state vector is selected as \(x := [V_c, i_l]^T\), motivated by [27] and [20].

The dynamics of the system can be modeled by the two modes based on the two values of the switching input \(v \in \{0, 1\}\) corresponding to the respective positions of the switch.

\[
\dot{x} = \frac{d}{dt} x = \begin{bmatrix} V_c \\ i_l \end{bmatrix} = \begin{bmatrix} C^{-1} \left( i_l - i_o \right) \\ L^{-1} \left( -V_c - (r_c + r_i) i_l + r_c i_o + V_s v \right) \end{bmatrix},
\]

Note the exogenous signals \(i_o\) and \(V_s\) present in the dynamics. Freezing the signals at constant values (i.e., no load and line disturbances) leads to a switching system with two autonomous subsystems.

The approach for controller design in this work calls for a discrete-time dynamics. Hence, the abovementioned system may be discretized using a small enough sampling time denoted with \(\Delta t\), such that the resulting discrete-time system approximates the continuous-time dynamics accurately. Denoting the state vector at discrete time \(k\) with \(x_k\) and using forward Euler integration for discretization, the system reads

\[
x_{k+1} = x_k + \Delta t \dot{x}_k,
\]

where the time-derivative of the state, i.e., \(\dot{x}_k\), is given by (1), based on the position of the switch.

The objective is manipulating the switch such that the voltage across the capacitor tracks some constant reference voltage \(V_{ref}\), where \(V_{ref} < V_s\), i.e., a step-down in the voltage is resulted. Finding an optimal switching schedule is sought in this study, hence, a performance measure or cost function is required to be selected. Motivated by [20] the following cost function is selected

\[
J = \sum_{k=0}^{\infty} \gamma^k (V_{ck} - V_{ref})^2,
\]

where \(V_{ck}\) denotes the voltage of the capacitor at time \(k\) and the discount factor, denoted with \(\gamma \in (0, 1]\) is incorporated to avoid an unbounded cost. Minimizing (3) while enforcing the desired behavior of tracking \(V_{ref}\), penalizes the voltage ripples, which is also desired.

The next section briefly reviews the approach selected for solving the switching problem of minimizing (3) subject to the dynamics given by (2) and (1).

III. OPTIMAL SWITCHING USING ADP

A. Problem formulation

A discrete-time switching system with \(M\) autonomous subsystems may be presented by

\[
x_{k+1} = f_v(x_k), k \in \mathbb{Z}_+, v \in \mathcal{V},
\]

where the dynamics of the modes are given by \(f_v : \mathbb{R}^n \to \mathbb{R}^n, \forall v \in \mathcal{V} := \{1, 2, ..., M\}\) and the dimension of the state vector \(x_k\) is given by positive integer \(n\). The discrete time index is denoted by subscript \(k\) in \(x_k\) and subscript \(v\) in \(f_v(.)\) corresponds to the respective mode/subsystem. Finally, the set of non-negative integers is denoted with \(\mathbb{Z}_+\). A switching schedule identifies \(v_k, \forall k \in \mathbb{Z}_+, \) where the active mode at time \(k\) is denoted with \(v_k\). The optimal solution is the switching schedule that minimizes the discounted cost function

\[
J = \sum_{k=0}^{\infty} \gamma^k Q(x_k, v_k),
\]

where function \(Q : \mathbb{R}^n \times \mathcal{V} \to \mathbb{R}_+, \text{ continuous with respect to } x, \text{ penalizes the state error (the difference between the actual value and the desired constant value for the state vector) during the horizon. Function } Q(., .) \text{ may generally be dependent on the active mode, for the purpose of having the flexibility of assigning different penalties to different modes. In (3), however, } Q(x, v) \text{ is defined as } (x(1) - V_{ref})^2 \text{ for a constant } V_{ref}, \text{ where } x(1) \text{ denotes the first state element, which is } V_c. \text{ Finally, set } \mathbb{R}_+ \text{ denotes the non-negative real numbers and } \gamma \in (0, 1]\text{ denotes the discount factor.}

B. Proposed solution

If the optimal value function, sometimes called optimal cost-to-go, is known, the optimal solution to the switching problem follows. The optimal value function is a function of state that gives the total cost of evaluating the cost function along the state trajectory propagated from the given current state using the optimal switching schedule. Denoting the value function with \(V^* : \mathbb{R}^n \to \mathbb{R}_+, \) one has

\[
V^*(x_0) = \sum_{k=0}^{\infty} \gamma^k Q(x_k^*, v^*(x_k^*)).
\]
where \( v^* : \mathbb{R}^n \rightarrow \mathcal{V} \) denotes the optimal feedback switching schedule (the \textit{optimal switching policy}). Moreover, the state trajectory resulting from the optimal decisions is denoted with \( x^*_k, \forall k = 0, 1, 2, \ldots \) and \( x^*_0 := x_0 \). Denoting \( v^*(x^*_k) \) with \( v^*_k \), Eq. (6) leads to
\[
V^*(x_k) = Q(x_k, v^*_k) + \gamma V^*(f_o(x_k)), \forall k \in \mathbb{Z}_+.
\] (7)

By the Bellman principle of optimality, [28], one has
\[
V^*(x) = \min_{v \in \mathcal{V}} \left( Q(x, v) + \gamma V^*(f_o(x)) \right), \forall x.
\] (8)

and the optimal mode, in a feedback form, is given by
\[
v^*(x) = \arg \min_{v \in \mathcal{V}} \left( Q(x, v) + \gamma V^*(f_o(x)) \right), \forall x.
\] (9)

The problem, hence, is finding the optimal value function, given by Eq. (8). An idea for solving this equation is using value iteration (VI), [25], where starting from an initial guess on \( V^0(x) \), one iterates through
\[
V^{j+1}(x) = \min_{v \in \mathcal{V}} \left( Q(x, v) + \gamma V^j(f_o(x)) \right), \forall x \in \mathcal{X},
\] (10)

until it converges, where the iteration index is given by \( j \). This is done through selecting a function approximator, e.g., neural networks, for approximating the value function within a compact set, denoted by \( \mathcal{X} \), which represents the domain of interest. It should be noted that the solution will be valid only if the entire state trajectory remains within \( \mathcal{X} \), hence, \( \mathcal{X} \) needs to be selected carefully and large enough. The implementation of the VI is as simple as selecting multiple random sample states \( x \in \mathcal{X} \) and evaluating the right hand side of (10) at each sample. These values can be used as \textit{targets} for learning the left hand side, for example using least squares. Given \( V^j(.) \), the iteration leads to \( V^{j+1}(.) \) and this successive approximation continues until the parameters of the function approximator converge. The following theorem provides sufficient conditions for this convergence.

Before that, it is worth mentioning that exact reconstruction of the right hand side of Eq. (10) with the left hand side, that is, exact ‘approximation’ is generally not possible. Hence, \textit{approximation errors} will be introduced into the problem.

Interested readers are referred to [29]–[31] for investigation of effects of these errors on the quality of the result.

\textbf{Theorem 1.} If there exists at least one switching policy using which cost function (5) remains bounded for all \( x_0 \in \mathcal{X} \) and if the initial guess \( V^0(.) \) in the value iteration given by (10) is selected as \( V^0(x) = 0, \forall x \in \mathcal{X} \), then, the iterations converge to the optimal value function given by Eq. (8).

\textbf{Proof:} A convergence proof was presented in [25] for ‘undiscounted’ cost functions. The idea was establishing an analogy between the \textit{iterations} of the VI and the \textit{horizon length} of a finite-horizon optimal control problem with a \textit{fixed final time}. That idea can be simply extended to the case of having discounted cost functions subject to this study. Let the cost function with the finite horizon of \( N \) time steps be given by
\[
J^N = \sum_{k=0}^{N-1} \gamma^k Q(x_k, v_k).
\] (11)

The finite-horizon problem is defined as minimizing \( J^N \) subject to the dynamics given by (4). Once the final time is fixed, the value function and the control policy become time-dependent [28], [32], i.e., they may be denoted with \( V^*(\cdot, \cdot) \) and \( v^*(\cdot, \cdot) \) respectively, where the second argument is the number of remaining time steps, or \textit{time-to-go}. Let the optimal finite-horizon value function given state \( x_0 \) and time-to-go \( \tau \in \mathbb{M} := \{0, 1, \ldots, N\} \) be denoted by \( V^*: \mathbb{R}^n \times \mathbb{M} \rightarrow \mathcal{R}_+ \), where
\[
V^*(x_0, \tau) = \sum_{k=0}^{\tau-1} \gamma^k Q(x^*_k, v^*(x^*_k, \tau - k)),
\] (12)

where \( x^*_k := f \left( x^*_{k-1}, v^*(x^*_{k-1}, \tau - (k - 1)) \right), \forall k \) such that \( 1 \leq k \leq \tau \), and \( x^*_0 := x_0, \forall \tau \). In other word, the summation is evaluated along the trajectory generated by applying the time varying switching policy \( v^*(\cdot, \tau - k) \) at time \( k \). Clearly
\[
V^*(x, 0) = 0, \forall x,
\] (13)

and by Bellman equation for fixed-final-time problems, one has [28], [32]
\[
V^*(x, \tau + 1) = \min_{v \in \mathcal{V}} \left( Q(x, v) + \gamma V^*(f_o(x, \tau)) \right),
\] (14)

\forall x, \forall \tau \in \mathbb{M} - \{N\},

and
\[
v^*(x, \tau + 1) = \arg \min_{v \in \mathcal{V}} \left( Q(x, v) + \gamma V^*(f_o(x, \tau)) \right), \forall x, \forall \tau \in \mathbb{M} - \{N\}.
\] (15)

Selecting \( V^0(x) = 0, \forall x \) in initiating VI, and comparing Eq. (14) with (10) it follows that
\[
V^*(x, j) = V^j(x), \forall x, \forall j \in \mathbb{M}.
\] (16)

In other words, the \textit{immature} value function at the \( j \)-th iteration of VI is identical to the \textit{optimal} value function of the fixed-final-time problem of minimizing (11) with the final time of \( j \).

On the other hand, the sequence of functions \( \{V^j(.)\}_{j=0}^{\infty} \) is monotonically non-decreasing. This can be seen by induction, as \( V^0(x) \leq V^j(x), \forall x, \) given \( V^0(x) = 0, \forall x, \) and \( V^1(x) = \min_{v \in \mathcal{V}} Q(x, v), \forall x, \) which follows from (10) after setting \( V^0(x) = 0 \). Assuming \( V^{j-1}(x) \leq V^j(x), \forall x, \) it follows that \( V^j(x) \leq V^{j+1}(x), \forall x, \) because the argument subject to minimization in
\[
V^j(x) = \min_{v \in \mathcal{V}} \left( Q(x, v) + \gamma V^{j-1}(f_o(x)) \right), \forall x \in \mathcal{X},
\] (17)

will be smaller than or equal to that of Eq. (10) at any given \( x \in \mathcal{X} \). This proves the monotonicity of \( \{V^j(.)\}_{j=0}^{\infty} \). This sequence is also upper bounded by the finite cost-to-go of the assumed existing switching policy as each element of the sequence is an optimal cost-to-go given Eq. (16). Hence, the sequence converges to some \( V^\infty(.) = V^*(\cdot) \) as every non-decreasing and upper bounded sequence converges, [33]. It can be seen that \( V^*(\cdot, \infty) = V^*(\cdot) \) by comparing (11) with (5), because, \( J^N \rightarrow J \) as \( N \rightarrow \infty \).

Finally, once the function approximator is trained offline, it may be used for online feedback control. This is done through finding the (approximate) optimal schedule in a feedback form through (9), where \( V^*(\cdot) \) is replaced with the tuned function approximator. This controller will be implemented on the
Before concluding this section it is worth mentioning that each \( V^2(.) \) is desired to be continuous versus its input, in order to guarantee possible uniform approximation [34] using parametric function approximators. Given the switching between different \( \nu \)s in the right hand side of Eq. (10), this continuity is not obvious, even for problems with continuous \( Q(., v) \) and \( f_v(.) \) for any given \( v \). The desired continuity, however, is proved to exist, [25].

### IV. IMPLEMENTATION

Implementation of the scheme in practice will involve an offline training stage, where the parameters of the controller are tuned using Eq. (10). This can be done by a desktop computer ahead of time. After that, the embedded microcontroller in the voltage converter will be only in charge of finding the optimal mode using Eq. (9). The memory requirement includes storage for parameters of the function approximator, e.g., Neural Networks. The computational load will be as low as evaluating two scalar valued functions resulting from plugging \( v = 0 \) and \( v = 1 \) into the term subject to minimization in Eq. (9). Once the two scalar values are found, comparing them the microcontroller simply outputs the minimizing \( v \), which is the optimal mode. In terms of sensors for state feedback, a voltage sensor for measuring \( V_c \) and a current sensor for measuring \( i_l \) will be needed to construct the state vector to be fed into Eq. (9).

#### A. Variables and design choices

The discretization of the continuous-time system (1) is done using Euler integration with the sampling time of \( \Delta t = 0.005s \). The values for the parameters of the circuit are selected based on [20], namely, \( r_c = 0.005\Omega \), \( r_l = 0.05\Omega \), \( L = 3/(2\pi)H \), \( C = 70/(2\pi)F \), \( V_s = 1.8V \), \( i_o = 1A \), and \( V_{ref} = 1V \).

Function \( Q(x, v) = (V_c - V_{ref})^2 \) is selected for the state penalizing term in (5), also \( \gamma = 0.97 \) is selected as a design parameter (one can choose other values). For value function approximation, polynomial functions made of all the possible (non-repeating) combination of the input (i.e., the state elements) up to the fourth order are selected. Using this function approximator, the parameters subject to training are the coefficients of the polynomial terms. The method of least squares, detailed in Appendix of [35], is utilized with selecting 200 random states from \( X = (0, 1.8) \times (-5, 15) \). Domain \( X \) is selected based on the expected operation envelope of the system.

#### B. Simulation

Using the learning iteration given by (10), the evolution of the parameters/weights of the function approximator versus the iterations of the VI is presented in Fig. 2, which shows the desired convergence behavior. The converged values are then used for control of the initial condition \( x_0 := [0.5; 0]^T \), with the resulting state trajectories given in Fig. 3 (the first 15s of the simulation). As seen in this figure, the controller successfully generated the desired voltage across the capacitor, with the steady state error of less than 0.3%. The voltage ripples across the capacitor (not noticeable in the plot) were of the amplitudes of around \( 10^{-5}V \). Given small ripples in the current \( i_l \) (of amplitudes less than \( 10^{-2}A \)) and the very small resistance of the capacitor \( (r_c = 0.005\Omega) \) the ripple in the voltage across the load (which is \( r_c (i_l - i_o) + V_c \)) also turned out to be very small (less than \( 10^{-4}V \) in this case). Considering the switching schedule given in Fig. 3, this desired low ripple output is generated through a high frequency switching, as high as switching once at every sampling time, at some instants. Finally, it should be noted that the relatively slow transient response of the system (also observed in [20]) is due to the selected capacitance and inductance. These values are selected the same as in [20] for facilitating comparison of the results conducted in Section VII.

#### C. Analyzing sensitivity towards line disturbances

In order to evaluate the dynamic performance of the controller towards line disturbances, the source voltage was changed from 1.8V to 1.2V in the last five seconds of the simulation in Fig. 3. Considering the capacitor voltage, no change is observed, which demonstrates the great dynamic performance of the controller. Note that, the controller is trained with the constant \( V_s = 1.8 \). However, in the control stage, using Eq. (9), when \( f_v(.) \)'s are supposed to be evaluated, the new \( V_s = 1.2 \) is used in the last five seconds of the simulation. This leads to a solution which is ‘semi-feedback’ versus \( V_s \), as the parameters of the controller are tuned based on \( V_s = 1.8 \). The suitable performance of the controller in handling the line disturbance leads to the conclusion that the
scheduler has a descent robustness towards such disturbances, as long as the new $V_s$ is measured and used in the scheduler (9). Finally, before concluding this subsection, it is worth mentioning that this change in the $V_s$ is large enough such that if the open loop schedule generated assuming that no change would happen in the voltage source was applied, it would destabilize $V_c$ with oscillations about $0.6V$ with an amplitude as large as $0.5V$. It is the feedback nature of the controller that incorporated this change and provided a new schedule, as presented in Fig. 3, where the change in the switching schedule is noticeable in the last five seconds of the simulation.

D. Decreasing the switching frequency

It was observed in Fig. 3 that the controller led to high frequency switching between the modes (as high as switching once at every sampling time, at some instants). This behavior, however, is not desired, given physical limitations on the switching frequency in practice. An idea for decreasing the number of switching, is using a \textit{threshold}, denoted with $\tau \in \mathbb{R}_+$, in online control, such that, switching from an active mode to another mode is permitted, only if the reward, i.e., the decrease in the right hand side of the scheduler (9), is greater than $\tau$. This is called threshold remedy in [35]. The threshold remedy seems to be similar to having a term for incurring \textit{switching cost} in the cost function, [26]. This change in the cost function, however, will lead to some important changes in the structure of the value function as described in [26] and will give a solution which is optimal with respect to the switching cost as well. The threshold remedy, on the other hand, is much simpler to implement, as no change in the training stage will be done and the threshold will be applied solely in the online control.

Selecting $\tau = 4 \times 10^{-7}$, as an example, the previous case is re-simulated with the results given in Fig. 4. It can be seen that the switching frequency has decreased considerably, compared with Fig. 3, while the desired reference signal is still tracked with barely noticeable ripples (i.e., ripples with amplitude less than $10^{-2}V$ across the capacitor and similar ripples across the load). Note that, even though the threshold was not incorporated in the offline tuning stage, the feedback nature of the controller can handle small thresholds, as they will be considered as some \textit{disturbances} for the feedback controller. However, selecting very large thresholds leads to the degradation of the performance of the controller.

An alternative idea to switching cost or threshold is enforcing a \textit{minimum dwell time}. Again instead of rigorously applying a minimum dwell time in solving the optimal control problem, for example using the theory presented in [36], one may enforce the dwell time after each switch on the fly. This can be done by skipping the evaluation of the scheduler (9) after each switch, until the dwell time (selected as $0.1s$ in here) is elapsed. The result from applying this remedy is presented in Fig. 5. These results are desirable also. It is interesting to note that the robustness towards line disturbances is still retained, as the change in $V_s$ in the last $5s$ (discussed in subsection IV-C) is applied in both Fig. 4 and Fig. 5, with suitable results.

![Fig. 4. Simulation results for $x_0 := [0.5; 0]^T$ with threshold of $4 \times 10^{-7}$ and changing $V_s$ from $1.8V$ to $1.2V$ at $t = 15s$.](image)

![Fig. 5. Simulation results for $x_0 := [0.5; 0]^T$ with minimum dwell time of $0.1s$ and changing $V_s$ from $1.8V$ to $1.2V$ at $t = 15s$.](image)

E. Analyzing sensitivity towards load disturbances

On the topic of dynamic performance of the controller, it is of interest to check the sensitivity of the result to a change in the load current. For this purpose, the simulation is redone, with the difference that instead of changing $V_s$, the $i_o$ is changed from $1A$ to $2A$ at $t = 5s$ and then to $3A$ at $t = 10s$. The results are given in Fig. 6. It can be seen that these changes have led to some steady state errors in $V_c$, namely, around $3\%$ for each $1A$ change in the $i_o$. This change is considerable, compared with the negligible steady state error under the case of no load disturbance, or the case of line disturbances discussed before. Section V presents an idea for improving this behavior, through reducing the sensitivity of the controller to the load current.

F. Analyzing sensitivity towards parameter uncertainties

While the controller is designed based on the assumption that a perfect model of the dynamics of the system is available, one needs to deal with modeling uncertainties in reality. In the DC-DC converter, uncertainties are typically due to the internal resistance of the inductor and capacitor ($r_l$ and $r_c$,
respectively) or the inductance and capacitance $L$ and $C$. The point that the controller calculates the switching on the fly and in a feedback form leads to some interesting results in terms of its robustness towards such uncertainties. To demonstrate this performance, it is assumed that the actual internal resistance of the inductor is ten times its nominal resistance used in the offline training and also in decision making using Eq. (9). Moreover, the actual internal resistance of the capacitor is assumed to be ten times less than the nominal model. Under these assumptions, the simulation with the threshold $\tau = 4 \times 10^{-7}$ is done with the result shown in Fig. 7. It can be seen that the performance of the controller is still suitable in terms of tracking the desired voltage, even under such large parameter uncertainties.

The same simulation is repeated with modeling uncertainties in the inductance and the capacitance, instead of the internal resistances. To this end, the inductance and also the capacitance are assumed to be twice their nominal sizes, used in the training and also in the decision making. The results, depicted in Fig. 8 show that the controller handles these uncertainties well also. Such low sensitivity to parameters is particularly of interest given the fact that parameter uncertainties are unavoidable and in some cases these parameters are unobservable. Interested readers are referred to, for example, [37] where this sensitivity is considerable and some adaptive schemes are sought for lowering it.

Before concluding this subsection it is important to make two points. Firstly, the actual values of the parameters subject to uncertainty were assumed to be unknown in this section. Hence, the nominal values were used not only in the training phase, but also in the decision making and control. Secondly, the controller is not theoretically designed to be robust towards modeling uncertainties. Hence, even though it shows great potentials in this regard, the controller should not be expected to provide reliable results when large modeling uncertainties exist.

G. Incorporating voltage leakage

Each switching between the topologies leads to a voltage loss in reality, [20], [38], [39]. This loss is called voltage leakage and is very challenging to incorporate in the controller design. The reason is, once the leakage is considered, the switching problem converts to a switching problem with state jumps at the switching times, leading to a “more hybrid” problem. Before rigorously incorporating the state jumps in the controller design in Section VI, it is interesting to see the robustness of the already designed controller under the presence of the state jumps.

Motivated by [39], it is assumed that the voltage leakage can be modeled as a function of the voltage across the capacitor immediately before each switching. Let 3% voltage drop be assumed across the capacitor as an example for testing the controller. The state trajectories, given in Fig. 9, show the poor performance of the controller under the state jump. This problem will be addressed later in Section VI.

H. Changing initial condition

The tuned controller handles different initial conditions as long as the respective state trajectories remain within the domain in which the controller is tuned. To show this performance, a few different initial conditions, namely $x_0 = [0, 1]^T$, $x_0 = [1; 0]^T$ with load current increasing for 1.4 after each 5s, without incorporating changing load current in controller design (with threshold $4 \times 10^{-7}$).

![Fig. 6. Simulation results for $x_0 := [1; 0]^T$ with load current increasing for 1.4 after each 5s, without incorporating changing load current in controller design (with threshold $4 \times 10^{-7}$).](image1)

![Fig. 7. Simulation results for $x_0 := [0.5; 0]^T$ with modeling uncertainties in $r_L$ and $r_C$ and changing $V_c$ from 1.8V to 1.2V at $t = 15s$ (with threshold $4 \times 10^{-7}$).](image2)

![Fig. 8. Simulation results for $x_0 := [0.5; 0]^T$ with modeling uncertainties in $L$ and $C$ and changing $V_c$ from 1.8V to 1.2V at $t = 25s$ (with threshold $4 \times 10^{-7}$).](image3)
I. Enforcing a constraint on the inductor current

As seen in the previous simulation results, for example in Fig. 10, the inductor current assumes high values in the transient phase. In practice, however, one is interested in enforcing a constraint on the maximum inductor current, e.g., $i_{l,\text{max}} = 3A$ as in [20], [27]. Similar to [20], this constraint may be enforced as a soft constraint in the cost function using a step-like function that changes from zero to a positive value when $i_l$ exceeds $i_{l,\text{max}}$. Function $(1 + e^{-5(i_l - i_{l,\text{max}})})^{-1}$, which is an approximation of a step function located at $i_{l,\text{max}}$, is selected for this purpose. This function along with the previously selected $Q(x, v)$, scaled with 20 as a weight, are used in the cost function and the tuning is re-done. Given the constraint on $i_l$, the domain of interest is changed to $X = (0, 1.8) \times (-2, 4)$. Using the resulting controller, the initial conditions presented in Fig. 10 are re-simulated and the results are given in Fig. 11. Interestingly, the controller has noticed the constraint on the $i_l$ and has forced the inductor current to observe it, as compared with the results given in Fig. 10 for the case of no constraint.

Finally, as a limitation of the proposed method, it should be mentioned that the scheme applies a soft constraint on the inductor current. While tuning the respective weights in the cost function, it was observed that the desired adjustment on the inductor current was achieved, this behavior is not guaranteed. In order to have such a guarantee, the state inequality constraint needs to be incorporated as a hard constraint in the problem.

V. REDUCING SENSITIVITY TO LOAD CURRENT

The sensitivity of the result to load disturbances, unlike its insensitivity to the line disturbances, both observed in the previous section, shows that the tuned parameters of the controller are more dependent on the selected $i_o$. Note that, when $i_o$ and/or $V_c$ were changed in the analysis of the previous section, the new values were used in the scheduler on the fly, through updating $f_v(.)$’s. However, no retraining based on the new values were done, hence, the less desirable performance in changing $i_o$ compared with changing $V_c$, leads to the conclusion that the parameters of the controller depend more heavily on $i_o$.

Given this observation, the idea in this section is feeding $i_o$ to the function approximator as an exogenous signal, instead of freezing it at its nominal value. Let the value function be denoted with $V^*: \mathbb{R}^3 \times \mathcal{I}_o \to \mathbb{R}^+$, where $\mathcal{I}_o$ denotes the domain in which $i_o$ is expected to change. Moreover, let the notation for the model of the dynamics be redefined as

$$
\dot{x} = f_v(x, i_o) = \begin{bmatrix}
C^{-1}(i_l - i_o) \\
L^{-1}(-V_c - (r_c + r_l)i_l + r_c i_o + V_i v)
\end{bmatrix},
$$

(18)

i.e., the dependency of the dynamics on $i_o$ is explicitly shown in $f_v(x, i_o)$. Of course, the abovementioned continuous-time dynamics should be discretized as in (2), in order to be compatible with the ADP approach investigated in this study.

Given these changes, the Bellman equation reads

$$
V^*(x, i_o) = \min_{v \in V} \left( Q(x, v) + \gamma V^*(f_v(x, i_o), i_o) \right), \forall x, \forall i_o,
$$

(19)

and the optimal mode, in a feedback form versus $i_o$, is given by

$$
v^*(x, i_o) = \arg \min_{v \in V} \left( Q(x, v) + \gamma V^*(f_v(x, i_o), i_o) \right), \forall x, \forall i_o.
$$

(20)

Similarly, in order to find $V^*(x, i_o)$, one may use the VI given by

$$
V^{j+1}(x, i_o) = \min_{v \in V} \left( Q(x, v) + \gamma V^j(f_v(x, i_o), i_o) \right),
$$

(21)

$\forall x \in X, \forall i_o \in \mathcal{I}_o$.

The key point is the fact that Eq. (21) will need to be valid for different $i_o$’s (randomly) selected within $\mathcal{I}_o$. Hence, the converged value function will learn the entire $\mathcal{I}_o$, instead of a single $i_o$, as before. This is similar to the feature that the VI given by (10) was conducted on the entire $X$, instead of on a single $x$, so that the controller is valid for different initial
A. Problem formulation

Considering the optimal switching problem described in Section III, the state jump is added as follows. Let each switching at the current state \( x \) from mode \( v_1 \) to mode \( v_2 \), where \( v_1, v_2 \in \mathcal{V} \) and \( v_1 \neq v_2 \), lead to the (state-dependent) state jump modeled by \( \Delta(x, v_1, v_2) \) for some \( \Delta : \mathbb{R}^n \times \mathcal{V} \times \mathcal{V} \to \mathbb{R}^n \) continuous with respect to \( x \) for every given \( v_1 \) and \( v_2 \) and \( \Delta(x, v_1, v_1) = 0, \forall x, \forall v_1 \). In other words, given the dynamics presented by (4), the current state \( x \), and the previous active mode \( v_1 \), if the same mode is selected for the current time as well, then the next state will be \( f_{v_1}(x) \). But, if mode \( v_2 \) is decided to be selected at this instant, which means switching from \( v_1 \) to \( v_2 \), the next state will be \( f_{v_2}(x) + \Delta(x, v_1, v_2) \).

Given this argument, the next state of the system will be dependent not only on the mode that is selected at the current time, but also on the mode selected at the previous time. Considering this important change compared with the case of no state jump, how is the scheduler, given by (9), supposed to know which mode was selected at the previous time step? Note that the scheduler needs to find the proper next state, at which it needs to evaluate the \( V^*(.) \) in the right hand side of (9). An idea for addressing this problem is presented next.

B. Proposed solution

The idea for addressing the problem of the dependency of \( x_{k+1} \) on \( v_{k-1} \) as well as on \( v_k \) is very intuitive. When the previously selected mode plays a role in determining the next state, then, the previous active mode is also a part of the state of the system. It should be noted that the mode that will be selected at the current time, will be the input to the system and hence, can change the next state. However, just like the rest of the state elements, the previous active mode also matters. So, given the current time \( k \), the state vector \( x_k \) defined in subsection III-A, and the previously selected mode \( v_{k-1} \), the augmented state is defined as \( y_k := [x_k, v_{k-1}]^T \in \mathcal{X} \times \mathcal{V} \). Given Eq. (4), the dynamics of the augmented state vector will be given by

\[
y_{k+1} = F_v(y_k) := \left[ f_v(x_k) + \Delta(x_k, v_{k-1}, v) \right],
\]

(22)

Given (22), the problem is now simply the optimal switching problem defined in subsection III with the only difference that the state vector is \( y \) and the dynamics of modes are given by \( F_v(.) \)'s. Therefore, the same solution, given in subsection III-B is valid, i.e., the VI will be given by

\[
V^{j+1}(y) = \min_{v \in \mathcal{V}} \left( Q(x, v) + \gamma V^j(F_v(y)) \right), \forall y \in \mathcal{X} \times \mathcal{V},
\]

(23)

starting from some \( V^0(y) \), to find the solution to the Bellman equation

\[
V^*(y) = \min_{v \in \mathcal{V}} \left( Q(x, v) + \gamma V^*(F_v(y)) \right), \forall y \in \mathcal{X} \times \mathcal{V}.
\]

(24)
Once the value function $V^*(y)$ is found, the new scheduler will be

$$v^*(y) = \arg\min_{v \in \mathcal{V}} \left( Q(x, v) + \gamma V^*\left(F_v(y)\right) \right), \forall y \in \mathcal{X} \times \mathcal{V}. \tag{25}$$

Now, the question asked at the end of subsection VI-A is addressed. This is done through noting that the scheduler, as given by Eq. (25), will be of a feedback form based on the augmented state $y$, which includes the previously selected mode as well as $x$. Hence, the scheduler will know what mode was selected at the previous time step.

Considering the desired continuity of $V^j(.,.)$, for their uniform approximation, discussed at the end of subsection III-B, the next step is analyzing this continuity. Given the augmented state $y$ and its last element which changes within the discrete set $\mathcal{V}$, the issue is, the value function may not be continuous versus $y$. This problem can be resolved by utilizing $M$ many function approximators, each for approximating $V^*(v, [v])$, for every specific $v \in \mathcal{V}$. This is motivated by the method presented in [26] for incorporating switching costs, which led to a similar feature. However, Ref. [26] incorporates a switching cost, not a state jump. In other words, the current study admits a ’jump in the state’ after each switch while Ref. [26] addresses a ’jump in the cost’ after each switch.

What remains to show is the continuity of $V^*(v, [v])$ in $\mathcal{X}$ for every fixed $v$ which, given the state jump in the dynamics, is not obvious and does not directly follow from [25]. The following theorem proves this desired continuity.

**Theorem 2.** If functions $f_\alpha(.,.), Q(.,.), \Delta(.,., v_1, v_2)$, and the initial guess $V^0(.,[v])$ used in the value iteration (23) are continuous in $\mathcal{X}$ for every fixed $v, v_1$, and $v_2$, then, $V^j(.,[v])$ for every given $v$ and $j$ is continuous in $\mathcal{X}$.

**Proof:** The proof is by induction. Function $V^0(., [v])$, $\forall v$, is continuous. If continuity of $V^j(., [v])$, $\forall v$, leads to the continuity of $V^{j+1}(., [v])$, $\forall v$, then the proof is complete.

Let the scalar function $G : \mathbb{R}^n \times \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}_+$ be defined as

$$G(x, v_1, v_2) := Q(x, v_2) + \gamma V^j(f_\alpha(x) + \Delta(x, v_1, v_2), v_2), \tag{26}$$

and function $v : \mathbb{R}^n \times \mathcal{V} \rightarrow \mathcal{V}$ be given by

$$v(x, v_1) := \arg\min_{w \in \mathcal{V}} G(x, v_1, w). \tag{27}$$

Function $G(x, v_1, v(x, v_1))$ is identical to $V^{j+1}([v])$, considering (23), (26), and (27). Hence, continuity of $G(., v_1, v(., .))$ for every given $v_1$ completes the induction (given the assumed continuity of $V^j(., [v])$).

Let $\bar{x}$ be any point in $\mathcal{X}$, for any given $v_1 \in \mathcal{V}$ set

$$\bar{v} = v(\bar{x}, v_1). \tag{28}$$

Open set $\alpha \subset \mathcal{X}$ may be selected such that $\bar{x}$ belongs to the boundary of $\alpha$ and limit

$$\hat{v} = \lim_{||x - \bar{x}|| \rightarrow 0, x \in \alpha} v(x, v_1), \tag{29}$$

exists, where the vector norm is denoted with $||.||$. If for every such $\alpha$ one has $\bar{v} = \hat{v}$, then an open set $\overline{\beta} \subset \mathcal{X}$ containing $\bar{x}$ exists such that $v(x, v_1)$ is constant for all $x \in \overline{\beta}$, because the range of $v(x, v_1)$ is a discrete set. The continuity of $G(., v_1, v(., .))$ at $\bar{x}$ in this case follows from the continuity of $G(x, v_1, v_2)$ at $x = \bar{x}$, for every fixed/given $v_1, v_2 \in \mathcal{V}$, as $Q(., v_2), f_\alpha(.,.), \Delta(., v_1, v_2)$, and $V^j([v]_2)$ are continuous. Finally, the continuity of $G(., v_1, v(., .))$ at every $\bar{x} \in \mathcal{X}$, leads to the continuity in $\mathcal{X}$.

Now assume, for some $\alpha$, one has $\bar{v} \neq \hat{v}$. The continuity of $G(x, v_1, \bar{v})$ for the fixed $v_1$ and $\bar{v}$ leads to

$$G(\bar{x}, v_1, \bar{v}) = \lim_{\delta x \rightarrow 0} G(\bar{x} + \delta x, v_1, \bar{v}) \tag{30}$$

If one has

$$G(\bar{x}, v_1, \bar{v}) = G(\bar{x}, v_1, \bar{v}), \tag{31}$$

for every selected $\alpha$, then the continuity of $G(., v_1, v(., .))$ for the given $v_1$ follows. The reason is, (30) and (31) lead to

$$G(x, v_1, \bar{v}) = \lim_{\delta x \rightarrow 0} G(\bar{x} + \delta x, v_1, \bar{v}), \tag{32}$$

and (32) gives the continuity by definition [40]. By contradiction, one can show that (31) holds. Assume

$$G(\bar{x}, v_1, \bar{v}) > G(\bar{x}, v_1, \bar{v}), \tag{33}$$

for some $\bar{x}$ and some $\alpha$. The foregoing inequality leads to $v(\bar{x}, v_1) \neq \bar{v}$. But, this is against (28) per the definition of $v(., .)$ given by (27). Hence, (33) does not hold.

Now, assume that

$$G(\bar{x}, v_1, \bar{v}) < G(\bar{x}, v_1, \bar{v}), \tag{34}$$

then, there exists an open set $\gamma$ containing $\bar{x}$, such that

$$G(x, v_1, v) < G(x, v_1, \bar{v}), \forall x \in \gamma. \tag{35}$$

because, both sides of (34) are continuous at $\bar{x}$ for the fixed $\bar{v}$, $\bar{v}$, and $v_1$. Inequality (35) implies that at points which are close enough to $\bar{x}$, one has $v(x, v_1) \neq \bar{v}$. This, however, contradicts Eq. (29) which says that there always exists a point $x$ arbitrarily close to $\bar{x}$ at which $v(x, v_1) = \bar{v}$, considering the point that $v(., .)$ only assumes integer values. Therefore, inequality (35) also cannot hold, hence, (31) holds which leads to the continuity of $G(x, v_1, v(., .))$ at every $x \in \mathcal{X}$ for every fixed $v_1 \in \mathcal{V}$. This completes the proof.

**C. Implementation on DC-DC converter with voltage leakage**

The new controller synthesis which incorporates the voltage leakage modeled as state jumps is numerically analyzed here. Given the number of modes, two function approximators are trained for approximating $V^*(x) [0]$ and $V^*(x) [1]$. The same parameters and variable given in subsection IV-A are
selected, with the difference that the domain of interest is changed to $\mathcal{X} = (0, 1.1) \times (-5, 15)$, for improving the accuracy of the approximation. Moreover, the 3% voltage leakage selected in subsection IV-G is utilized for both the offline training using (23) and for online control using scheduler (25). Note that the 3% is just an example value to show the performance of the controller in handling state jumps. The results are given in Fig. 14. Compared with Fig. 9, it is evident that the controller has successfully incorporated the voltage leakage through adjusting the switching pattern such that the desired $V_{\text{ref}} = 1V$ is (approximately) tracked.

Finally, as a limitation of the proposed method, it may be mentioned that a model of the the voltage leakage needs to be available to be used in the offline training. This is due to the model-based nature of the proposed method. A future effort will be on extending the idea to the data-based case, where such models will be identified implicitly during online learning. Interested readers are referred to [41] for an idea.

VII. RESULTS COMPARISON

The presented method is compared with two other controllers in this section. The selected controllers are the Relaxed Dynamic Programming (RDP) framework in [17] and the Nonlinear Programming (NLP) approach presented in [20]. Identical parameters for the switching circuit given by Fig. 1, identical values for $V_s$, $V_{\text{ref}}$, and the same initial condition are used for the controllers to facilitate the comparison.

For comparing the developed ADP-based controller with the RDP, the simulation presented in [17] with $r_e = 0$, $r_l = 1\Omega$, $L = 0.1H$, $C = 4F$, $V_s = 1V$, $i_o = 0.3A$, and $V_{\text{ref}} = 0.5V$ was selected. The controller developed in Section IV, i.e., without feeding load current into the controller, was used for controlling the system. The results are presented in Fig. 15 where the load current was changed from the initial value of 0.3A to 0.1A and then to $-0.2A$ and back to 0.3A, where each current was applied for 10s. Comparing the ADP result with the RDP one (copied from [17]), it can be seen that the proposed method demonstrates a better performance even without feeding the load current to the network and also without defining an integral state for penalizing the tracking error as conducted in [17]. It is worth mentioning that the large sampling time of $\Delta t = 0.1s$, used in [17], was used here in the proposed ADP-based controller also, which led to higher voltage ripples compared with the results presented in previous sections.

Next, the simulation results presented in [20] are compared with the results in this work. Similar soft constraints on the current are applied and starting with the same initial condition, the capacitor voltage and inductor current are plotted in Fig. 16. This figure also shows that the proposed method compares favorably with the one presented in [20].

VIII. CONCLUSIONS

Approximate dynamic programming was shown to have interesting potentials in handling different challenges existing in the problem of switching in a voltage converter circuit. Sensitivity towards line and load disturbances as well as parameter uncertainties were analyzed. Constraints on the inductor current and the voltage leakage of the capacitor are some other challenges investigated in this study, in which, the proposed tool either inherently provided a suitable performance or the changes made in the controller addressed them. The proposed tools and methods including the controller for optimal switching with state jump may be utilized by different researchers and practitioner for different applications involving state jumps. Given the versatility of the proposed scheme, it may also be utilized for different power converters including switched-capacitor DC-DC converters with more than two topologies as well as DC-AC converters. Finally, the results presented in this work solely show the performance of the controller in simulations. Real implementation could reveal other challenges and is subject of future research.

REFERENCES
