FACTS devices controlled by means of reinforcement learning algorithms

D. Ernst[†] and L. Wehenkel
Department of Electrical and Computer Engineering - Institut Montefiore
University of Liège - Sart-Tilman B28 - B4000 Liège - Belgium
[†] Research Fellow F.N.R.S.
{ernst,lwh}@montefiore.ulg.ac.be

March 14, 2002

Abstract - Reinforcement learning consists of a collection of methods for approximating solutions to deterministic and stochastic optimal control problems of unknown dynamics. These methods learn by experience how to adjust a closed-loop control rule which is a mapping from the system states to control actions. This paper proposes an application of reinforcement learning methods to the control of a FACTS device aimed to damp power system oscillations. A detailed case study is carried out on a synthetic four-machine power system.

Keywords - FACTS, reinforcement learning, power system oscillations, discrete time optimal control, adaptive control

1 Introduction

Reinforcement learning (RL) takes its origin in optimal control theory and dynamic programming ([2]). It aims at approximating by experience solutions to problems of unknown dynamics. Year after year, the techniques evolved leading to a panel of more and more efficient algorithms. From a theoretical point of view, many breakthroughs have been realized notably concerning the convergence of the algorithms and their applications to nonlinear systems ([13], [10]). Also the steady increase in computer capacities makes RL methods more and more feasible. Therefore, the power system community started getting interested in such techniques, but surprisingly more in market modelling ([5]) rather than in nonlinear system control. However new needs appeared recently in power system dynamics control, especially with the introduction of new devices based on power electronics, like Flexible Alternating Current Transmission Systems (FACTS).

In this paper we focus on how to control by means of RL algorithms a FACTS device in order to damp power system oscillations, a phenomenon becoming even more important with the growth of extensive power systems and especially with the interconnection of these systems with ties of limited capacity.

Basically, the RL approach proposed in this paper to control the FACTS consists of an adaptive closed-loop control that tends to maximize a function, image of the quality of the oscillations damping. The only signal used by the controller is the electrical power transferred in the line measured at fixed intervals. This is roughly a discrete time optimal control approach in which the state dynamics and the observation equations are unknown.

The main advantages of such a controller are that it frees oneself from any knowledge of the power system dynamics equations, adapts itself to changing conditions and is able to act in a stochastic environment.

This paper is organised as follows. Section 2 sketches the main features of reinforcement learning algorithms applied to discrete time optimal control. The two main classes of algorithms, model learning and non-model learning algorithms are briefly explained. Section 3 describes the power system used to illustrate the RL based control algorithms and the characteristics of the FACTS device used. Section 4 explains the difficulties to overcome in order to apply successfully the rather general algorithms described in section 2 to the FACTS device control problem. It includes notably the design of a function image of the oscillations damping quality and a strategy to cope with the only availability of local measurements. Section 5 discusses the results obtained when acting in a constant load environment for different variants of RL algorithms. Section 6 studies the robustness of these adaptive control algorithms to large excursions of the load level of the system. Finally section 7 draws conclusions.

2 Reinforcement learning algorithms

Reinforcement learning will be presented here in the framework of discrete time [1] optimal control ([11]) of a deterministic non-linear system with constant sampling period and no terminal state [2].

If x_t represents the sampled state vector of the system at instant t, u_t the control action taken at t, then the state vector of the system at instant $t+1$ (the instant corresponding to the next sampling) is given by :

[1] We restrict our attention to discrete time to keep things as simple as possible, even though many of the ideas can be extended to the continuous-time case (e.g. see [3] or [10]).

[2] Introduction of stochastic aspects as well as terminal states are almost straightforward but burdens the notations.

$$x_{t+1} = f(x_t, u_t) \tag{1}$$

The RL methods we use in this paper belong to the *temporal-difference* type of methods that suppose the existence of a *reward* r_{t+1} associated to the transition from x_t to x_{t+1} while taking action u_t ([13]).

We define the *discounted return* $R(x_0, u_t)$ which depends on the initial data x_0 and on the control $u_t \in U$ ($0 \leq t < \infty$) where U represents a finite set of possible values for u_t :

$$R(x_0, u_t) = \sum_{k=0}^{+\infty} \gamma^k r_{k+1} \tag{2}$$

where γ is a parameter, $0 \leq \gamma < 1$, called the *discount rate*.

The aim of reinforcement learning methods in the framework of infinite-time horizon with discounted reward is to find the optimal control $u_t^* \in U$ ($0 \leq t < \infty$) that maximizes the *discounted return*.

Value function

We define the *value function* $V(x)$, the maximum value of expression (2) as a function of the initial state at $t = 0$:

$$V(x) = \max_{\substack{u_t \\ 0 \leq t < \infty}} R(x, u_t) \tag{3}$$

Using the dynamic programming principle (introduced in [2]), we can prove that the value function satisfies the condition :

$$V(x) = \max_{u \in U}[r(x,u) + \gamma V(f(x,u))] \tag{4}$$

where $r(x,u)$ and $f(x,u)$ are respectively the reward observed and the next state reached when taking action u while being in state x.

Dynamic programming computes the value function in order to find the optimal control with a feed-back control policy. Indeed, from the value function we deduce the following optimal feed-back control policy :

$$u^*(x) \in \arg\max_{u \in U}[r(x,u) + \gamma V(f(x,u))] \tag{5}$$

Q function

We define the Q *function*, function of x and u as :

$$Q(x,u) = r(x,u) + \gamma V(f(x,u)), \tag{6}$$

Then $V(x)$ can be expressed as a function of $Q(x,u)$:

$$V(x) = \max_{u \in U}[Q(x,u)] \tag{7}$$

Equation (5) can be rewritten :

$$u^*(x) \in \arg\max_{u \in U}[Q(x,u)] \tag{8}$$

Equation (8) provides a straightforward way to determine the optimal control law from the knowledge of the Q *function*.

Objective of reinforcement learning algorithms

Reinforcement learning algorithms estimate the Q function by interacting with the system. From the knowledge of the Q function, they can decide by using equation (8) which value of the command to associate to a state in order to maximize the *discounted return* (defined by eqn. (2)).

Unfortunately, RL in a continuous state-space implies that the Q function has to be approximated ([13]). We have used a *discretization* technique to approximate it because it is easy to implement, numerically stable and allows the use of model learning algorithms.

Discretization technique

A discretization technique consists in dividing the state space into a finite number of regions and then considering that on each region the Q function depends only on u. Then, in the RL algorithms, the notion of state used is not the real state of the system (x) but rather the region of the state space to which x belongs. We will use the letter s rather than x to denote the state of the system in order to stress that we refer now not to x itself but to a region of the state space. Moreover, the finite set containing all the discretized states of the system is denoted by S. Figure 1a illustrates this.

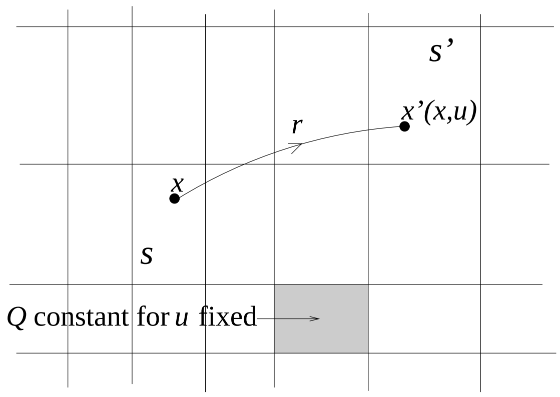
The discretization of the state space introduces some stochastic aspects. While being in one region of the state space and taking an action, the region of the state space reached at the next sampling instant is undetermined as illustrated on figure 1b.

The stochastic aspects introduced by the discretisation lead to suppose that $Q(s,u)$ does not obey anymore to the deterministic equation (6) but rather to

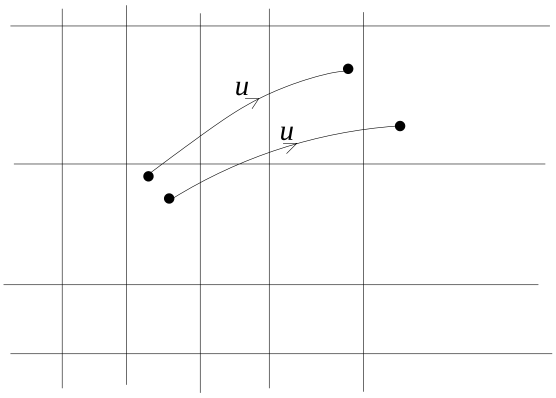
$$Q(s,u) = r(s,u) + \gamma \sum_{s'} p(s'|s,u) \max_{u \in U} Q(s',u) \tag{9}$$

where $p(s'|s,u)$ represents the probability to reach at the next sampling instant the state s' when being in the state s while taking action u.

Rewards $r(s,u)$ and probabilities $p(s'|s,u)$ describe the model of the discretized system. They associate to each discretized state and to each value of the command u transition probabilities to other states and the value of a reward. Assuming that they describe a Markov Decision Process (MDP), $Q(s,u)$ can be easily estimated using a classical Dynamic Programming (DP) technique like *value iteration* or *policy iteration* ([2]). The optimal control to associate to a state is then the one that maximizes Q for this state.



(a) Discretization



(b) Stochastic aspect

Figure 1: Discretization of the state space

RL methods either estimate the transition probabilities and the associated rewards (model learning methods) and then compute the Q function, or compute directly the Q function without learning any model (non-model learning methods).

Non-model learning method

The controller observes in which state s the system is, decides a value for the command u and observes which state s' is reached at the next sampling instant. Moreover, it has the knowledge of the reward r associated to the transition $s \to s'$. A non-model learning algorithm known as Q-learning [3]. ([14]) is provided in Table 1.

Table 1: Non-model learning algorithm (Q-learning)

```
Initialise $Q$ arbitrarily for all $s$ and $u$
Observe $s$
Repeat indefinitely
    Choose action $u \in U$ from $s$ using the knowledge of $Q$
    (e.g. $\epsilon$-greedy method)
    Take action $u$ and observe $s'$ and $r$
    $Q(s,u) \leftarrow Q(s,u) + \alpha[r + \gamma \max_{u \in U} Q(s',u) - Q(s,u)]$
    $s \leftarrow s'$
```

The ϵ-greedy method used to choose the action u suggests that there is a probability ϵ that the action chosen is not necessarily the one which maximizes Q, but an action taken at random. This provides the algorithm with some exploratory behaviour such that on average each $1/\epsilon$ time a random action is taken ([8]).

The parameter α represents the amount of the error corrected. It should be large enough to allow a fast convergence of the algorithm and sufficiently small to avoid an instability of the algorithm. It can be shown that the Q-learning algorithm converges to the exact solution in the framework of an MDP if an ϵ-greedy policy is used and if α satisfies the stochastic approximation ([13]). For a non linear system, convergence to the exact solution can only be stated if the size of the discretized states tends to zero. But then the learning time of the algorithms would be infinite.

The algorithm implemented to control the FACTS is a more elaborated version of the Q-learning algorithm known as Q-learning(λ) aimed to speed up the convergence of the method ([12]).

Model learning method

Table 2: Model learning algorithm

```
Initialize $Q(s,u) = 0 \quad \forall s \in S$ and $\forall u \in U$
Initialize parameters of the model :
    $N(s'|s,u) = 0 \quad \forall s, s' \in S$ and $u \in U$
    $p(s'|s,u) = 0 \quad \forall s, s' \in S$ and $u \in U$
    $r(s,u) = 0 \quad \forall s \in S$ and $u \in U$
Observe $s$
Repeat indefinitely
    Choose action $u \in U$ from $s$ using the knowledge of $Q$
    (e.g. $\epsilon$-greedy method)
    Take action $u$ and observe $s'$ and $r$
    Update the model :
        $N(i|s,u) \leftarrow \beta N(i|s,u) \quad \forall i \in S$
        $r(s,u) \leftarrow \frac{r(s,u)\sum_{j \in S} N(j|s,u) + r}{\sum_{j \in S} N(j|s,u) + 1}$
        $N(s'|s,u) \leftarrow N(s'|s,u) + 1$
        $p(i|s,u) \leftarrow \frac{N(i|s,u)}{\sum_{j \in S} N(j|s,u)} \quad \forall i \in S$
    Compute $Q$ by solving equations (9)
    $s \leftarrow s'$
```

A generic algorithm for model learning method is given in Table 2[4]. . The N function used in this algorithm do not intervene to describe the model as such but is necessary for its updating. The term β ($0 \leq \beta \leq 1$) provides the algorithm with some adaptive behaviour (necessary for a non-autonomous system) by giving more importance if $\beta < 1$ to the last data acquired from the system.

The version of the model learning method algorithm implemented to control the FACTS is known as Prioritized

[3]The Q-learning denomination for this algorithm is widely used in the RL literature even if the algorithm learns just an approximation of the Q-function. The correct term should be *Q-function approximation learning*

[4]The algorithm learns an approximation of the model of the system, that is an approximation of the p and r functions. A more correct denomination for it would be *model-approximation learning-algorithm*

Sweeping ([9]). This particular type of algorithm reduces the time needed to estimate the Q function from the model by refreshing its value for only a fraction of the states.

Notion of state

Although in this section the notion of state s used in the RL algorithms has been closely linked to the state vector x of the system (s represents the discretized part of the state space in which x is situated), the algorithms described in Table 1 and 2 are often used in a partially observable environment (x partially unknown). The loss of information on the state of the system and the use of discretization techniques to handle continuous variables are two major sources of approximations present in RL techniques and imply that nothing can be stated about the convergence of the algorithms or the quality of the closed-loop control law finally observed.

3 Power system description

The control procedure is illustrated on a four-machine power system modelled with more than 50 state variables. The system characteristics are inspired from [7]. It is represented in figure 2. The type of FACTS used is a Thyristor Controlled Series Capacitor (TCSC), particularly well suited to damp power system oscillations due to its ability to change rapidly the power transferred in a line ([6]). It is installed on a $240\,kV$ line connecting bus $\#7$ to bus $\#9$. Basically, the TCSC consists of a variable reactance (more

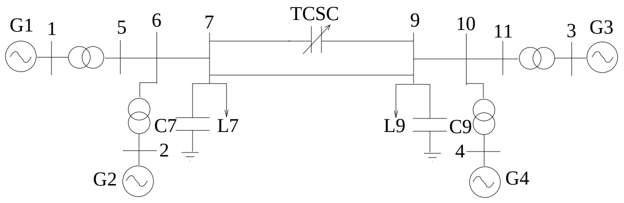
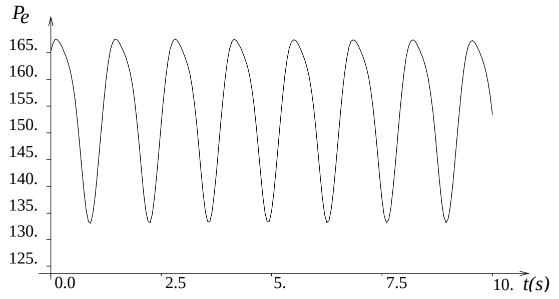


Figure 2: A four machines power system

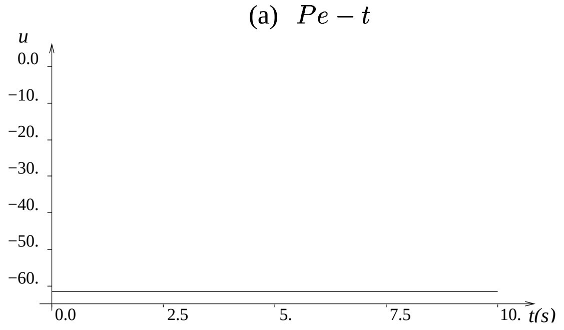
often a capacitance) placed in series with a line. The control variable u represents the *reactance reference* of the TCSC. The TCSC reactance can be different from the reference due to the delay necessary to adapt the reactance value to a new reference.

In our example the value of u is limited to the interval $[-61.5756, 0]\,(\Omega)$ which corresponds to a compensation of 30 % of the line when the FACTS acts at full range of its capacity.

If the TCSC acts just like a fixed capacitor, electrical power oscillations occur. Figure 3a represents the evolution over a period of 10 s of the electrical power transmitted through the TCSC while figure 3b represents the value of u over the same period of time. The magnitude of the electrical power oscillations is approximately $40\,MW$ and the average power transmitted is $150\,MW$. The aim of the reinforcement learning algorithms is to generate the appropriate variation of u to damp power system oscillations in the line. Next section discusses the strategies used to adapt algorithms described in section 2 to this power system oscillations problem.



(a) $Pe - t$



(b) $u - t$

Figure 3: TCSC acting like a fixed capacitor

4 Reinforcement learning applied to control

Application of the reinforcement learning algorithms to control the FACTS has two main difficulties. The first one is linked to the choice of an appropriate reward r such that the control algorithms that tend to maximize the value of the expression $\sum_{t=0}^{+\infty} \gamma^t r_{t+1}$ (see eqn. (2)) also damp the electrical power oscillations. The second difficulty concerns the necessity for the reinforcement learning algorithms to observe the entire state vector x of the system. Acquisition of such an amount of information is not realistic. The only information available to our controller will be the measurement at fixed rate of the electrical power transmitted in the line. Then a strategy commonly used in partially observable environments ([4]) will be used to compensate the loss of information. We will also discuss in this section how to discretise the command and how to define a quality of the control, an image of how well the algorithms have learned to control the FACTS.

State definition

The controller will act in a partially observable environment, the only measurements realized on the system being the electrical power P_e transmitted in the line. The state at time t (denoted by s_t) is represented by the history of the measurements realized on the power system and the actions taken by the controller.

Then $s_t = < \cdots, P_{e_{t-2}}, u_{t-2}, P_{e_{t-1}}, u_{t-1}, P_{e_t} >$. The larger the length of the history the more precise the information about the system dynamics available for the controller will be. Unfortunately, increasing history length also increases the time needed by the algorithms to converge. In the treated example, we decided to take s_t as the three last observations of the electrical power and the two last actions taken. State s_t is then composed of the 5-uple $< P_{e_{t-2}}, u_{t-2}, P_{e_{t-1}}, u_{t-1}, P_{e_t} >$. Moreover, the value of the electrical power will have to be discretized.

The discretization step will be $5\,MW$.

Sampling period and value of γ

The period between two samplings of the electrical power is chosen equal to $50\,ms$ which means that the value of u could change every $50\,ms$. Moreover the value of γ is chosen equal to 0.98. If γ is too small, the reinforcement learning algorithm takes short-term benefit control actions. If γ is too large the algorithm converges slowly ([13]). Simulations carried out have shown that $\gamma = 0.98$ represents a reasonable tradeoff.

Reward definition

The reward must express how the controller is expected to act. In the considered example, the aim of the controller is to limit the magnitude of the electrical power oscillations in the line. One strategy is to consider that the variations of the electrical power around its average should be minimised. This can be achieved by choosing the reward r_t equal to $-|P_{e_t} - \overline{P_e}|$. Indeed, the algorithms will tend to maximize the sum $\sum_{t=0}^{+\infty} -\gamma^t |P_{e_t} - \overline{P_e}|$ and then to produce some damping of P_e.

Command

As mentioned in section 2, the number of possible values for the command u must be discrete. Without this restriction the value of u could be any in the interval $[0, -61.5756]$. The more accurate the interval discretization, the better the quality of the control and the larger the convergence time. We will study two variants for the discretization. One uses a set of command U with 3 elements and the other with 5.

Quality of the control

The *discounted return at time t* is defined as

$$R_t = \sum_{i=t}^{+\infty} \gamma^{i-t} r_{i+1}$$

and is a measure of how good the control acts after controlling the system for a duration t. The higher the return, the better the control procedure. To give a quality to the control procedure at time t we will slightly modify this value by taking its average over the next minute (denoted by $\overline{R_t}$). This strategy has been used in order to avoid difficulties to visualize the results due to the high variance of R_t. With such a score measure, oscillations represented at 3a are credited of a value of $\overline{R}$ equal to -580. Then according to the equality $1 + \gamma + \gamma^2 + \cdots = \frac{1}{1-\gamma}$, we can compute the average of the deviation observed : $580(1-\gamma) = 11.6\,MW$.

Summary of the section

We summarize hereafter some important features used to apply RL algorithms to the FACTS control problem :

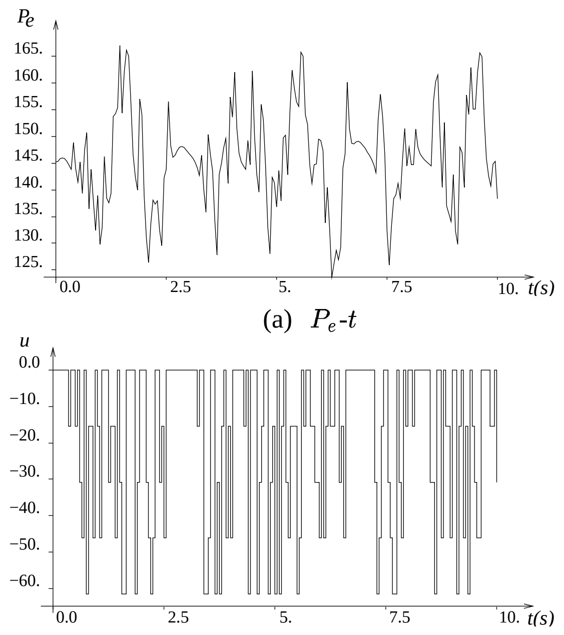
- Time between two successive measurements of P_e equals $50\,ms$.
- Time between index t and $t+1$ equals $50\,ms$.
- The state at time t is a 5-uple denoted by s_t with $s_t =< P_{e_{t-2}}, u_{t-2}, P_{e_{t-1}}, u_{t-1}, P_{e_t} >$.
- The reward $r_t = -|P_{e_t} - \overline{P_e}|$ where $\overline{P_e}$ represents the average of the electrical power transmitted in the line.
- The set U will be either composed by 3 of by 5 elements. If $\#U = 5$ then $U = \{0., -15.39, -30.78, -46.18, -61.57\}$. If $\#U = 3$ then $U = \{0., -30.78, -61.57\}$.
- The average score $\overline{R}_t$ at time t is equal to $\frac{\sum_{i=t}^{1199+t} R_i}{1200}$.
- $\gamma = 0.98$, $\epsilon = 0.001$, $\alpha = 0.1$ and $\beta = 1$.

5 Simulation results for an autonomous system

In this section, we will present simulation results obtained for model learning and non-model learning methods and for different sizes of the set U. These results refer to an autonomous and deterministic power system. First we show results obtained for a model-learning method with five possible values for the command u ($\#U = 5$). Then we illustrate the influence of the method used (model-learning vs non-model learning) and the size of U ($\#U = 5$ vs $\#U = 3$) on the speed of convergence and on the quality of the control.

Model-learning method with $\#U = 5$

The figures used to illustrate the results are to be compared with figures 3a and 3b corresponding to a TCSC acting like a fixed capacitor.



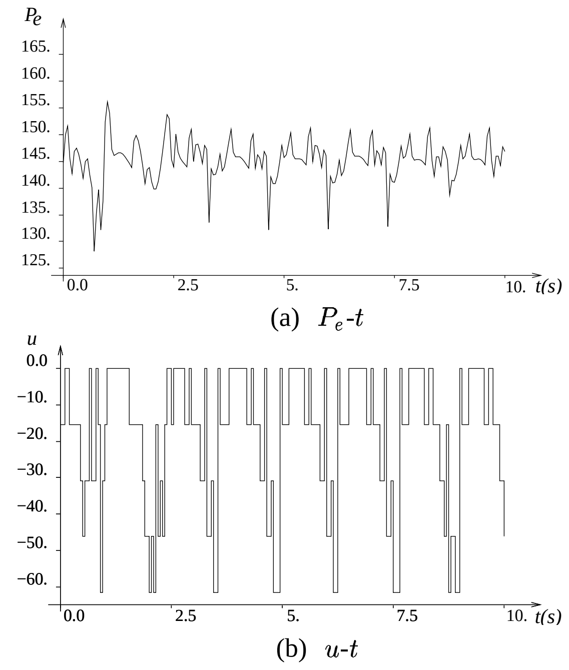
(a) P_e-t

(b) u-t

Figure 4: After 10 minutes of control

After $10\,min$ of control (the reinforcement learning algorithm has imposed each $50\,ms$ the value of u for already $10\,min$), we have drawn respectively on figures 4a and 4b the evolution of the electrical power transmitted in the line (P_e) and the action taken (i.e. the value of u) over a period of $10\,s$. The magnitude of the P_e oscillations is still very large and the evolution of the action u seems to be driven by an almost random process. The control algorithm does not have yet a sufficient knowledge about the system to act efficiently.

After $1\,h$ of control (see figures 5a and 5b), the electrical power transferred in the line starts being well damped. An organized structure appears in the sequence of actions taken. After $10\,h$ of control (see figures 6a and 6b), the results are more impressive. The magnitude of the electrical power oscillations has strongly decreased. The variation of the control variable u has a periodic behaviour of approximately the same frequency ($0.8\,Hz$) as the electrical power oscillations observed when no control occurs. The harsh aspect of the electrical power observed comes from the discontinuous variation of the command variable u. Such behaviour could be circumvented by increasing the time delay of the FACTS (image of the time needed by the TCSC to meet the reactance reference u) or by imposing a continuous variation of u by means of an integrator.


(a) P_e-t

(b) u-t

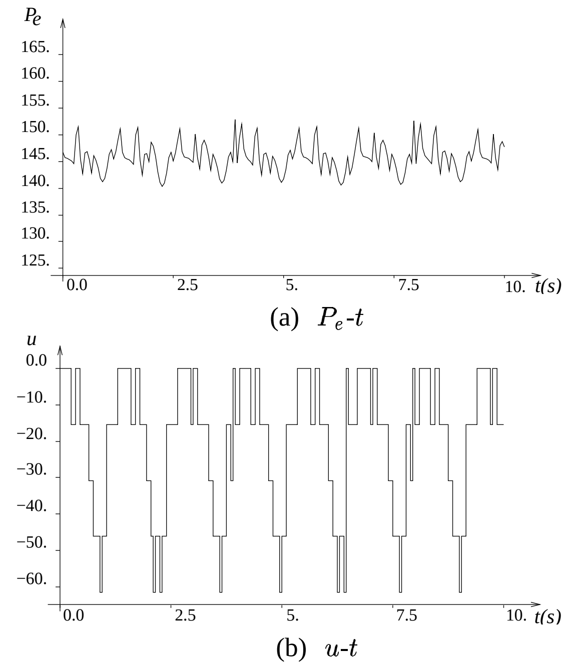
Figure 5: After 1 hour of control

Size of U, model and non-model learning methods

Figure 7 represents the evolution of the quality of the control $\overline{R}_t$ as a function of t for model-learning and non-model learning methods, both for 3 and 5 possible values for u ($\#U = 3$ or $\#U = 5$).

The best value of $\overline{R}$ obtained after $10\,h$ of control (which occurs for the model-learning method with 5 possible values for u) equals -90. This value has to be compared to -580 at $t = 0$. The deviation from the average for the electrical power corresponds to a change from $11.6\,MW$ to $1.8\,MW$ (see section 3).

For the same value of $\#U$, the figure shows that the quality of the control $\overline{R}_t$ is slightly better for the model learning method than the non-model learning one, whatever the value of t. For the same method, figure 7 also shows that an increase in $\#U$ improves the quality of the control finally obtained but penalises the speed of convergence of the algorithms.


(a) P_e-t

(b) u-t

Figure 6: After 10 hours of control

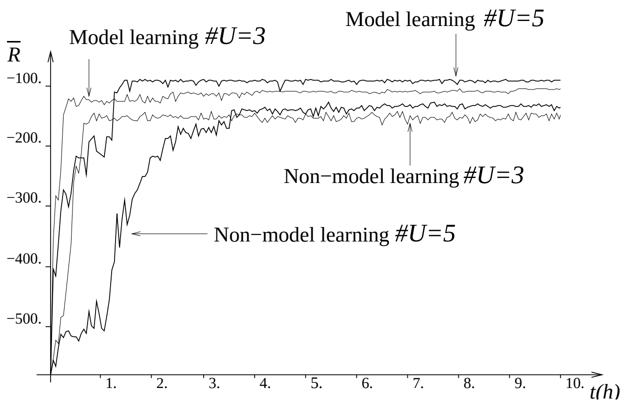


Figure 7: Quality of the control $\overline{R}_t$

Even if the model learning method is preferred the non-model learning one (for a comparison between model learning and non-model learning methods, see [1]), the choice of the size of U is more difficult due to the tradeoff between the quality of the control obtained and the speed of convergence of the algorithms. The best strategy would probably be to start the control with a small set of commands and then to increase it.

6 Non-autonomous and stochastic system

In the previous section, the RL algorithms have been used with an autonomous and deterministic system. Unfortunately, these two properties are not met in real power systems. Even under normal operating conditions (no short-circuits, no loss of a transmission element, $\cdots$) stochastic aspects linked to the load variation are common. Non-autonomous aspects are on the other hand mainly linked to the load variations and changes in generation patterns.

The same system as the one described in section 3 is used here to illustrate how RL algorithms behave in a stochastic and non-autonomous environment. For this, white noise has been introduced at each load and load variation is supposed to be cyclic with a 5h period[5].

Figure 8 represents the evolution of $\overline{R}$ over a period of $15\,h$ (t_i relative to each piece of curve indicates the time t that corresponds to the beginning of the curve). During the first $5\,h$ the TCSC is acting just like a fixed capacitor. The variation of $\overline{R}$ during these first five hours is only linked to the continuously changing operating conditions. After $5\,h$, the RL control procedure (model learning algorithm with $\#U = 5$, $\epsilon = 0.001$ and $\beta = 0.95$ (see Table 2)) starts acting.

An important feature that can be drawn from this figure is that even after 1 cycle of control, the RL algorithm continues to improve its ability to damp electrical power oscillations efficiently. The more the same operating conditions are encountered, the better the damping obtained for these operating conditions tends to be. The RL algorithm behaves well for this non-autonomous system due to its capability to adapt itself fast to a new environment. Nevertheless, with a too fast changing environment, RL algorithm performances strongly decrease.

The stochastic aspect introduced by the loads has only a second order influence on the control quality due to the fact that RL algorithms are from the beginning shaped for control in stochastic environments (see eqn. (9)) and are thus well adapted to stochastic perturbations.

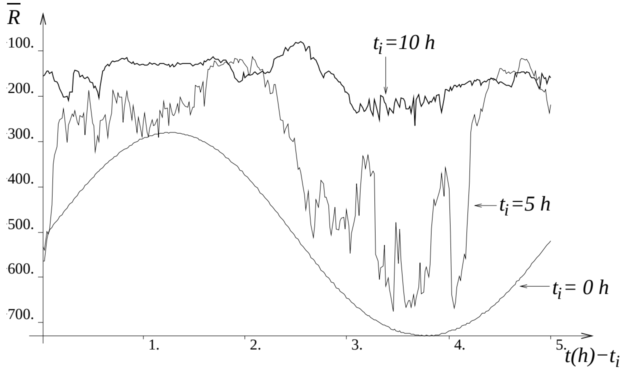


Figure 8: Quality of the control for a non-autonomous system

7 Conclusions and future prospects

The reinforcement learning algorithms used here were able to act correctly in order to damp power system oscillations. Model learning and non-model learning algorithms have been used with better results for the model learning ones. The non-autonomous environment in which the controller had to act did not foil the method due to its ability to adapt itself fast to changing operating conditions. Compared to classical control laws designed off-line, the controller gets rid of any modelling of the power system and adapts itself to a changing environment. Improvements of the control algorithms could still be done by using more elaborated RL techniques (like the use of local regression techniques, variable resolution discretization, more sophisticated methods to decide which action to choose, methods to define automatically s from the history of the observations and actions, $\cdots$) but are out of the scope of this paper.

[5]Equation $s(t) = s(0) - 0.3s(0)\sin(2\pi\frac{1}{5*3600}t) + \xi(t)$ models the load, where s stands for active or reactive parts of the load, and $\xi(t)$ is the white noise factor.

REFERENCES

[1] Christopher G. Atkeson and Juan Carlos Santamaria. A Comparison of Direct and Model-Based Reinforcement Learning. *International Conference on Robotics and Automation*, 1997.

[2] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[3] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[4] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994.

[5] Steven A. Harp, Sergio Brignone, Bruce F. Wollenberg, and Tariq Samad. SEPIA. A Simulator for Electric Power Industry Agents. *IEEE Control Systems Magazine*, 20(4):53–59, August 2000.

[6] Narain G. Hingorani and Laszlo Gyugyi. *Understanding FACTS.* IEEE press, 2000.

[7] P. Kundur. *Power System Stability and Control*. McGraw-Hill, 1994.

[8] Nicolas Meuleau. Exploration of Multi-State Environments: Local Measures and Back-Propagation of Uncertainty. *Machine Learning*, 35:117–154, 1999.

[9] A.W. Moore and Atkeson C.G. Prioritized Sweeping: Reinforcement Learning with Less Data and Less Real Time. *Machine Learning*, 13:103–130, 1993.

[10] R. Munos. A study of reinforcement learning in the continuous case by the means of viscosity solutions. *Machine Learning*, 40:265–299, 2000.

[11] Katsuhiko Ogata. *Discrete-Time Control Systems*. Prentice Hall, second edition, 1995.

[12] S.P. Singh and R.S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158, 1996.

[13] R.S. Sutton and A.G. Barto. *Reinforcement Learning, an introduction*. The MIT Press, 1998.

[14] C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.