Reinforcement Learning Versus Model Predictive Control: A Comparison on a Power System Problem

Damien Ernst, *Member, IEEE*, Mevludin Glavic, *Senior Member, IEEE*, Florin Capitanescu, and Louis Wehenkel, *Member, IEEE*

***Abstract*—This paper compares reinforcement learning (RL) with model predictive control (MPC) in a unified framework and reports experimental results of their application to the synthesis of a controller for a nonlinear and deterministic electrical power oscillations damping problem. Both families of methods are based on the formulation of the control problem as a discrete-time optimal control problem. The considered MPC approach exploits an analytical model of the system dynamics and cost function and computes open-loop policies by applying an interior-point solver to a minimization problem in which the system dynamics are represented by equality constraints. The considered RL approach infers in a model-free way closed-loop policies from a set of system trajectories and instantaneous cost values by solving a sequence of batch-mode supervised learning problems. The results obtained provide insight into the pros and cons of the two approaches and show that RL may certainly be competitive with MPC even in contexts where a good deterministic system model is available.**

***Index Terms*—Approximate dynamic programming (ADP), electric power oscillations damping, fitted Q iteration, interior–point method (IPM), model predictive control (MPC), reinforcement learning (RL), tree-based supervised learning (SL).**

I. Introduction

MANY control problems can be formalized under the form of optimal control problems having discrete-time dynamics and costs that are additive over time. Model predictive control (MPC) and reinforcement learning (RL) are two different approaches to solve such problems. MPC was originally designed to exploit an explicitly formulated model of the process and solve in a receding horizon manner a series of open-loop deterministic optimal control problems [1], [2]. The main motivation behind the research in MPC was initially to find ways to stabilize large-scale systems with constraints around some equilibrium points (or trajectories) [3]. RL was designed to infer closed-loop policies for stochastic optimal control problems from a sample of trajectories gathered from interaction with the real system or from simulations [4], [5]. This field was initially derived from the psychological theory of the same name that was studying how an agent ought to learn to take actions in an environment to maximize some long-term reward signals. Contrary to the research carried out in MPC, the emphasis in RL has not been put on the stability properties of the control policies, but on other aspects such as the learning speed, the stability of the learning process itself, the scaling properties of the algorithms, or the design of strategies for generating rapidly informative trajectories [6].

While RL, like stochastic dynamic programming (DP), has in principle a very broad scope of application, it is similarly challenged when the state space and/or action spaces of the control problem are very large or continuous. In such a case, RL has to be combined with techniques allowing one to generalize over the state-action space the data contained in the typically very sparse sample of trajectories. Over the last two decades, most of the research in this context has focused on the use of parametric function approximators, representing either some (state-action) value functions or parameterized policies, together with some stochastic gradient descent algorithms [7]–[10]. Even if some successes have been reported (e.g., [11]–[14]), these techniques have not yet moved from the academic to the real world as successfully as MPC techniques, which have already been largely adopted in practice [15].

The problem of generalization over an information space is not unique to RL and also occurs in the batch-mode supervised learning (SL) framework. A batch-mode SL algorithm considers a sample of input–output pairs, with the input being an information state and the output a class-label or a real number, and induces from the sample a model which explains at best these input–output pairs. Examples of SL algorithms are neural networks [16], methods based on kernels such as support vector machines [17], [18] or tree-based methods [19], [20]. While generalization strategies used in RL were struggling to cope with spaces of even modest sizes, batch-mode SL algorithms have been successfully applied to real-life problems with extremely large information spaces, such as those in which an element is described by several thousands of components [17], [20].

Therefore, it was in some sense natural for researchers from the RL community to start investigating whether they could exploit state-of-the-art batch-mode SL algorithms to solve their generalization problem.

Since the beginning of the year 2000, one has seen the emergence of new RL algorithms whose main characteristic is to solve iteratively a sequence of batch-mode SL regression problems. They are inspired by the DP principle which gives a

Manuscript received March 27, 2007; revised December 19, 2007 and March 18, 2008. First published December 16, 2008; current version published March 19, 2009. The work of D. Ernst was supported by the Belgian National Fund for Scientific Research (FNRS). This paper was recommended by Associate Editor M. Huber.

D. Ernst is with the Belgian National Fund for Scientific Research, 1000 Brussels, Belgium, and also with the Department of Electrical Engineering and Computer Science, University of Liège, 4000 Liège, Belgium (e-mail: ernst@montefiore.ulg.ac.be).

M. Glavic, F. Capitanescu, and L. Wehenkel are with the Department of Electrical Engineering and Computer Science, University of Liège, 4000 Liège, Belgium (e-mail: glavic@montefiore.ulg.ac.be; capitane@montefiore.ulg.ac.be; wehenkel@montefiore.ulg.ac.be).

Digital Object Identifier 10.1109/TSMCB.2008.2007630

way to solve optimal control problems by iteratively extending their optimization horizon [21]. More precisely, they solve iteratively a sequence of regression problems by mimicking the behavior of the classical value iteration algorithm from the DP theory. These algorithms differ by the type of SL methods considered (e.g., kernel-based regressors in [22], artificial neural networks in [23], and mainly tree-based methods in [24]), and they can be seen as particular instances of the general fitted Q iteration algorithm introduced in [25]. As shown by Riedmiller [23] and Ernst *et al.* [24], the fitted Q iteration algorithm outperforms other popular RL algorithms on several nontrivial problems. These two papers and some other works published by Riedmiller [26] and Ernst *et al.* [28] highlight also that the fitted Q iteration algorithm can infer, even for some high-dimensional problems, good policies from relatively small samples of trajectories, which suggests that it may pave the way to many successful applications of RL to real-life problems.

This paper presents in the deterministic case and when a model of the system and cost functions are available, the MPC approach and the fitted Q iteration algorithm in a unified framework and compares simulation results obtained by them on a (nonlinear) optimal control problem of electric power system oscillations damping. The test problem was chosen sufficiently complex to be nontrivial, and at the same time sufficiently simple to lend itself to detailed analysis and to avoid simplifications while applying MPC to it. In particular, we did not consider neither uncertainties nor disturbances and avoided discrete states to compare RL with MPC in the latter's original field of application.

This paper is organized as follows. In Section II, the type of optimal control problems considered is defined, some results from the DP theory recalled and explanations on how the MPC and the RL approaches address this type of problem in the finite horizon case are given. Section III considers the case of large or infinite horizons and shows that both MPC and RL can tackle these problems by truncating the optimization horizon. The section provides also a characterization of the policies both methods target in this way and gives an upper bound on their suboptimality. Section IV presents in details the application of these two approaches to the electric power system oscillations damping problem. Section V elaborates on to what extend the qualitative nature of the results obtained could be extended to other classes of problems (e.g., linear, stochastic). Finally, Section VI concludes.

II. MPC AND RL IN THE FINITE HORIZON CASE

A. Optimal Control Problem

Consider a discrete-time system whose dynamics over T stages is described by a time-invariant equation

$$x_{t+1} = f(x_t, u_t), \qquad t = 0, 1, \ldots, T-1 \tag{1}$$

where for all t, the state x_t is an element of the state space X and the action u_t is an element of the action space U. $T \in \mathbb{N}_0$ is referred to as the *optimization horizon*.

The transition from t to $t+1$ is associated with an instantaneous cost signal $c_t = c(x_t, u_t) \in \mathbb{R}$ which is assumed to be bounded by a constant B_c, and for every initial state x_0 and for every sequence of actions, the discounted cost over T stages is defined as

$$C_T^{(u_0, u_1, \ldots, u_{T-1})}(x_0) = \sum_{t=0}^{T-1} \gamma^t c(x_t, u_t) \tag{2}$$

where $\gamma \in [0, 1]$ is the discount factor.

In this context, an optimal control sequence $u_0^*, u_1^*, \ldots, u_{T-1}^*$, is a sequence of actions that minimizes the cost over T stages.[1]

Within the general class of deterministic, time varying and nonanticipating control policies, namely, policies $\pi = (\pi_0, \pi_1, \ldots, \pi_{T-1})$ which given a sequence of states $x_0, \ldots, x_t$ provide a control action $u_t = \pi_t(x_0, \ldots, x_t)$, we focus on three subclasses: *open-loop* policies which select at time t the action u_t based only on the initial state x_0 of the system and the current time $(u_t = \pi_o(t, x_0))$, *closed-loop* policies which select the action u_t based on the current time and the current state $(u_t = \pi_c(t, x_t))$, and closed-loop *stationary* policies for which the action is selected only based on the knowledge of the current state $(u_t = \pi_s(x_t))$.

Let $C_T^\pi(x_0)$ denote the cost over T stages associated with a policy π when the initial state is x_0. A *T-stage optimal policy* is by definition a policy that leads for every initial state x_0 to a sequence of actions which minimizes $C_T^\pi(x_0)$. This is the kind of policy we are looking for.

To characterize optimality of T-stage policies, let us define iteratively the sequence of *state-action value functions* $Q_N : X \times U \to \mathbb{R}, N = 1, \ldots, T$ as follows:

$$Q_N(x, u) = c(x, u) + \gamma \inf_{u' \in U} Q_{N-1}(f(x, u), u') \tag{3}$$

with $Q_0(x, u) = 0$ for all $(x, u) \in X \times U$.

We have the following two theorems (see, e.g., [29] for the proofs):

Theorem 1: A sequence of actions $u_0^*, u_1^*, \ldots, u_{T-1}^*$ is optimal if and only if $Q_{T-t}(x_t, u_t^*) = \inf_{u' \in U} Q_{T-t}(x_t, u') \forall t \in \{0, \ldots, T-1\}$.

Theorem 2: If $u_0^*, u_1^*, \ldots, u_{T-1}^*$ is an optimal sequence of actions, then $C_T^{(u_0^*, u_1^*, \ldots, u_{T-1}^*)}(x_0) = \inf_{u \in U} Q_T(x_0, u)$.

Under various sets of additional assumptions (e.g., U finite or see [30] when U is infinite), the existence of an optimal closed-loop (or open-loop) policy which is a T-stage optimal policy is guaranteed. The notation $\pi_{c,T}^*$ (or $\pi_{o,T}^*$) is used to refer to a closed-loop (or open-loop) T-stage optimal policy. From Theorem 1, we see that every policy $\pi_{c,T}^*$ is such that $\pi_{c,T}^*(x, t) \in \arg\inf_{u \in U} Q_{T-t}(x, u)$. Similarly, for every policy $\pi_{o,T}^*$, we have $\pi_{o,T}^*(x_0, t) \in \arg\inf_{u \in U} Q_{T-t}(x_t, u)$ with $x_t =$

[1] This problem statement does not explicitly consider constraints other than those implied by the system dynamics. However, constraints on the input and/or the state can be modeled in this formulation by penalizing the cost function in an appropriate way. The reader is referred to Section IV on experiments to see how we have penalized the cost function to take into account constraints on the state imposed by some stability issues.

$f(x_{t-1}, \pi^*_{\mathrm{o},T}(x_0, t-1))$, for all $t = 1, \ldots, T-1$. Note also that, in general, a T-stage stationary policy which is optimal does not exist.

B. MPC

In their original setting, MPC techniques target an optimal open-loop policy $\pi^*_{\mathrm{o},T}$, and assume that the system dynamics and cost function are available in analytical form.

For a given initial state x_0, the terms $\pi^*_{\mathrm{o},T}(x_0, t), t = 0, 1, \ldots, T-1$ of the optimal open-loop policy may then be computed by solving the minimization problem

$$\inf_{\substack{(u_0, u_1, \ldots, u_{T-1}, x_1, x_2, \ldots, x_{T-1}) \\ \in U \times \cdots \times U \times X \times \cdots \times X}} \sum_{t=0}^{T-1} \gamma^t c(x_t, u_t) \tag{4}$$

subject to the T equality constraints (1).[2]

Under appropriate assumptions, the minimization problem (4) can be tackled by classical convex programming algorithms. However, for many practical problems, its resolution may be a difficult task, with no guarantees that the solution found by the used optimizer is indeed optimal. Moreover, the model of the process may not represent perfectly well the real system which may lead to some additional suboptimalities. Therefore, the MPC techniques actually rather produce an approximation $\hat{\pi}^*_{\mathrm{o},T}$ of a T-stage optimal control policy. The better the approximation, the smaller the error $C_T^{\hat{\pi}^*_{\mathrm{o},T}}(x_0) - C_T^{\pi^*_{\mathrm{o},T}}(x_0)$. To mitigate the effect of such suboptimalities and to improve robustness with respect to disturbances, an MPC scheme usually solves at every instant t a minimization problem similar to the one described by (4), but where the optimization horizon is $T - t$, and the initial state is x_t. Then, from the (approximate) solution, it derives the action $\hat{\pi}^*_{\mathrm{o},T-t}(x_t, 0)$ and applies it at time t [i.e., $u_t = \hat{\pi}^*_{\mathrm{o},T-t}(x_t, 0)$].

C. Learning From a Sample of Trajectories

Let us now consider the problem of learning from a sample of observations, assuming that the system dynamics and cost function are not given in analytical (or even algorithmic) form. Thus, the sole information assumed to be available about the system dynamics and cost function is the one that can be gathered from the observation of system behaviors in the form: $(x_0, u_0, c_0, x_1, \ldots, c_j, x_{j+1})$.

Since, except for very special conditions, the exact optimal policy cannot be decided from such a limited amount of information, RL techniques compute from this an *approximation* of a T-stage optimal policy, expressed in closed-loop form $\hat{\pi}^*_{\mathrm{c},T}$.

The fitted Q iteration algorithm on which we focus in this paper, actually relies on a slightly weaker assumption, namely, that a set of one step system transitions is given, each one providing the knowledge of a new sample of information (x_t, u_t, c_t, x_{t+1}) named four-tuple.

Let us denote by $\mathcal{F}$ the set $\{(x_t^l, u_t^l, c_t^l, x_{t+1}^l)\}_{l=1}^{|\mathcal{F}|}$ of available four-tuples. Fitted Q iteration computes from $\mathcal{F}$ the functions $\hat{Q}_1, \hat{Q}_2, \ldots, \hat{Q}_T$, approximations of the functions $Q_1, Q_2, \ldots, Q_T$ defined by (3), by solving a sequence of batch-mode SL problems. From these, a policy which satisfies

$$\hat{\pi}^*_{\mathrm{c},T}(t, x) \in \arg\inf_{u \in U} \hat{Q}_{T-t}(x, u)$$

is taken as approximation of an optimal control policy.

Posing $\hat{Q}_0(x, u) = 0$, for all $(x, u) \in X \times U$, the training set for the Nth SL problem of the sequence $(N \geq 1)$ is

$$\left\{ (i^l, o^l) = \left(\left(x_t^l, u_t^l\right), c_t^l + \gamma \inf_{u \in U} \hat{Q}_{N-1}\left(x_{t+1}^l, u\right) \right) \right\}_{l=1}^{|\mathcal{F}|} \tag{5}$$

where the i^l (respectively o^l) denote inputs (respectively outputs). The SL regression algorithm produces from the sample of inputs and outputs the function $\hat{Q}_N$.

Under some conditions on the system dynamics, the cost function, the SL algorithm, and the sampling process used to generate $\mathcal{F}$, the fitted Q iteration algorithm is consistent, i.e., is such that every function $\hat{Q}_N$ converges to Q_N when $|\mathcal{F}|$ grows to infinity [22].

III. Time Horizon Truncation

Let $0 < T' < T$ and let $\pi^*_{\mathrm{s},T'}$ denote a stationary policy such that $\pi^*_{\mathrm{s},T'}(x) \in \arg\inf_{u \in U} Q_{T'}(x, u)$. We have the following theorem (proof in Appendix A).

Theorem 3: The suboptimality of $\pi^*_{\mathrm{s},T'}$ with $T' < T$ used on a T-stage optimal control problem is bounded by

$$\sup_{x \in X} \left(C_T^{\pi^*_{\mathrm{s},T'}}(x) - C_T^{\pi^*_T}(x) \right) \leq \frac{\gamma^{T'}(4 - 2\gamma) B_{\mathrm{c}}}{(1-\gamma)^2} \tag{6}$$

where π^*_T denotes a T-stage optimal control policy.

Theorem 3 shows that the suboptimality of the policy $\pi^*_{\mathrm{s},T'}$ when used as solution of an optimal control problem with an optimization horizon T $(T > T')$ can be upper bounded by an expression which decreases exponentially with T'.

When dealing with a large or even infinite optimization horizon T, the MPC and RL approaches use this property to truncate the time horizon to reduce computational burdens. In other words, they solve optimal control problems with a truncated time horizon T' $(T' < T)$; the obtained solution yields a policy, which approximates the true optimal policy and is used to control the system.

In the MPC approach, such $\hat{\pi}^*_{\mathrm{s},T'}$ policies are computed by using a time receding horizon strategy. More precisely, at every instant t, the MPC scheme solves a minimization problem similar to the one described by (4) but where the optimization horizon is T' and the initial state is x_t. Then, from the (approximate) solution, it derives the action $u_t = \hat{\pi}^*_{\mathrm{o},T'}(x_t, 0)$. Since

[2] In the traditional MPC formulation, besides equality constraints describing the dynamics of the system, inequality constraints on inputs and states are often included. We have chosen here for easing the presentation of both approaches in a unified framework to consider that these constraints on the states and inputs are included through an appropriate penalization of the cost function. However, when solving the minimization problem, it may be convenient to state explicitly the inequality constraints to get a functional shape for the cost criterion which can be more easily exploited by an optimizer. Such a strategy is used in our numerical example (Section IV).

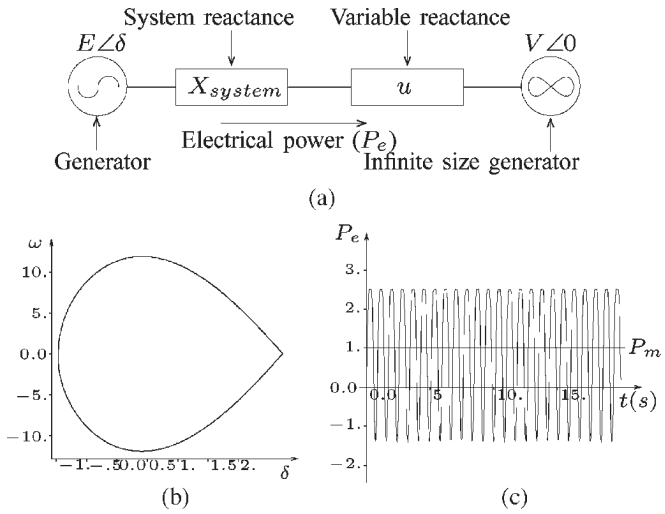

Fig. 1. Some insights into the control problem. (a) Representation of the power system. (b) Stability domain $X_{\rm S}$ of uncontrolled system. (c) Electrical power oscillations. When $(\delta_0, \omega_0) = (0, 8)$ and $u \equiv 0$.

$\pi_s(x) = \pi^*_{{\rm o},T'}(x, 0)$ is a $\pi^*_{{\rm s},T'}$ policy, the action chosen may also be seen as being selected by using a $\hat{\pi}^*_{{\rm s},T'}$ policy.

In the RL approach, the $\hat{\pi}^*_{{\rm s},T'}$ policy is computed by iterating the fitted Q iteration algorithm only T' times and taking $\hat{\pi}^*_{{\rm s},T'}(x, u) \in \arg\inf_{u \in U} \hat{Q}_{T'}(x, u)$ as policy to control the system.

The interested reader is referred to [31] for similar suboptimality bounds for various other MPC and approximate DP (ADP) schemes, in particular in the time-varying and finite horizon case.

IV. Simulation Results

This section presents simulation results obtained by using the two approaches on a test problem. The control problem and the experimental protocol are described in detail to allow reproduction of these results.

A. Control Problem Statement

We consider the problem of controlling the academic benchmark electric power system shown in Fig. 1(a) in order to damp electrical power oscillations. More information about the physics of this power system control problem can be found in [32] which provides also results obtained by using a control Lyapunov function approach to synthesize the controller. References [33] and [24] report results obtained by several RL algorithms (Q-learning, model-based, kernel-based, and fitted Q iteration) on a similar problem. We note that in these experiments, the fitted Q iteration algorithm led consistently to the best performances. The reader is referred to [34] for an account of techniques nowadays used to damp electrical power oscillations.

The system is composed of a generator connected to a machine of infinite size (whose inertia and short-circuit power are large enough for its speed and terminal voltage to be assumed constant [35]) through a transmission line, with a variable reactance u installed in series. The system has two state variables: the angle δ and the speed ω of the generator. Their dynamics, assuming a simple second-order generator model, are given by the differential equations

$$\dot{\delta} = \omega$$
$$\dot{\omega} = \frac{P_{\rm m} - P_{\rm e}}{M} \quad \text{with}$$
$$P_{\rm e} = \frac{EV}{u + X_{\rm system}} \sin\delta$$

where $P_{\rm m}$, M, E, V, and $X_{\rm system}$ are parameters equal, respectively, to 1, 0.03183, 1, 1, and 0.4 p.u. The symbol p.u. stands for "per unit." In the field of power transmission, a per-unit system is the expression of system quantities as fractions of a defined base unit quantity. Calculations are simplified because quantities expressed as per-unit are the same regardless of the voltage level.

$P_{\rm m}$ represents the mechanical power of the machine, M its inertia, E its terminal voltage, V the voltage of the terminal bus system, and $X_{\rm system}$ the overall system reactance.

Although the system dynamics is defined whatever the value of δ and ω, we limit the control problem state space to the stability domain of the nominal stable equilibrium of the uncontrolled ($u \equiv 0$) system, defined by

$$(\delta_{\rm e}, \omega_{\rm e}) = \left(\arcsin\frac{X_{\rm system} P_{\rm m}}{EV}, 0\right) = (0.411, 0) \qquad (7)$$

to which corresponds an electrical power transmitted in the line equal to $P_{\rm m}$. The separatrix of the stability domain $X_{\rm S}$ is shown in Fig. 1(b), and the stability domain is defined by (see [36] and [37] for more information)

$$X_{\rm S} = \left\{(\delta, \omega) : \frac{1}{2}M\omega^2 - P_{\rm m}\delta - \frac{EV\cos(\delta)}{X_{\rm system}} \leq -0.439\right\}.$$

When the uncontrolled system is perturbed, undamped electrical power ($P_{\rm e}$) oscillations appear in the line [Fig. 1(c)]. Acting on the variable reactance u allows one to influence the power flow through the line, and the control problem consists of finding a policy for u to damp the electrical power oscillations. From this continuous time control problem, a discrete-time one of infinite horizon is defined such that policies leading to small costs also tend to produce good damping of $P_{\rm e}$.

The discrete-time dynamics is obtained by discretizing time with a step of 0.050 s. The state of the system is observed at these discrete time steps, and the value of u is allowed to change only at these time steps, and is constrained to belong to the interval $U = [-0.16, 0]$. If δ_{t+1} and ω_{t+1} do not belong to the stability domain $X_{\rm S}$ of the uncontrolled system then a *terminal state* is supposed to be reached, which is denoted by $x = x_\perp$. This is a state in which the system remains stuck, i.e., $x_\perp = f(x_\perp, u)$, for all $u \in U$. The state space X of the discrete time optimal control problem is thus composed of the uncontrolled stability domain $X_{\rm S}$ plus this (undesired) terminal state $x_\perp$ (i.e., $X \triangleq X_{\rm S} \cup \{x_\perp\}$).

The cost function $c(x, u)$ should penalize deviations of the electrical power from its steady-state value ($P_{\rm e} = P_{\rm m}$), and

ensure that the system remains inside the stability domain. The following cost function is thus used:

$$c(x_t, u_t) = \begin{cases} 0, & \text{if } x_t = x_{t+1} = x_\perp \\ 1000, & \text{if } x_t \in X_{\mathrm{S}} \text{ and } x_{t+1} = x_\perp \\ (P_{\mathrm{e}t+1} - P_{\mathrm{m}})^2, & x_t \in X_{\mathrm{S}} \text{ and } x_{t+1} \in X_{\mathrm{S}} \end{cases} \tag{8}$$

where $P_{\mathrm{e}_{t+1}} = (EV/X_{\mathrm{system}} + u_t)\sin(\delta_{t+1})$. There is no penalization of the control efforts in the cost function [e.g., no term of the type $\|u_t\|$ in $c(x_t, u_t)$], contrary to what is usually done in MPC to avoid the controller to switch too rapidly between extreme values of the control variables.

The decay factor γ ($\gamma = 0.95$) has been chosen close to one in order to ensure that the discounted costs do not decay too rapidly with time, in comparison with the time constant of the system oscillations. With this value, γ^t reaches a value of 10% after 45 time steps, i.e., after 2.25 s of real time, which is about two to three times larger that the natural oscillation period of the system [see Fig. 1(c)].

The value of 1000 penalizing the first action leading to the terminal state $x_\perp$ guarantees that policies moving the system outside of X_{S} are suboptimal, whatever the horizon T. Indeed, suppose that a policy π_1, starting from $x_0 \in X_{\mathrm{S}}$, reaches $x_\perp$ for the first time at $t+1$. Then, the open-loop policy π_2 with $\pi_2(x_0, t') = \pi_1(x_0, t')$, for all $t' = 0, \ldots, t-1$ and $\pi_2(x_0, t') = 0$, for all $t' = t, \ldots, T-1$ would keep the system inside X_{S}. Thus, π_2 would hence yield a (strictly) lower cost than π_1, since [see (8)]

$$1000 > (1-\gamma)^{-1}\left(-\frac{EV}{X_{\mathrm{system}}} - P_{\mathrm{m}}\right)^2$$

where $(-(EV/X_{\mathrm{system}}) - P_{\mathrm{m}})^2$ represents an upper bound on the instantaneous costs nonrelated to the exit of the system from the stability domain. Thus, π_1 cannot be an optimal policy.

Note also that, although the cost at time t is formulated in terms of both u_t and x_{t+1}, it can actually be expressed as a function of x_t and u_t only, since x_{t+1} can be expressed as a function of x_t and u_t.

We introduce also a set X_{test} which is going to be used later in this paper as a set of "test states"

$$X_{\mathrm{test}} = \{(\delta, \omega) \in X_{\mathrm{S}} | i, j \in \mathbb{Z}, (\delta, \omega) = (0.1 * i, 0.5 * j)\}.$$

B. Application of RL

1) Four-Tuples Generation: To collect the four-tuples, 100 000 one-step episodes have been generated with x_0 and u_0 for each episode drawn at random in $X_{\mathrm{S}} \times U$. In other words, starting with an empty set $\mathcal{F}$, the following sequence of instructions has been repeated 100 000 times:

1) draw a state x_0 at random in X_{S};
2) draw an action u_0 at random in U;
3) apply action u_0 to the system initialized at state x_0, and simulate[3] its behavior until $t = 1$ (0.050 s later);
4) observe x_1 and determine the value of c_0;
5) add (x_0, u_0, c_0, x_1) to the set of four-tuples $\mathcal{F}$.

2) Fitted Q Iteration Algorithm: The fitted Q iteration algorithm computes $\hat{\pi}_{\mathrm{s},T'}$ by solving sequentially T' batch-mode SL problems. As an SL algorithm, the Extra-Trees method is used [20]. This method builds a model in the form of the average prediction of an ensemble of randomized regressions trees. Its three parameters, the number M of trees composing the ensemble, the number $n_{\min}$ of elements required to split a node, and the number K of cut-directions evaluated at each node, have been set, respectively, to 50, 2 (fully developed trees), and the dimensionality of the input space (equal to three for our problem: two state variables +1 control variable). The choice of Extra-Trees is justified by their computational efficiency, and by the detailed study of [24] which shows on various benchmark problems that Extra-Trees obtain better results in terms of accuracy than a number of other tree-based and kernel-based methods.

To approximate the value of $\inf_{u \in U} \hat{Q}(x^l_{t+1}, u)$, whenever needed, the minimum over the 11 element subset

$$U' = \{0, -0.016 * 1, -0.016 * 2, \ldots, -0.16\} \tag{9}$$

is computed.

C. Application of MPC

1) Nonlinear Optimization Problem Statement: Two main choices have been made to state the optimization problem. First, the cost of 1000 used earlier to penalize trajectories leaving the stability domain of the uncontrolled system was replaced by equivalent inequality constraints [see below, (15)]. Second, the equality constraints $x_{t+1} = f(x_t, u_t)$ are modeled by relying on a trapezoidal method, with a step size of 0.05 s, the time between t and $t+1$ [see below, (11) and (12)]. Contrary to the previous one, this choice may lead to some suboptimalities.

Denoting $(\delta_1, \ldots, \delta_{T'}, \omega_1, \ldots, \omega_{T'}, u_0, \ldots, u_{T'-1})$ by $\mathbf{x}$ and the step size by h, the problem states as

$$\min_{\mathbf{x}} \sum_{t=0}^{T'-1} \gamma^t \left(\frac{EV}{X_{\mathrm{system}} + u_t}\sin(\delta_{t+1}) - P_{\mathrm{m}}\right)^2 \tag{10}$$

subject to $2T'$ equality constraints ($t = 0, 1, \ldots, T'-1$)

$$\delta_{t+1} - \delta_t - (h/2)\omega_t - (h/2)\omega_{t+1} = 0 \tag{11}$$

$$\omega_{t+1} - \omega_t - (h/2)\frac{1}{M}\left(P_{\mathrm{m}} - \frac{EV\sin\delta_t}{u_t + X_{\mathrm{system}}}\right) - (h/2)\frac{1}{M}\left(P_{\mathrm{m}} - \frac{EV\sin\delta_{t+1}}{u_t + X_{\mathrm{system}}}\right) = 0 \tag{12}$$

and $3T'$ inequality constraints ($t = 0, 1, \ldots, T'-1$)

$$u_t \leq 0 \tag{13}$$

$$-u_t \leq 0.16 \tag{14}$$

$$\frac{1}{2}M\omega^2_{t+1} - P_{\mathrm{m}}\delta_{t+1} - \frac{EV}{X_{\mathrm{system}}}\cos(\delta_{t+1}) + 0.439 \leq 0. \tag{15}$$

[3]To determine the behavior of the power system we have used the trapezoidal method with 0.005-s step size.

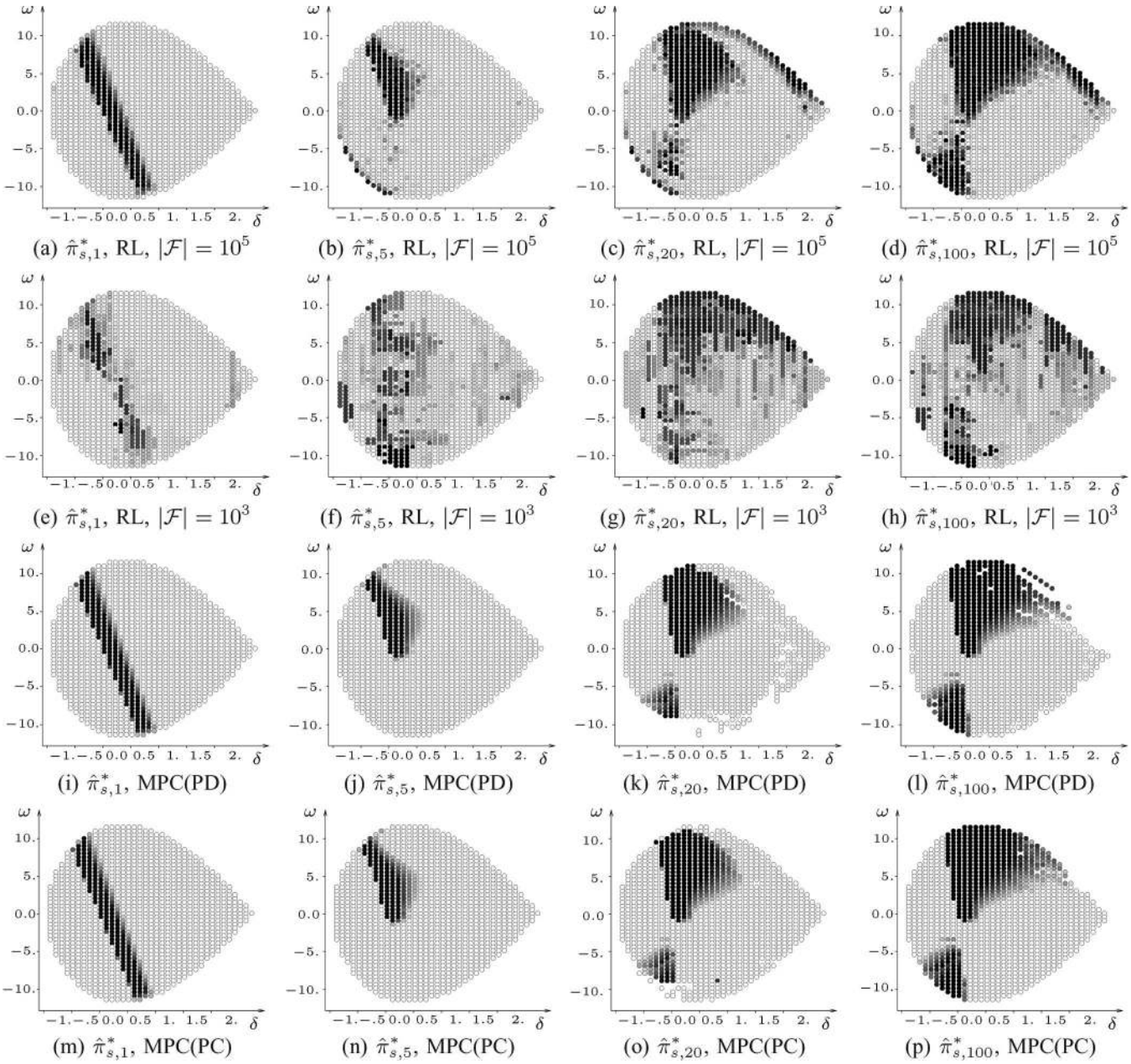

Fig. 2. Representation of the policies $\hat{\pi}^*_{\mathrm{s},T'}(x)$. (a)–(d) gives for different values of T' the policies $\hat{\pi}^*_{\mathrm{s},T'}(x)$ obtained by the RL algorithm with 100 000 four-tuples, (e)–(h) with the RL algorithm with 1000 four-tuples, and (i)–(l) [respectively (m)–(p)] with the MPC(PD) [respectively MPC(PC)] algorithm. (a) $\hat{\pi}^*_{\mathrm{s},1}$, RL, $|\mathcal{F}| = 10^5$. (b) $\hat{\pi}^*_{\mathrm{s},5}$, RL, $|\mathcal{F}| = 10^5$. (c) $\hat{\pi}^*_{\mathrm{s},20}$, RL, $|\mathcal{F}| = 10^5$. (d) $\hat{\pi}^*_{\mathrm{s},100}$, RL, $|\mathcal{F}| = 10^5$. (e) $\hat{\pi}^*_{\mathrm{s},1}$, RL, $|\mathcal{F}| = 10^3$. (f) $\hat{\pi}^*_{\mathrm{s},5}$, RL, $|\mathcal{F}| = 10^3$. (g) $\hat{\pi}^*_{\mathrm{s},20}$, RL, $|\mathcal{F}| = 10^3$. (h) $\hat{\pi}^*_{\mathrm{s},100}$, RL, $|\mathcal{F}| = 10^3$. (i) $\hat{\pi}^*_{\mathrm{s},1}$, MPC(PD). (j) $\hat{\pi}^*_{\mathrm{s},5}$, MPC(PD). (k) $\hat{\pi}^*_{\mathrm{s},20}$, MPC(PD). (l) $\hat{\pi}^*_{\mathrm{s},100}$, MPC(PD). (m) $\hat{\pi}^*_{\mathrm{s},1}$, MPC(PC). (n) $\hat{\pi}^*_{\mathrm{s},5}$, MPC(PC). (o) $\hat{\pi}^*_{\mathrm{s},20}$, MPC(PC). (p) $\hat{\pi}^*_{\mathrm{s},100}$, MPC(PC).

2) Nonlinear Optimization Solvers: Two interior-point method (IPM) algorithms are used in our simulations: the pure primal-dual [38] and the predictor-corrector [39]. They are denoted by MPC(PD) and MPC(PC), respectively.

D. Discussion of Results

1) Results of RL: Fig. 2(a)–(d) shows the policy $\hat{\pi}_{\mathrm{s},T'}$ computed for increasing values of T'. The representation has been carried out by plotting bullets centered on the different elements $x \in X_{\mathrm{test}}$. The color (gray level) of a bullet centered on a particular state x gives information about the magnitude of $|\hat{\pi}^*_{\mathrm{s},T'}(x)|$. Black bullets correspond to the largest possible value of $|\hat{\pi}^*_{\mathrm{s},T'}|(-0.16)$, white bullets to the smallest one (0) and gray to intermediate values with the larger the magnitude of $|\hat{\pi}^*_{\mathrm{s},T'}|$, the darker the gray. As one may observe, the policy considerably changes with T'. To assess the influence of T' on the ability of the policy $\hat{\pi}_{\mathrm{s},T'}$ to approximate an optimal policy over an infinite time horizon, we have computed for different values of T', the value of $\lim_{T\to\infty} C_T^{\hat{\pi}_{\mathrm{s},T'}}((\delta,\omega) = (0,8))$. There is no particular rationale for having chosen the particular initial state (0, 8) for evaluating the policy. Some side simulations have also shown that similar findings are observed by using other initial states. The results are reported on the first

TABLE I
CUMULATIVE COSTS OBSERVED FOR DIFFERENT POLICIES $\hat{\pi}^*_{\mathrm{s},T'}$

	$T'=1$	$T'=5$	$T'=20$	$T'=100$
RL, $\lvert\mathcal{F}\rvert = 10^5$	$44.3^\dagger$	39.3	20.5	20.9
RL, $\lvert\mathcal{F}\rvert = 10^3$	$43.9^\dagger$	47.2	27.7	26.9
MPC	$55.4^\dagger$	38.3	20.3	20.3

line of Table I. The suffix † in the table indicates that the policy drives the system to the *terminal state* $x_\perp$ before $t = 1000$.

It can be seen that the cost tends to decrease when T' increases. This means that the suboptimality of the policy $\hat{\pi}_{\mathrm{s},T'}$ tends to decrease with T', as predicted by Theorem 3.

Performances of the fitted Q iteration algorithm are influenced by the information it has on the optimal control problem, represented by the set $\mathcal{F}$. Usually, the less information, the larger $\lim_{T\to\infty} C_T^{\hat{\pi}^*_{\mathrm{s},T'}}(x) - \lim_{T\to\infty} C_T^{\pi^*_{\mathrm{s},T'}}(x)$, assuming that T' is sufficiently large. To illustrate this, the RL algorithm has been run by considering this time a 1000-element set of four-tuples, with the four-tuples generated in the same conditions as before. The resulting policies $\hat{\pi}^*_{\mathrm{s},T'}$ are shown in Fig. 2(e)–(h). As shown in the second line of Table I, these policies indeed lead to higher costs than those obtained with 100 000 four-tuples.

It was mentioned before, when defining the optimal control problem, that policies leading to small costs were also leading to good damping of P_{e}. This is shown by putting Table I in perspective with Fig. 3(a)–(h) which draw the evolution of the electrical power when the system is controlled by the policies $\hat{\pi}^*_{\mathrm{s},T'}$ computed up to now. Observe also that the 1000-element set of four-tuples leads when $T' \neq 1$ to larger residual oscillations than the 100 000-element set.

2) MPC Results: Fig. 2(i)–(l) [respectively (m)–(p)] shows the different policies $\hat{\pi}_{\mathrm{s},T'}$ computed for increasing values of T' when using MPC(PD) [respectively MPC(PC)]. To compute the value of $\hat{\pi}_{\mathrm{s},T'}$ for a specific state x, the optimization algorithm needs to be run. When T' is smaller or equal to five, the interior point algorithms are always converging. However, when T' is larger than five, some convergence problems arise. Indeed, on the 1304 states x for which the policy $\hat{\pi}_{\mathrm{s},T'}$ was estimated by means of MPC(PD) [respectively MPC(PC)], 154 (respectively 49) failed to converge for $T' = 20$ and 111 (respectively 26) for $T' = 100$. States for which the MPC algorithm failed to converge are not represented on the figures.

These convergence issues are analyzed in Section IV-E. First, we compare the results of MPC and RL methods.

3) MPC Versus RL Policies: It is interesting to notice that, except for the states for which MPC did not converge, MPC policies look quite similar to RL policies computed with a large enough number of samples. This is not surprising, since both methods indeed aim to approximate a $\pi^*_{\mathrm{s},T'}$ policy.

From the "infinite" horizon costs reported in Table I for MPC and RL policies determined for various values of T', we observe that for $T' = 5, T' = 20$ and $T' = 100$, those computed by MPC (slightly) outperform those computed by RL. Notice that the results reported are denoted by MPC, since those of MPC(PC) and MPC(PD) are exactly the same in this context.

Consider now the time evolution of P_{e} when the system, starting from $(\delta, \omega) = (0, 8)$ is controlled by RL and MPC policies. These $P_{\mathrm{e}}-t$ curves are shown in Fig. 3 (where again the curves related to the MPC(PD) and MPC(PC) policies were identical). While the $\hat{\pi}_{\mathrm{s},20}$ and $\hat{\pi}_{\mathrm{s},100}$ policies computed by the RL algorithm produce (small) residual oscillations, the policies $\hat{\pi}_{\mathrm{s},20}$ and $\hat{\pi}_{\mathrm{s},100}$ computed by MPC are able to damp them completely. This is mainly explained by the ability the MPC algorithms had to exploit the continuous nature of the action space while the RL algorithm discretized it into a finite number of values.

E. Convergence Problems of MPC

We first report in Table II the number of cases (among the total of 1304) for which the MPC computation using PD or PC algorithms diverges, and that for two different initialization strategies. Either (δ_t, ω_t) is initialized to (δ_0, ω_0) for all $t = 1, \ldots, T'-1$ or δ_t and ω_t are initialized to the middle of their respective variation interval, i.e., $\delta_t = (2.73 + (-0.97)/2) = 0.88$, $\omega = (11.92 + (-11.92)/2) = 0$. For both initialization schemes, u_t is initialized to the middle of its variation interval $(u_t = -0.08)$.

One can observe that the PC algorithm clearly outperforms the PD one, due to its ability to better exploit the nonlinearities of the problem and to control in a more effective way the decrease of the barrier parameter. When using the best initialization strategy, the MPC(PC) approach has a rate of convergence of 96.2% for $T' = 20$ and 98.0% for $T' = 100$ which can be deemed satisfactory. Furthermore, in all our simulations, very strict convergence criteria have been used[4] for PD and PC algorithms. In this respect, we have observed that a good number of cases declared as divergent met most of these criteria offering thus a practical feasible solution, possibly close to the optimal one.

Even though many of these convergence problems for the MPC(PD) and MPC(PC) approaches could thus be mitigated along the lines suggested above, we have analyzed the possible reasons behind these problems.

1) *Infeasibility of the optimization problem for some values of x_0 and T'.* If the equality constraints $x_{t+1} = f(x_t, u_t)$ were represented exactly, this could not be the case since for $u_t = 0, t = 0, 1, \ldots, T'-1$, the trajectory stays inside the stability domain and, hence, satisfies all constraints. However, approximating the discrete dynamics by (11) and (12) could possibly lead to the impossibility of keeping the system state inside the stability domain for some initial points. To assess whether these convergence problems were indeed mostly caused by approximating the equality constraints, the system modeled by (11) and (12) has been simulated over 100 time steps and with $u_t = 0$, for all t. We found out that for the 1304 states for which the policy was estimated, 20 were indeed leading

[4]We declare that a (locally) optimal solution of MPC problem is found (and the optimization process terminates) when: primal feasibility, scaled dual feasibility, scaled complementarity gap, and objective function variation from an iteration to the next fall below some standard tolerances [39].

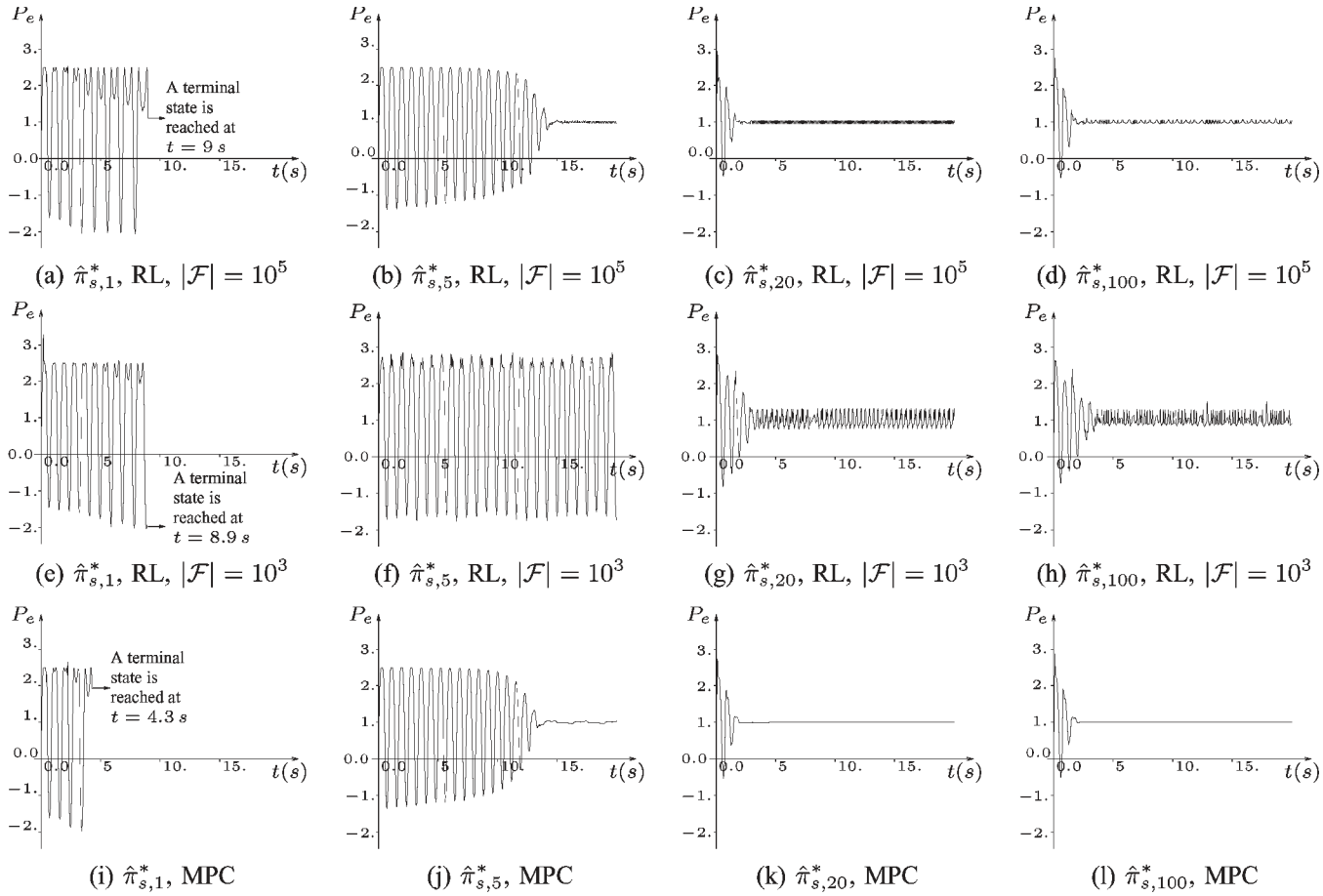

Fig. 3. Representation of $P_e(t)$ when the system starts from $(\delta, \omega) = (0, 8)$ and is controlled by the policy $\hat{\pi}^*_{s,T'}(x)$. (a) $\hat{\pi}^*_{s,1}$, RL, $|\mathcal{F}| = 10^5$. (b) $\hat{\pi}^*_{s,5}$, RL, $|\mathcal{F}| = 10^5$. (c) $\hat{\pi}^*_{s,20}$, RL, $|\mathcal{F}| = 10^5$. (d) $\hat{\pi}^*_{s,100}$, RL, $|\mathcal{F}| = 10^5$. (e) $\hat{\pi}^*_{s,1}$, RL, $|\mathcal{F}| = 10^3$. (f) $\hat{\pi}^*_{s,5}$, RL, $|\mathcal{F}| = 10^3$. (g) $\hat{\pi}^*_{s,20}$, RL, $|\mathcal{F}| = 10^3$. (h) $\hat{\pi}^*_{s,100}$, RL, $|\mathcal{F}| = 10^3$. (i) $\hat{\pi}^*_{s,1}$, MPC. (j) $\hat{\pi}^*_{s,5}$, MPC. (k) $\hat{\pi}^*_{s,20}$, MPC. (l) $\hat{\pi}^*_{s,100}$, MPC.

TABLE II
STUDY OF THE CONVERGENCE PROBLEMS FACED BY THE MPC ALGORITHMS

	MPC (PD)		MPC (PC)	
	$T'=20$	$T'=100$	$T'=20$	$T'=100$
$(0.88, 0)$	154	111	49	26
(δ_0, ω_0)	202	348	136	366

to a violation of the constraints when chosen as initial state. This may provide, for example, an explanation for a significant percentage of the 26 divergent cases observed with the MPC(PC) approach when $T' = 100$. However, these 20 states form only a small portion of the 111 state set for which the MPC(PD) approach failed to converge.

2) *Choice of initial conditions*. The results shown in Figs. 2 and 3 were obtained by initializing every δ_t, ω_t, u_t intervening in the optimization problem to the middle of their respective variation interval. Table II reports also the number of times convergence problems have been met by initializing δ_t and ω_t to δ_0 and ω_0, respectively. As one can observe, such a choice leads to even more convergence problems. Other initialization strategies were also tried, such as initializing δ_t, ω_t so that equality constraints (11) and (12) are satisfied, but they yielded even poorer convergence.

3) *Parameters tuning.* Parameters intervening in the PD and PC algorithms, and particularly the initial value of the barrier parameter, indeed strongly influence the convergence properties and often need to be tuned to the optimization problem tackled. The results reported in this paper have been obtained by tuning finely these parameters and declaring divergence only when the algorithm failed to converge for several initial values of the barrier parameter.

F. Other MPC Solvers

Of course, many other nonlinear optimization techniques have been proposed in the literature to handle nonlinear MPC and some of these could also potentially mitigate/eliminate the convergence problems (e.g., sequential quadratic (SQP) approaches using PD IPM to solve the subproblems [40], feasibility-perturbed SQP programming [41], active set quadratic programming [42], reduced space interior point strategy [43]).

G. Computational Aspects

This section discusses some CPU considerations behind the MPC and the RL approaches. The discussion is based on the

TABLE III
CPU TIMES (IN SECONDS ON AN AMD 2800+/1.6-GHz PROCESSOR) FOR THE MPC AND THE RL APPROACHES

		$T'=1$	$T'=5$	$T'=20$	$T'=100$
MPC(PD) approach. Average CPU time to compute an action $\hat{\pi}^*_{s,T'}(x_0)$. The average is computed by considering states $x_0 \in X_{test}$ for which the PD algorithm converges for $T' \in \{1, 5, 20, 100\}$.		0.0087	0.0216	0.0997	0.7812
RL approach. CPU time to compute $\hat{Q}_1$, $\hat{Q}_2$, …, $\hat{Q}_{T'}$	$\lvert\mathcal{F}\rvert = 10,000$	2.66	42.62	182.6	914.3
	$\lvert\mathcal{F}\rvert = 20,000$	5.56	92.7	389.78	1961.23
	$\lvert\mathcal{F}\rvert = 100,000$	34.32	569.24	2452.48	3476.39
RL approach. Average CPU time to extract from $\hat{Q}_{T'}$ an action $\hat{\pi}^*_{s,T'}(x_0)$. The average is computed by considering states $x_0 \in X_{test}$.	$\lvert\mathcal{F}\rvert = 10,000$	0.00053	0.00053	0.00052	0.00054
	$\lvert\mathcal{F}\rvert = 20,000$	0.00063	0.00061	0.00058	0.00061
	$\lvert\mathcal{F}\rvert = 100,000$	0.00082	0.00082	0.00078	0.00076

results reported in Table III. The CPU times related to the MPC(PC) algorithm are not given in this table since they are essentially of the same order of magnitude of those corresponding to the MPC(PD) algorithm.

For MPC(PD), CPU times given in Table III are averages over all $x_0 \in X_{\text{test}}$ (except those for which the algorithm did not converge) needed to solve the nonlinear optimization problem to determine the action $\hat{\pi}_{\text{s},T'}(x_0)$. These CPU times refer to an implementation of the PD algorithm exploiting the sparse structure of the Hessian matrices, which considerably reduces the computational effort. Nevertheless, most of the CPU time is consumed by the factorizations of the Hessian matrices needed by the PD algorithm. We observe that the CPU times for the MPC approach grow faster than linearly with respect to T'. This growth is mainly due to two factors: a superlinear increase of the time needed to factorize the Hessian with T' and an increase in the number of iterations before convergence of the PD algorithm. Indeed, the average number of iterations is 10.5 for $T'=1$, 11.4 for $T'=5$, 19.5 for $T'=20$, and 25.9, for $T'=100$.

For RL, we give CPU times both for the offline computation of the sequence of functions $\hat{Q}_N(N=1,\ldots,T')$ from the set of samples $\mathcal{F}$ (Table III, second line), and for the average (with respect to the full set of states $x_0 \in X_{\text{test}}$) online CPU time necessary to extract from $\hat{Q}_{T'}$ the action $\hat{\pi}^*_{\text{s},T'}(x_0)$ (Table III, third line). We observe that offline CPU times increase linearly with T', which is due to the iterative nature of the fitted Q iteration algorithm, essentially repeating at each iteration the same kind of computations.[5] On the other hand, we see from Table III, that the online CPU time needed to extract from $\hat{Q}_{T'}$, the value of $\hat{\pi}_{\text{s},T'}(x_0)$ stays essentially constant with T'. Indeed, the CPU time needed to extract $\hat{\pi}_{\text{s},T'}(x_0)$ from $\hat{Q}_{T'}$ is mostly spent on the determination of $\hat{Q}(x_0,u)$, for all $u \in U'$, done by propagating the different inputs (x_0,u) through the ensemble of regression trees. The depth of these trees grows with the size of the set of four-tuples $\mathcal{F}$, and concerning the influence of the size of the set of four-tuples $\mathcal{F}$ on the CPU times for the RL approach, we observe from Table III that there is a slightly superlinear increase with respect to $|\mathcal{F}|$ of the offline CPU time needed to compute $\hat{Q}_1,\ldots,\hat{Q}_{T'}$ and a strongly sublinear increase with $|\mathcal{F}|$ of the online CPU time required to extract from $\hat{Q}_{T'}$ the action $\pi^*_{\text{s},T'}(x_0)$. These empirical observations comply nicely with a theoretical complexity analysis, which shows that in the case of balanced trees these CPU times should increase on the order of $|\mathcal{F}|\log|\mathcal{F}|$ and $\log|\mathcal{F}|$, respectively.

[5]Note that the ratio between the CPU time used to compute $\hat{Q}_1$ and that used to compute $\hat{Q}_1,\ldots,\hat{Q}_5$ is smaller than 1/5; this is explained by the fact that, at the first iteration, the computation of $\arg\inf_{u\in U}\hat{Q}_{N-1}(x^l_{t+1},u)$ when building the first training set [(5)] is trivial, since $\hat{Q}_0 \equiv 0$.

V. ABOUT COMPARING MPC AND RL FOR LINEAR, UNCERTAIN AND/OR STOCHASTIC SYSTEMS

Comparison between the RL and the MPC approaches was carried out in this paper on a nonlinear deterministic system whose dynamics was known with no uncertainties. In this section, we discuss to what extend the qualitative nature of the results would still hold if the system were to be linear, uncertain and/or stochastic.

1) Linear: If the system dynamics were to be linear and the cost function convex, the global optimization problem solved by MPC methods would be convex. The huge arsenal of convex programming optimization techniques could therefore be used with some theoretical guarantees of convergence to an optimum. In such a linear world, MPC techniques would certainly perform better. The RL algorithm used in this paper would, however, not exploit the linearity of the system and, in principle, there is no reason for it to behave better in a linear world. However, researchers in RL have focused on the inference of control strategies from trajectories for linear systems (see, e.g., [44]), and the approaches they have proposed should certainly be used to fairly compare MPC and RL when dealing with linear systems.

2) Uncertain: The RL methods can in principle work directly from real-life system trajectories and, as long as those trajectories are available, such ability could offer them a way to circumvent the problems related to uncertainties on the model. However, sometimes, even if such trajectories are available, the information they contain is insufficient for the RL techniques to infer some good policies. Generating trajectories from a model, even an uncertain one, could therefore help these techniques to behave better. In such a framework, RL techniques would therefore be sensitive to uncertainties, as MPC techniques are. Moreover, to state that MPC techniques are intrinsically suboptimal when the model is plagued with uncertainties is certainly limitative. They could indeed be coupled with some online system identification techniques that could monitor the trajectories to fit the model. Moreover, the time-receding horizon strategy adopted in MPC confers these methods (as well as the RL approach) some robustness properties with respect to

uncertainties (and also noise) which have been vastly studied in the control literature [45]–[47].

3) Stochastic: RL techniques have been initially designed for solving stochastic optimal control problems and focus on the learning of closed-loop policies which are optimal for such problems. The open-loop nature of the policy computed at each time step by MPC approaches makes them intrinsically suboptimal when facing problems with noise. They bear therefore a handicap that RL methods do not have. However, this statement needs to be moderated. First, the time-receding horizon strategy they adopt makes them to some extend recover properties that some closed-loop solutions can have. Moreover, there is now a vast body of work in the MPC literature for extending these techniques to stochastic problems. Among them, we cite those relying on some min–max approaches aimed at finding a solution which is optimal with respect to the worst case "disturbance sequence" occurring [46] and those known as chance (or probabilistically) constrained MPC [48], [49]. There is also significant body of work in the field of multistage stochastic programming (MSSP) (see, e.g., [50] and [51]), which is very close in essence to MPC since the computation of the actions in MSSP is also done by solving a global optimization problem, and that offers possibilities to extend in a close-to-optimal way the standard MPC approach to stochastic systems. However, the MSSP framework leads to optimization problems which computational tractability is much lower than those of deterministic MPC approaches.

VI. Conclusion

We have presented in this paper MPC and RL as alternative approaches to deterministic optimal control.

Our experiments, on a simple but nontrivial and practically relevant optimal control problem from the electric power systems field, for which a good model was given, have shown that the fitted Q iteration algorithm was providing essentially policies equivalent to those computed by an IPM-based MPC approach. The experiments have also highlighted the fact that MPC is essentially (slightly) less robust than fitted Q iteration-based RL from the numerical point of view, but has a slight advantage in terms of accuracy.

For control problems where a good enough model is available in appropriate form, we, thus, suggest to use the two approaches in combination. The fitted Q iteration could be used offline together with samples generated by Monte Carlo simulations, as an effective ADP method. In online mode, MPC could exploit the policies precompiled in this way by RL, together with the system model, in order to start optimization from a better initial guess of the optimal trajectory. This combination of "offline global" and "online local" approaches could allow to circumvent limitations of MPC such as convergence problems or the risk of being trapped in a local optimum. It should be noted that several other authors have already considered ADP and MPC as two complementary methods. For example, in [52], Negenborn *et al.* propose to use ADP in order to reduce the computationally intensive MPC optimizations over a long control horizon to an optimization over a single step by using the value function computed by ADP. In [53], Lee and Lee argue that ADP may mitigate two important drawbacks of the conventional MPC formulation, namely, the potentially exorbitant online computational requirement and the inability to consider the uncertainties in the optimal control calculation. Moreover, in this ADP context, it would be interesting to investigate whether other algorithms which also reformulate the search for an optimal policy as a sequence of batch-mode SL problems, could be used as good optimizers for MPC problems [54]–[56].

On the other hand, for control problems where the information comes mainly from observations of system trajectories, one would need to use general system identification techniques before applying MPC, while the fitted Q iteration can be applied directly to this information without any special hypothesis on system behavior or cost function. Thus, in this context, the arbitration among these two approaches will depend on the quality of the prior knowledge about system dynamics and cost functions that could be exploited in the context of system identification. Thus, even more in this context, the very good results obtained with the "blind" fitted Q iteration method make it an excellent candidate to solve such problems, even without exploiting any prior knowledge.

Between these two extremes, there is a whole continuum corresponding to more or less knowledge available about the problem to solve, like, for example, knowledge of the system dynamics up to a small number of unknown parameters, in particular assumption of linear behavior, or having an analytical description of the cost function only, or of the system dynamics only. Admittedly, many interesting applications fall in this category, and we suggest that the proper way to address these problems is to combine model-based techniques such as MPC and learning-based techniques such as RL.

Appendix A

Before proving Theorem 3, we first prove the following theorem.

Theorem 4: A policy $\pi^*_{\mathrm{s},T'}$ leads to a cost over an infinite number of stages which is not larger than the cost over an infinite number of stages associated with an optimal policy plus a term $2(\gamma^{T'} B_{\mathrm{c}}/(1-\gamma)^2)$, or equivalently

$$\sup_{x\in X}\left(\lim_{T\to\infty} C_T^{\pi^*_{\mathrm{s},T'}}(x) - \lim_{T\to\infty} C_T^{\pi^*_T}(x)\right) \leq 2\frac{\gamma^{T'} B_{\mathrm{c}}}{(1-\gamma)^2}. \quad (16)$$

Proof: Before starting the proof, we introduce some notations and recall some results. Let H be the mapping that transforms $K : X \to \mathbb{R}$ into

$$(HK)(x) = \inf_{u\in U}\left(c(x,u) + \gamma K\left(f(x,u)\right)\right). \quad (17)$$

The operator H is usually referred to as the Bellman operator. Let us define the sequence of functions $J_N : X \to \mathbb{R}$ by the recursive equation $J_N = HJ_{N-1}$ with $J_0 \equiv 0$. Since H is a contraction mapping, the sequence of functions J_N converges, in infinite norm, to the function J, unique solution of the equation $J = HJ$. Observe that $J_N(x) = \inf_{u\in U} Q_N(x,u)$ for all $x \in X$.

Let π_s be a stationary policy and H^{π_s} the mapping that transforms $K : X \times U \rightarrow \mathbb{R}$ into

$$(H^{\pi_s}K)(x) = \left(c\left(x, \pi_s(x)\right) + \gamma K\left(f\left(x, \pi_s(x)\right)\right)\right). \quad (18)$$

We define the sequence of functions $J_N^{\pi_s} : X \rightarrow \mathbb{R}$ by the recursive equation $J_N^{\pi_s} = H^{\pi_s} J_{N-1}^{\pi_s}$ with $J_0^{\pi_s} \equiv 0$. Since H^{π_s} is a contraction mapping, the sequence of functions $J_{N-1}^{\pi_s}$ converges to J^{π_s}, unique solution of $J^{\pi_s} = H^{\pi_s} J^{\pi_s}$.

We recall two results from the DP theory that can be found, for example, in [29]

$$C_T^{\pi_T^*}(x) = J_T(x) \qquad C_T^{\pi_s}(x) = J_T^{\pi_s}(x) \quad \forall x \in X \quad (19)$$

with the equalities still begin valid at the limit

$$\lim_{T\rightarrow\infty} C_T^{\pi_T^*}(x) = J(x) \qquad \lim_{T\rightarrow\infty} C_T^{\pi_s}(x) = J^{\pi_s}(x). \quad (20)$$

We now start the proof. We can write

$$J^{\pi^*_{s,T'}}(x) - J(x) \leq J^{\pi^*_{s,T'}}(x) - J(x) + H^{\pi^*_s} J_{T'-1}(x) - H^{\pi_{s,T'}} J_{T'-1}(x)$$

since $\pi^*_{s,T'}(x) \in \arg\inf_{u\in U}(c(x,u) + \gamma J_{T'-1}(f(x,u))$. Thus, in norm

$$\begin{aligned}\left\|J^{\pi^*_{s,T'}} - J\right\|_\infty &\leq \left\|J^{\pi^*_{s,T'}} - H^{\pi^*_{s,T'}} J_{T'-1}\right\|_\infty + \left\|J - H^{\pi^*_s} J_{T'-1}\right\|_\infty \\ &\leq \gamma \left\|J^{\pi^*_{s,T'}} - J_{T'-1}\right\|_\infty + \gamma \|J - J_{T'-1}\|_\infty \\ &\leq \gamma \left\|J^{\pi^*_{s,T'}} - J\right\|_\infty + 2\gamma \|J - J_{T'-1}\|_\infty \\ &\leq \frac{2\gamma}{1-\gamma} \|J - J_{T'-1}\|_\infty.\end{aligned}$$

Since

$$\begin{aligned}\|J - J_{T'-1}\|_\infty &= \left\|H^{\pi^*_s} J - H^{\pi^*_s} J_{T'-2}\right\|_\infty \\ &\leq \gamma \|J - J_{T'-2}\|_\infty \leq \gamma^{T'-1} \|J\|_\infty \leq \frac{\gamma^{T'-1}}{1-\gamma} B_{\mathrm{c}}\end{aligned}$$

we have

$$\left\|J^{\pi^*_{s,T'}} - J\right\|_\infty \leq \frac{2\gamma^{T'}}{(1-\gamma)^2} B_{\mathrm{c}}. \quad (21)$$

From (20), we can write

$$\sup_{x\in X}\left(\lim_{T\rightarrow\infty} C_T^{\pi^*_{s,T'}}(x) - \lim_{T\rightarrow\infty} C_T^{\pi_T^*}(x)\right) = \left\|J^{\pi^*_{s,T'}} - J\right\|_\infty.$$

By using this latter expression with inequality (21), we prove the theorem. ■

Proof of Theorem 3: We have

$$\begin{aligned}C_T^{\pi^*_{s,T'}}(x) - C_T^{\pi_T^*}(x) &\leq \lim_{T\rightarrow\infty} C_T^{\pi^*_{s,T'}}(x) + \frac{\gamma^T B_{\mathrm{c}}}{1-\gamma} - C_T^{\pi_T^*}(x) \leq \lim_{T\rightarrow\infty} C_T^{\pi^*_{s,T'}}(x) \\ &\quad + \frac{\gamma^T B_{\mathrm{c}}}{1-\gamma} - \left(\lim_{T\rightarrow\infty} C_T^{\pi_T^*}(x) - \frac{\gamma^T B_{\mathrm{c}}}{1-\gamma}\right) \\ &= \lim_{T\rightarrow\infty} C_T^{\pi^*_{s,T'}}(x) - \lim_{T\rightarrow\infty} C_T^{\pi_T^*}(x) + 2\frac{\gamma^T B_{\mathrm{c}}}{1-\gamma}\end{aligned}$$

By using this inequality with Theorem 4, we can write

$$C_T^{\pi^*_{s,T'}}(x) - C_T^{\pi_T^*}(x) \leq 2\frac{\gamma^{T'} B_{\mathrm{c}}}{(1-\gamma)^2} + 2\frac{\gamma^T B_{\mathrm{c}}}{1-\gamma} \leq \frac{\gamma^{T'}(4-2\gamma) B_{\mathrm{c}}}{(1-\gamma)^2}.$$

■

ACKNOWLEDGMENT

The authors would like to thank the contribution of the "Interuniversity Attraction Poles" Programme VI/04 - DYSCO of the Belgian Science Policy. M. Glavic would like to thank the FNRS for supporting his research stays at the University of Liège.

REFERENCES

[1] M. Morari and J. H. Lee, "Model predictive control: Past, present and future," *Comput. Chem. Eng.*, vol. 23, no. 4, pp. 667–682, May 1999.
[2] J. Maciejowski, *Predictive Control With Constraints*. Englewood Cliffs, NJ: Prentice-Hall, 2001.
[3] D. Mayne and J. Rawlings, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 2000.
[4] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.
[5] R. Sutton and A. Barto, *Reinforcement Learning, An Introduction*. Cambridge, MA: MIT Press, 1998.
[6] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.
[7] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Cambridge Univ., Cambridge, U.K., 1989.
[8] R. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3/4, pp. 229–256, May 1992.
[9] J. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Mach. Learn.*, vol. 16, no. 3, pp. 185–202, Sep. 1994.
[10] R. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, vol. 12. Cambridge, MA: MIT Press, 2000, pp. 1057–1063.
[11] G. Tesauro, "TD-Gammon, a self-teaching backgammon program, achieves master-level play," *Neural Comput.*, vol. 6, no. 2, pp. 215–219, Mar. 1994.
[12] S. Singh and D. Bertsekas, "Reinforcement learning for dynamic channel allocation in cellular telephone systems," in *Advances in Neural Information Processing Systems*, vol. 9, M. Mozer, M. Jordan, and T. Petsche, Eds. Cambridge, MA: MIT Press, 1997, pp. 974–980.
[13] J. Bagnell and J. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," in *Proc. Int. Conf. Robot. Autom.*, 2001, pp. 1615–1620.
[14] D. Ernst, M. Glavic, and L. Wehenkel, "Power systems stability control: Reinforcement learning framework," *IEEE Trans. Power Syst.*, vol. 19, no. 1, pp. 427–435, Feb. 2004.
[15] S. Qin and T. Badgwell, "An overview of industrial model predictive control technology," in *Proc. Chem. Process Control*, 1997, vol. 93, pp. 232–256. no. 316.
[16] M. Hassoun, *Fundamentals of Artificial Neural Networks*. Cambridge, MA: MIT Press, 1995.

[17] B. Schölkopf, C. Burges, and A. Smola, *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA: MIT Press, 1999.
[18] C. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge, MA: MIT Press, 2000.
[19] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[20] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006.
[21] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.
[22] D. Ormoneit and S. Sen, "Kernel-based reinforcement learning," *Mach. Learn.*, vol. 49, no. 2/3, pp. 161–178, Nov. 2002.
[23] M. Riedmiller, "Neural fitted Q iteration—First experiences with a data efficient neural reinforcement learning method," in *Proc. 16th Eur. Conf. Mach. Learn.*, 2005, pp. 317–328.
[24] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *J. Mach. Learn. Res.*, vol. 6, pp. 503–556, Apr. 2005.
[25] D. Ernst, P. Geurts, and L. Wehenkel, "Iteratively extending time horizon reinforcement learning," in *Proc. 14th Eur. Conf. Mach. Learn.*, N. Lavra, L. Gamberger, and L. Todorovski, Eds., Sep. 2003, pp. 96–107.
[26] M. Riedmiller, "Neural reinforcement learning to swing-up and balance a real pole," in *Proc. Int. Conf. Syst., Man, Cybern.*, Big Island, HI, 2005, vol. 4, pp. 3191–3196.
[27] D. Ernst, G. Stan, J. Gonçalvez, and L. Wehenkel, "Clinical data based optimal STI strategies for HIV: A reinforcement learning approach," in *Proc. BENELEARN*, 2006, pp. 65–72.
[28] D. Ernst, R. Marée, and L. Wehenkel, "Reinforcement learning with raw pixels as state input," in *Proc. Int. Workshop Intell. Comput. Pattern Anal./Synthesis*, Aug. 2006, vol. 4153, pp. 446–454.
[29] D. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed., vol. I, Belmont, MA: Athena Scientific, 2000.
[30] O. Hernandez-Lerma and J. Lasserre, *Discrete-Time Markov Control Processes. Basic Optimality Criteria*. New York: Springer-Verlag, 1996.
[31] D. Bertsekas, "Dynamic programming and suboptimal control: From ADP to MPC," in *Proc. 44th IEEE Conf. Decision Control, Eur. Control Conf.*, 2005, p. 10.
[32] M. Ghandhari, "Control Lyapunov functions: A control strategy for damping of power oscillations in large power systems," Ph.D. dissertation, Roy. Inst. Technol., Stockholm, Sweden, 2000. [Online]. Available: http://www.lib.kth.se/Fulltext/ghandhari001124.pdf.
[33] D. Ernst, M. Glavic, P. Geurts, and L. Wehenkel, "Approximate value iteration in the reinforcement learning context. Application to electrical power system control," *Int. J. Emerging Elect. Power Syst.*, vol. 3, no. 1, p. 37, 2005.
[34] G. Rogers, *Power System Oscillations*. Norwell, MA: Kluwer, 2000.
[35] P. Kundur, *Power System Stability and Control*. New York: McGraw-Hill, 1994.
[36] M. A. Pai, *Energy Function Analysis for Power System Stability*, ser. Power Electronics and Power Systems. Norwell, MA: Kluwer, 1989.
[37] M. Pavella and P. Murthy, *Transient Stability of Power Systems: Theory and Practice*. Hoboken, NJ: Wiley, 1994.
[38] A. Fiacco and G. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Hoboken, NJ: Wiley, 1968.
[39] S. Mehrotra, "On the implementation of a primal-dual interior point method," *SIAM J. Optim.*, vol. 2, no. 4, pp. 575–601, Nov. 1992.
[40] J. Albuquerque, V. Gopal, G. Staus, L. Biegler, and E. Ydstie, "Interior point SQP strategies for large-scale, structured process optimization problems," *Comput. Chem. Eng.*, vol. 23, no. 4, pp. 543–554, May 1999.
[41] M. Tenny, S. Wright, and J. Rawlings, "Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming," *Comput. Optim. Appl.*, vol. 28, no. 1, pp. 87–121, Apr. 2004.
[42] R. Bartlett, A. Wachter, and L. Biegler, "Active sets vs. interior point strategies for model predictive control," in *Proc. Amer. Control Conf.*, Chicago, IL, Jun. 2000, pp. 4229–4233.
[43] A. Cervantes, A. Wachter, R. Tutuncu, and L. Biegler, "A reduced space interior point strategy for optimization of differential algebraic systems," *Comput. Chem. Eng.*, vol. 24, no. 1, pp. 39–51, Apr. 2000.
[44] S. Bradtke, "Reinforcement learning applied to linear quadratic regulation," in *Advances in Neural Information Processing Systems*, vol. 5. San Mateo, CA: Morgan Kaufmann, 1993, pp. 295–302.
[45] E. Zafiriou, "Robust model predictive control of processes with hard constraints," *Comput. Chem. Eng.*, vol. 14, no. 4/5, pp. 359–371, May 1990.
[46] M. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, Oct. 1996.
[47] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in Identification and Control*, vol. 245, A. Garruli, A. Tesi, and A. Viccino, Eds. New York: Springer-Verlag, 1999, pp. 207–226.
[48] P. Li, M. Wendt, and G. Wozny, "Robust model predictive control under chance constraints," *Comput. Chem. Eng.*, vol. 24, no. 2, pp. 829–834, Jul. 2000.
[49] P. Li, M. Wendt, and G. Wozny, "A probabilistically constrained model predictive controller," *Automatica*, vol. 38, no. 7, pp. 1171–1176, Jul. 2002.
[50] W. Romisch, "Stability of stochastic programming problems," in *Stochastic Programming. Handbooks in Operations Research and Management Science*, vol. 10, A. Ruszczynski and A. Shapiro, Eds. Amsterdam, The Netherlands: Elsevier, 2003, pp. 483–554.
[51] H. Heitsch, W. Romisch, and C. Strugarek, "Stability of multistage stochastic programs," *SIAM J. Optim.*, vol. 17, no. 2, pp. 511–525, Aug. 2006.
[52] R. Negenborn, B. De Schutter, M. Wiering, and J. Hellendoorn, "Learning-based model predictive control for Markov decision processes," in *Proc. 16th IFAC World Congr.*, Jul. 2005, p. 6.
[53] J. M. Lee and J. H. Lee, "Simulation-based learning of cost-to-go for control of nonlinear processes," *Korean J. Chem. Eng.*, vol. 21, no. 2, pp. 338–344, Mar. 2004.
[54] G. Gordon, "Stable function approximation in dynamic programming," in *Proc. 12th Int. Conf. Mach. Learn.*, 1995, pp. 261–268.
[55] M. Lagoudakis and R. Parr, "Reinforcement learning as classification: Leveraging modern classifiers," in *Proc. ICML*, 2003, pp. 424–431.
[56] A. Fern, S. Yoon, and R. Givan, "Approximate policy iteration with a policy language bias," in *Advances in Neural Information Processing Systems*, vol. 16, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004, p. 8.

Damien Ernst (M'98) received the M.Sc. and Ph.D. degrees from the University of Liège, Liège, Belgium, in 1998 and 2003, respectively.

He spent the period 2003–2006 with the University of Liège as a Postdoctoral Researcher of the Belgian National Fund for Scientific Research (FNRS), Brussels, Belgium, and held during this period positions as Visiting Researcher with Carnegie Mellon University, Pittsburgh, PA; the Massachusetts Institute of Technology, Cambridge, MA; and the Swiss Federal Institute of Technology Zurich, Zurich, Switzerland. He spent the academic year 2006–2007 working with the Ecole Supérieure d'Electricité, Paris, France, as a Professor. He is currently a Research Associate with the FNRS. He is also affiliated with the Systems and Modelling Research Unit, University of Liège. His main research interests are in the fields of power system dynamics, optimal control, reinforcement learning, and design of dynamic treatment regimes.

Mevludin Glavic (M'04–SM'07) received the M.Sc. degree from the University of Belgrade, Belgrade, Serbia, in 1991 and the Ph.D. degree from the University of Tuzla, Tuzla, Bosnia, in 1997.

He spent the academic year 1999/2000 with the University of Wisconsin—Madison, as a Fulbright Postdoctoral Scholar. From 2001 to 2004, he was a Senior Research Fellow with the University of Liege, where he is currently an Invited Researcher. His research interests include power system control and optimization.

Florin Capitanescu received the Electrical Power Engineering degree from the University "Politehnica" of Bucharest, Bucharest, Romania, in 1997 and the DEA and Ph.D. degrees from the University of Liège, Liège, Belgium, in 2000 and 2003, respectively.

He is currently with the University of Liège. His main research interests include optimization methods and voltage stability.

Louis Wehenkel (M'93) received the Electrical Engineer (electronics) and Ph.D. degrees from the University of Liège, Liège, Belgium, in 1986 and 1990, respectively.

He is currently a Full Professor of electrical engineering and computer science with the University of Liège. His research interests lie in the fields of stochastic methods for systems and modeling, machine learning and data mining, with applications in power systems planning, operation, and control, and bioinformatics.