Direct Model Predictive Control: A Theoretical and Numerical Analysis

Marie-Liesse Cauwet, Jérémie Decock, Jialin Liu, Olivier Teytaud

Direct Model Predictive Control: A Theoretical and Numerical Analysis

M-L. Cauwet*, J. Decock†, J. Liu‡ and O. Teytaud§
* Montefiore Institute - Department of EE&CS, University of Liège, Liège, Belgium
mlcauwet@ulg.ac.be
† CEA Saclay, DSM/Irfu/SAp, Gif-sur-Yvette, France
jeremie.decock@cea.fr
‡ Queen Mary University of London, UK
jialin.liu@qmul.ac.uk
§ TAO/Inria Saclay-IDF, Univ. Paris-Saclay, Gif-Sur-Yvette, France
olivier.teytaud@lri.fr; now working at Google

***Abstract*—This paper focuses on online control policies applied to power systems management. In this study, the power system problem is formulated as a stochastic decision process with large constrained action space, high stochasticity and dozens of state variables. Direct Model Predictive Control has previously been proposed to encompass a large class of stochastic decision making problems. It is a hybrid model which merges the properties of two different dynamic optimization methods, Model Predictive Control and Stochastic Dual Dynamic Programming. In this paper, we prove that Direct Model Predictive Control reaches an optimal policy for a wider class of decision processes than those solved by Model Predictive Control (suboptimal by nature), Stochastic Dynamic Programming (which needs a moderate size of state space) or Stochastic Dual Dynamic Programming (which requires convexity of Bellman values and a moderate complexity of the random value state). The algorithm is tested on a multiple-battery management problem and two hydroelectric problems. Direct Model Predictive Control clearly outperforms Model Predictive Control on the tested problems.**

***Index Terms*—Dynamic Optimization, Power System Management, Predictive Control, Theoretical Analysis.**

I. Introduction

Energy management problems are a class of sequential decision making problems under uncertainties. Nowadays, the underlying stochastic process (e.g. production and consumption uncertainties, market prices forecasts) gains in complexity. Different research domains handle such systems, such as mix integer programming or stochastic programming, however these methods are limited to represent complex stochastic processes [2]. Section I-A formalizes the sequential decision making under study in this paper. Section I-B dresses an overview of the best algorithms dedicated to sequential decision making problems and their limitation.

A. Formalism of Markov Decision Processes

The sequential decision making problem is modeled by a Markov Decision Process. The dynamical system under study is in a state, and evolves to a new state, depending on random variables and decisions. After a given number of such transitions, the system is stopped. Each transition provides a cost, and the cost is cumulated from the initial state to this stopping time. More formally, given:

- $\mathcal{D}$: set of time steps at which a decision is made;
- $\overline{\mathcal{D}}$: set of time steps at which no decision is made;
- $\mathcal{S} \subset \mathbb{R}^d$: set of states;
- $L_t(s)$: set of legal actions in state s at time t;
- Ω: set of random variables;
- an initial state $s_0 \in \mathcal{S}$;
- a final step time T;
- a sequence of random variables $\omega_0, \dots, \omega_{T-1} \in \Omega$;
- a policy $\Pi : \mathcal{S} \times \mathcal{D} \to L_t(s)$;
- a transition function $\mathfrak{T}_{nl} : \mathcal{S} \times \mathcal{D} \times L_t(s) \to \mathcal{S}$;
- a random transition function $\mathfrak{RT} : \mathcal{S} \times \overline{\mathcal{D}} \times \Omega \to \mathcal{S}$;
- a cost function $C_{nl,t} : \mathcal{S} \times L_t(s) \to \mathbb{R}$;

we define:

$$\forall t \in \mathcal{D},\ a_t = \Pi(s_t, t) : \text{the decision at } t \quad (1)$$

$$\forall t \in \mathcal{D},\ s_{t+1} = \mathfrak{T}_{nl}(s_t, t, a_t) \quad (2)$$

$$\forall t \in \overline{\mathcal{D}},\ s_{t+1} = \mathfrak{RT}(s_t, t, \omega_t) \quad (3)$$

$$\forall t \in \mathcal{D},\ c_t = C_{nl,t}(s_t, a_t) : \text{the cost at time } t \quad (4)$$

$$C_\Pi = \sum_{t \in \mathcal{D}} c_t : \text{the total cost function.} \quad (5)$$

Eq. 2 refers to decision-based transitions, whereas Eq. 3 refers to randomized transitions. In all the following, we consider: a given transition function $\mathfrak{T}_{nl}$ and a linear approximation $\mathfrak{T}$ of $\mathfrak{T}_{nl}$; a given cost function $C_{nl,t}$ and a linear approximation C_t of $C_{nl,t}$.

Then C_Π is a random variable, only depending on Π. Dynamic optimization is the search for an approximation of Π^* such that $\mathbb{E}[C_{\Pi^*}]$ is as small as possible:

$$\Pi^* \in \arg\min_{\Pi} \mathbb{E}[C_\Pi]. \quad (6)$$

B. State of the Art in Dynamic Optimization

Dynamic Programming methods have been proposed back to the 50's to solve this problem, with the pioneering work of [3]: the Stochastic Dynamic Programming (SDP). It is

based on computing a value function backwards in time. For a given state, the value function (also termed Bellman function) provides the expected reward that an agent will get, if it starts in this state and makes optimal decisions. Without enhancements, SDP is only tractable for simple problems.

Dual SDP (SDDP) [12] is the best known improved variant of SDP. Convexity of Bellman values and a random process of moderate state space size [16] are required for applying this method. It consists in approximating the value function by linear cuts. Assuming that the Bellman function is convex, a piecewise linear approximation can be obtained thanks to subgradient values at various locations in the state space. SDDP runs simulations and computes subgradients in the visited states. The moderately sized state space of the random process is a key assumption. The Bellman function should be indexed by all state variables, including the variables describing the state of the random process. This is often intractable, hence the random process is often heavily simplified, in particular by stage-wise independence [16], or at least the random process is replaced by a small Markovian representation.

An alternative to dynamic programming is the use of Model Predictive Control (MPC) [4]. The principle is as follows: each time a decision is required, a forecasting modules is applied for estimating the random variables of the next h time steps, where h is the tactical horizon. Then, all uncertainties are removed – just assuming that the forecasts are exact for the next h time steps. Then, the decision which optimizes the reward over the next h time steps is made. This methodology is quite simple and the assumptions are clearly understood: we assume that the forecasts are perfect and that the effects beyond the horizon h can be neglected. There are many methods for mitigating these two assumptions. Regarding the first one, it is customary to restart the optimization as soon as you get new observations or to replace forecasts by pessimistic estimates in order to reduce the risk. On the other hand, adding an approximate value function for the reward beyond the horizon h mitigates the effect of the second assumption. MPC is quite convenient, robust, simple and remains a main tool in spite of theoretical shortcomings. In particular, it often outperforms [20] the far more sophisticated SDDP which relies on a modelization by a usually stage-wise independent random process [16] or at least a Markov process with moderate size.

Another possibility is to use Direct Policy Search (DPS) [13], which optimizes a parametric function used as a policy of the Decision Process. This parametric function (e.g. a neural network) can be more or less problem specific. Compared to the above approaches, DPS has the strength that it does not need a simplified model: simulation-based optimization of the policy can be applied as soon as simulations are possible. On the other hand, traditional DPS fails in front of huge action spaces unlike all previously presented approaches which can deal with huge constrained action spaces provided that all involved functions are linear or can be encoded in linear programming (or in other frameworks with moderate complexity). Main applications considered in our work are made of huge constrained problems (unit commitment problems) whose action space is too big for DPS.

Direct Model Predictive Control (DMPC) has been proposed in [6] to relax the aforementioned assumptions. DMPC is a hybrid model which merges the properties of the above methods. The structure of SDDP remains, but the Bellman values are replaced by a parametrized function, e.g., a neural network, optimized offline. Notably, no assumption on the convexity of the reward is required. Besides, the stochastic setting is handled (contrarily to simple forms of MPC) and no assumption on the Markovian nature of the stochastic processes is required (contrarily to SDDP).

Section II introduces the DMPC method. We carry out a theoretical analysis of DMPC in Section III and prove that, provided that the noisy optimization routine converges, DMPC reaches an optimal policy. Section IV presents a test on a multiple-battery management problem and two hydroelectric problems. Note that, compared to the results of [6] on multiple-battery management problem, an extended version of DMPC is implemented (neural network parametrizations, more evaluations).

II. Direct Model Predictive Control (DMPC)

A. Formulation

This section is devoted to the presentation of DMPC, formulated in Alg. 1. It consists into selecting the next action according to a policy Π_θ, parametrized by θ, defined as follows:

$$\forall\,(s_t,t)\in\mathcal{S}\times\mathcal{D},\ \Pi_\theta(s_t,t)\in\underset{a_t\in L_t(s_t)}{\arg\min}\ C_t(s_t,a_t)+\mathcal{N}_\theta[s_t,\mathfrak{T}(s_t,t,a_t),t]. \tag{7}$$

$\mathcal{N}_\theta$ is a mapping parametrized by θ, such that for all $(s_t,s'_t,t)\in\mathcal{S}\times\mathcal{S}\times\mathcal{D}$, $\mathcal{N}_\theta[s_t,s'_t,t]\in\mathbb{R}$. In particular, we perform the optimization (Alg. 1, Line 3) using the linear approximations C and $\mathfrak{T}$, and not the 'real' transition and cost functions $C_{nl,\cdot}$ and $\mathfrak{T}_{nl}$ respectively. We will see later how this a priori suboptimality can be mitigated thanks to the valorization term $\mathcal{N}_\theta(.)$.

Algorithm 1 Direct Model Predictive Control (DMPC): Simulation

Input:
an initial state s_0;
a policy Π_θ as in Eq. 7;
a generator of random sequences get_random$(\cdot)$
a final time step T
Output:
a cost $C_{\Pi_\theta} := DMPC(s_0,\Pi_\theta,T,\text{get\_random}(\cdot))$

$C_{\Pi_\theta}\leftarrow 0$
$t\leftarrow 0$
1: **while** $t<T$ **do**
2: **if** $t\in\mathcal{D}$ **then**
3: $a_t\leftarrow\Pi_\theta(s_t,t)$ as in Eq. 7 ▷ **Linear** cost and transition functions
4: $s_{t+1}\leftarrow\mathfrak{T}_{nl}(s_t,t,a_t)$ ▷ **Non-linear** transition function
5: $C_{\Pi_\theta}\leftarrow C_{\Pi_\theta}+C_{nl}(s_t,a_t)$ ▷ **Non-linear** cost function
6: **else**
7: $\omega_t\leftarrow\text{get\_random}(t)$
8: $s_{t+1}\leftarrow\mathfrak{RT}(s_t,t,\omega_t)$
9: **end if**
10: $t\leftarrow t+1$
11: **end while**
Return C_{Π_θ}

When $\mathcal{N}_\theta$ is handcrafted rather than optimized, this is MPC. When $\mathcal{N}_\theta$ is computed backwards, this is SDP. When $\mathcal{N}_\theta$ is approximated by linear cuts built on subgradients extracted from simulations, this is SDDP. The mathematical analysis just assumes that $\mathcal{N}_\theta$ can approximate any necessary function, provided that θ is correctly chosen. This assumption will be formalized in Theorem 1. In practice, in the experiments presented in Section IV, $\mathcal{N}_\theta$ is a neural network, chosen for the approximation properties of neural nets. θ is then selected such that:

$$\theta^* = \arg\min_\theta \ C_{\Pi_\theta} \tag{8}$$

In practice, θ is optimized by offline simulations describes in Alg. 2.

Algorithm 2 Offline optimization of DMPC parametrization.

Input:
an initial state s_0;
an initial parameter θ_0;
a black box noisy optimization algorithm Opt, e.g. [8], [15], [7], [10], [17], [18]
a generator of random sequences get_random$(\cdot)$
a final time step T

Output:
an optimal parameter θ^*

$\theta^* \leftarrow \theta_0$
$n \leftarrow 0$
while budget not exhausted **do**
  $C_{\Pi_{\theta_n}} \leftarrow DMPC(s_0, \Pi_{\theta_n}, T, \text{get\_random}(\cdot))$
  $\theta_{n+1} \leftarrow \text{Opt}(C_{\Pi_{\theta_0}}, \dots, C_{\Pi_{\theta_n}}, \theta_0, \dots, \theta_n)$
  $n \leftarrow n+1$
end while
  Return $\theta^* := \theta_n$

In addition, we assume that $L_t(s_t)$ is defined by linear constraints, i.e., $\forall (s_t, t) \in \mathcal{S} \times \mathcal{D}$, there exist some matrix $A = A(t, s_t)$ and $B = B(t, s_t)$ such that $L_t(s_t) = \{a_t; As_t \geq a_t \text{ and } Bs_t = b\}$.

B. DMPC Brings Consistency into MPC

First and foremost, no assumption on the convexity of the reward is required, contrarily to SDDP and variants of dynamic programming based on convex piecewise linear functions. Also there is no assumption on the Markovian nature of the stochastic processes involved (contrarily to SDDP). Besides, we work in a stochastic setting, contrarily to brute force conformant planning or simple forms of MPC.

III. Consistency Analysis

We show that the DMPC structure includes optimal policies under the following linearity assumptions on the approximations $\mathfrak{T}$ and C of $\mathfrak{T}_{nl}$ and $C_{nl,.}$ respectively:
(A1) $\forall (s_t, t) \in \mathcal{S} \times \mathcal{D}$, $a_t \mapsto \mathfrak{T}(s_t, t, a_t)$ is linear;
(A2) $\forall (s_t, t) \in \mathcal{S} \times \mathcal{D}$, $a_t \mapsto C_t(s_t, a_t)$ is linear.
Theorem 2 states that the DMPC method is optimal under the extra assumptions that the 'real' transition function and cost functions are linear, i.e., $\mathfrak{T} = \mathfrak{T}_{nl}$ and $C_t = C_{nl,t}$ respectively. Then, Section III-B (Theorem 3) shows that under an additional technical hypothesis, these extra assumptions can be relaxed.

A. Optimality of DMPC

In all the paper, $\|.\|$ denotes the standard Euclidean norm and $(.|.)$ denotes the scalar product. In Section III-A, and only in this section, $\mathfrak{T} = \mathfrak{T}_{nl}$ and $C_t = C_{nl,t}$.

Definition 1: A mapping $p : \mathcal{S} \times \mathcal{D} \to \mathcal{S}$ will be termed consistent if for all $(s_t, t) \in \mathcal{S} \times \mathcal{D}$, there exists an action $a_t \in L_t(s_t, t)$ such that $\mathfrak{T}(s_t, t, a_t) = p(s_t, t)$.

Theorem 1 (DMPC policies can follow arbitrary prescriptions): Under assumptions (A1) and (A2), for all consistent mapping $p : \mathcal{S} \times \mathcal{D} \to \mathcal{S}$, there exists a mapping $\mathcal{N}_\theta$ such that:

$$\forall (s_t, t) \in \mathcal{S} \times \mathcal{D}, \ \forall \ a \in \arg\min_{a \in L_t(s_t)} C_t(s_t, a) + \mathcal{N}_\theta[s_t, \mathfrak{T}(s_t, t, a), t],$$
$$\mathfrak{T}(s_t, t, a) = p(s_t, t).$$

Furthermore, $\mathcal{N}_\theta[s, ., t]$ can be written as a max of $(d+1)$ linear functions, where d is the dimension of $\mathcal{S}$. The mapping p is termed a prescription function.

Proof 1: Let p be a prescription function. In the following, we use Lemma 1 (see Appendix A for the proof).

Lemma 1: There exist $d+1$ unit vectors $w_1, \dots, w_{d+1}$ in $\mathbb{R}^d$, and there exists a constant $c > 0$ such that, for any unit vector $u \in \mathbb{R}^d$, there exist $i \in \{1, \dots, d+1\}$ such that $(u|w_i) > c$.

For all $s \in \mathcal{S}$, we define $v(s) = \max_{i \in \{1,\dots,d+1\}} (s|w_i)$, where $(w_i)_{i \in \{1,\dots,d+1\}}$ are as in Lemma 1. By Lemma 1, there exists $c > 0$ such that,

$$\forall s \in \mathcal{S} \setminus \{0\}^d, v(s) > c\|s\|. \tag{9}$$

Under assumptions (A1) and (A2), by applying [14, Theorem 10.5 P126-127], the function $s' \mapsto \min_{a|s'=\mathfrak{T}(s_t,t,a_t)} C_t(s_t, a_t)$ is Lipschitzian, i.e, there exists a constant $D = D(s_t) > 0$ such that $\forall \ (s_1, s_2) \in \mathcal{S} \times \mathcal{S}$,

$$\left| \min_{a_t \in L_t(s_t) | s_1 = \mathfrak{T}(s_t,t,a_t)} C_t(s_t, a_t) - \min_{a_t \in L_t(s_t) | s_2 = \mathfrak{T}(s_t,t,a_t)} C_t(s_t, a_t) \right| \leq D\|s_1 - s_2\|. \tag{10}$$

For a constant $D > 0$ fixed in Eq. 10, we define:

$$\forall \ (s, s', t) \in \mathcal{S} \times \mathcal{S} \times \mathcal{D}, \ \mathcal{N}_\theta[s, s', t] := \frac{2D}{c} v(s' - p(s, t)). \tag{11}$$

Then the minimum of $s' \mapsto \mathcal{N}_\theta[s, s', t]$ is reached in $s' = p(s, t)$ and $\mathcal{N}_\theta[s, ., t]$ is written as a max of $(d+1)$ linear functions. Define $f(a) = C_t(s_t, a) + \mathcal{N}_\theta[s_t, \mathfrak{T}(s_t, t, a), t]$ and

$$a^* \in \arg\min_{a_t \in L_t(s_t) | \mathfrak{T}(s_t,t,a_t) = p(s_t,t)} C_t(s_t, a_t) + \mathcal{N}_\theta[s_t, \mathfrak{T}(s_t, t, a_t), t].$$

Such an a^* exists because we have assumed that the prescription is consistent. Then

$$f(a^*) = C_t(s_t, a^*) + \mathcal{N}_\theta[s_t, p(s_t, t), t] = C_t(s_t, a^*).$$

Let $a' \in L_t(s_t)$ be an action satisfying:

$$a' \in \arg\min_{a_t \in L_t(s_t)} f(a_t).$$

Define $s' = \mathfrak{T}(s_t, t, a')$. To conclude, we need to prove that $s' = p(s_t, t)$. By definition, a' minimize f on $L_t(s_t)$ and a^* minimize f on a subset of $L_t(s_t)$, therefore,

$$\begin{aligned}
&f(a^*) \geq f(a'),\\
&\geq C_t(s_t, a') + \mathcal{N}_\theta[s_t, s', t],\\
&\geq f(a^*) - D\|s' - p(s,t)\| + \mathcal{N}_\theta[s_t, s', t] \text{ by Eq. 10,}\\
&\geq f(a^*) - D\|s' - p(s,t)\| + 2D\|s' - p(s,t)\| \text{ by Eqs. 9, 11,}\\
&\geq f(a^*) + D\|s' - p(s,t)\|.
\end{aligned}$$

Hence $s' = p(s,t)$, which is the expected result. □

Theorem 2 (The structure of DMPC policies includes optimal policies): Let us assume (A1) and (A2). Assume that $C_t = C_{nl,t}$, $\mathfrak{T} = \mathfrak{T}_{nl}$and that an optimal policy exists[1]. Then there exists a mapping $\mathcal{N}_\theta$ such that Π_θ minimizes $\Pi \to \mathbb{E}[C_\Pi]$ i.e. $\forall \Pi, \mathbb{E}[C_\Pi] \geq \mathbb{E}[C_{\Pi_\theta}]$. In addition, $\mathcal{N}_\theta[s, ., t]$ can be written as a max of $(d+1)$ linear functions, where d is the dimension of $\mathcal{S}$.

Proof 2: By assumption, there exists Π^* which minimizes $\Pi \to \mathbb{E}[C_\Pi]$. Let $p^* : \mathcal{S} \times \mathcal{D} \to \mathcal{S}$ be the prescription function defined by:

$$\forall (s_t, t) \in \mathcal{S} \times \mathcal{D},\ p^*(s_t, t) = \mathfrak{T}(s_t, t, \Pi^*(s_t, t)). \tag{12}$$

Then, by Theorem 1, there exists $\mathcal{N}^*_\theta$ such that

$$\begin{aligned}
&\forall (s_t, t) \in \mathcal{S} \times \mathcal{D},\\
&\forall\ a_t^{dmpc} \in \underset{a_t \in L_t(s_t)}{\arg\min}\, C_t(s_t, a_t) + \mathcal{N}^*_\theta[s_t, \mathfrak{T}(s_t, t, a_t), t],\\
&\mathfrak{T}(s_t, t, a_t^{dmpc}) = p^*(s_t, t).
\end{aligned} \tag{13}$$

Let a_t^* be the action chosen by a given optimal policy Π^* at (s_t, t), i.e. $a_t^* = \Pi^*(s_t, t) \in \underset{\Pi}{\arg\min}\, \mathbb{E}[C_\Pi]$ and by definition, $C_\Pi = \sum_{t \in \mathcal{D}} c_t$. By Eq. 12,

$$p^*(s_t, t) = \mathfrak{T}(s_t, t, \Pi^*(s_t, t)) = \mathfrak{T}(s_t, t, a_t^*).$$

Together with Eq. 13, $\mathfrak{T}(s_t, t, a_t^{dmpc}) = \mathfrak{T}(s_t, t, a_t^*)$. Then

$$\mathcal{N}^*_\theta[s_t, \mathfrak{T}(s_t, t, a_t^{dmpc}), t] = \mathcal{N}^*_\theta[s_t, \mathfrak{T}(s_t, t, a_t^*), t]. \tag{14}$$

Hence, by definition of a_t^{dmpc}, a_t^* and Eq. 14, $\forall\ (s_t, t) \in \mathcal{S} \times \mathcal{D}$,

$$\begin{aligned}
C_t(s_t, a_t^{dmpc}) + \mathcal{N}^*_\theta[s_t, \mathfrak{T}(s_t, t, a_t^{dmpc}), t]&\\
\leq C_t(s_t, a_t^*) + \mathcal{N}^*_\theta[s_t, \mathfrak{T}(s_t, t, a_t^*), t],&\\
C_t(s_t, a_t^{dmpc}) \leq C_t(s_t, a_t^*).&
\end{aligned} \tag{15}$$

By Theorem 1, the distribution of $s_0, \ldots, s_T$ is the same with actions chosen by Π^* and with actions chosen by Π_θ. We show this by induction on the step time t:

- The initialization s_0 is the same in both cases.
- Induction:

(I1) By Eq. 13, $s_{t+1}|s_t$ is the same in both cases when $t \in \mathcal{D}$.

(I2) Both are Markov chains, hence $s_{t+1}|s_t$ is the same in both cases when $t \notin \mathcal{D}$.

[1]An optimal policy exists, by Bellman principle of optimality, as soon as relevant extrema in Bellman's equation are reached

The expected cost for Bellman's policy Π^* is therefore

$$\mathbb{E} \sum_{t \in \mathcal{D}} C_t(s_t, a_t^*),$$

whereas for DMPC it is

$$\mathbb{E} \sum_{t \in \mathcal{D}} C_t(s_t, a_t^{dmpc}).$$

These two summations are over the same distribution for s_t, and for each s_t, Eq. 15 shows that DMPC has a less or equal cost. This concludes the proof of Theorem 2. □

B. Non-linear Setting

In the previous section, Theorem 2 states that the DMPC methodology provides an asymptotically optimal solution to the Markov Decision Problem. To achieve this, it was assumed that the transition and cost functions $\mathfrak{T}_{nl}$ and $C_{nl,.}$ defined in Eqs. 2 and 4 are linear in the action variable. However, in real word applications, it happens that these assumptions are not satisfied [9], [19], [11]. With many algorithms, there is no solution for using non-linear transitions and costs. Usually, to mitigate the computational cost, a linear approximation is used instead of the non-linear original version. In this section, we want to emphasize that, while a linear approximation is necessary for having a polynomial computational cost, the advantage of simulation-based algorithms (such as DMPC) is that there is no need for *simulating* with the linear functions. Just the function used *inside the decision function* (Eq. 7) has to be linear. Thanks to the use, *in the simulation part* (Alg. 1, Lines 4-8), of the real (nonlinear) transition and cost functions, we get an optimal policy on the real problem, in spite of the use (for the sake of polynomial decision time) of linear functions in the policy function Π_θ. We show that DMPC can thereby accommodate such a case under some injectivity condition detailed in Theorem 3. We can still have an optimal policy if the $\mathcal{N}_\theta$ function is ad hoc. In some sense (made precise in Theorem 3), the simulations, performed with the 'real' transition $\mathfrak{T}_{nl}$ and cost $C_{nl,.}$, can offset the linear approximations of the non-linear functions.

Theorem 3 (DMPC can counterbalance the bias induced by its linear approximations): Under assumptions (A1) and (A2), and if for some optimal policy Π^*,

$$\forall (s_t, t) \in S \times \mathcal{D}, \exists!\ a_t \in L_t(s_t),\ \mathfrak{T}(s_t, t, a_t) = \mathfrak{T}(s_t, t, \Pi^*(s_t, t)), \tag{16}$$

where "$\exists!$" stand for "there exists a unique", then for all cost functions $c_0, \ldots, c_{T-1}$, there exists a mapping $\mathcal{N}_\theta$ such that the corresponding policy Π_θ minimizes $\Pi \mapsto \mathbb{E}[C_\Pi]$. Furthermore, $\mathcal{N}_\theta[s, ., t]$ can be written as a max of $(d+1)$ linear functions, where d is the dimension of $\mathcal{S}$.

Proof 3: Let us define Π^* the optimal policy, for the real transitions and costs (i.e. $\mathfrak{T}_{nl}$ and $C_{nl,t}$, and not their linear counterparts $\mathfrak{T}$ and C_t). The proof of Theorem 2 applies, except that instead of the prescription $p(s_t, t) = \mathfrak{T}_{nl}(s_t, t, \Pi^*(s_t, t))$ we use the prescription $p(s_t, t) = \mathfrak{T}(s_t, t, \Pi^*(s_t, t))$. The proof of Theorem 2 can be applied until Eq. 13. Eq. 13 and Eq. 16 imply that Π_θ plays optimal moves, and therefore it is optimal. □

IV. Experiments

A. Experiments with a 10 Batteries Problem

We first provide experiments on a buying/selling energy problem defined as follows. 10 batteries are used to store energy bought on a simulated market; stored energy can be resold on the market; market price varies realistically at each simulation time step (based on real data). Algorithms are evaluated on multiple simulations of 168 decision time steps (each one representing one hour in the simulation). They have to make one decision per battery and per decision time step (i.e. decide the quantity of electricity to either insert or extract of each battery) in order to find the policy that maximize the average profit (i.e. buy when the market price is low and sell when the market price is high). For each $t \in \mathcal{D}$, an approximation of the market price is made for the next 5 decision time steps so that a decision is made for the next 5 decision time steps at once. The energy market price changes for each simulation (the time series are randomly taken in a set of 1000 scenarios). For each battery, the maximum input and output capacity is 1000 kW and the maximum storage capacity is 96000 kWh. Batteries have different efficiencies (the percentage of energy actually sold for a given output command) and a smart control has to take it into account: the "output efficiency" of the i-th battery is $0.8 - \max\left(\frac{\lfloor i/2 \rfloor}{10}, 0.4\right)$.

Please note that previous sections have described algorithms for minimization problems but in this experiment algorithms are tested on a maximization problem thus notations are adapted.

Three algorithms are compared:

- MPC with a null valorization of the stock at tactical horizon (i.e. $\mathcal{N}_\theta(s, s', t) = 0$ in Eq. 7 with s the stock level of the batteries);
- MPC with the best constant marginal valorization of the stock at tactical horizon (i.e. $\mathcal{N}_\theta(s, s', t) = \alpha \cdot s'$ with α a constant optimized to maximize the total gain);
- DMPC, with a penalization (denoted $\mathcal{N}_\theta$ in Eq. 7) provided by a fully connected neural network, with various numbers of neurons on the hidden layer. According to Eq. 8, this neural network is optimized with a *Self-Adaptive Evolution Strategy* [5] using resampling principles (see [1]) for ensuring convergence in spite of noise (with n^2 evaluations at iteration n). The inputs to the neural network are the stock levels of the batteries, plus a cosine and a sine function of time with periodicity corresponding to 24 hours.

Results are presented in Fig. 1. They are satisfactory, but need a long optimization time, hence new policies (not a neural network) are designed before switching to large-scale problems in the next section (Section IV-B).

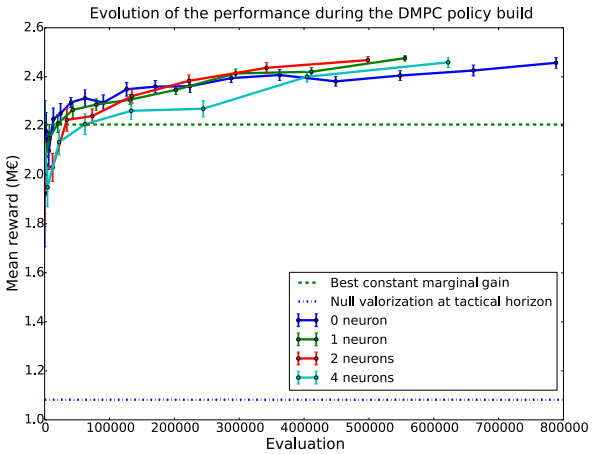

Figure 1. DMPC experimental results with 0, 1, 2, 4 neurons over 1000 scenarios, compared to the two baselines: a null valorization and the best constant marginal valorization at tactical horizon. Here, printed results are rewards (i.e. the higher the better). 0 neuron is almost immediately better than the baselines; later, 2 neurons perform well; 4 neurons need more computation time for outperforming baselines but eventually works well.

B. Experiments on a Real-World Hydroelectric Problem

1) Problem Description.: We here focus on a real world problem, the management of a hydroelectric dam, for national consumption, optimized jointly with many thermal power plants. The naive MPC algorithm does not perform well on this problem, due to the necessary long term management: a valorization term is necessary in the unit commitment, so that water is stored when the stock is low and/or there are still load peaks in the forthcoming months. Two algorithms are compared:

- The MPC improved by a long term management (LTM) constraint on stock level, optimized offline. This is the baseline (horizontal curve in our results), and needs an offline optimization phase (in the present case, combined with human expertise).
- The DMPC approach, with a handcrafted policy rather than a neural network. The handcrafted policy is described in Eq. 17.

$$\mathcal{N}_\theta(s, s', t) = (\theta_1 cos(t/T_0) + \theta_2 sin(t/T_0) + \theta_3 + \theta_4 s/S_0) \cdot s'. \quad (17)$$

where t is the time step, T_0 is one year, S_0 is a stock constant, s is a stock level, and $\mathcal{N}_\theta(s, s', t)$ is the valorization as in Eq. 7. The two first terms can represent any sinusoid of period $2\pi T_0$, the third is a constant marginal cost of stocks, and the fourth term is a first order approximation of the decrease of marginal costs when a stock increases (law of diminishing returns).

2) Discussion on the Long Term Management (LTM) Constraint.: When applying MPC to a real world problem with long term dependencies (e.g. an hydroelectric stock), it is clearly visible that using a tactical horizon corresponding to the horizon at which forecasts are excellent is not sufficient: costs (over the strategic horizon) become huge, because hydroelectric stocks are used way too early. Simulating such a system with 48 hours tactical horizon, without dealing with long term storage, is unrealistic - estimating the cost of a fictitious system with such a simple tool leads to overestimating its cost, in particular if it needs storage.

On the other hand, using a large tactical horizon is unrealistic; we do not have excellent long term forecasts for wind, load and market prices. The cost of a system simulated with one month anticipation is underestimated, in particular when there are many uncertainties - typically, the cost of a system with a lot of renewable energies is underestimated.

A simple solution is to use human expertise: define a constraint (LTM constraint), lower bound on stock levels, so that you are sure that some energy is saved up in dams. The LTM constraint used in our experiments has been defined by collecting human expertise. This approach leads to MPC costs empirically close to the real world costs; however, this method has the following drawbacks:

- The LTM constraint (extracted by humans by analysis of data) is not effective when the system is modified, and we need to simulate fictitious systems for making investment decisions.
- The one month tactical horizon used in the version designed by human experts is not realistic, as we certainly not have high precision forecasts over one month.
- A pernicious effect is that the LTM constraint is designed over data, and tested on the same data; this is typically an example of *overfitting*. Nothing guarantees that performance will be the same on other data than those used for designing the LTM constraint.

An alternate solution would be optimizing a LTM constraint on top of the MPC. However, this is expensive and leads to an unprincipled hard constraint on the stock. Indeed, the present work is motivated by the decision to get rid of the LTM constraint and its difficult assumptions.

3) Results.: Results are presented in Fig. 2. We observe that DMPC always outperforms the baselines (horizontal curves). In addition, the results are, to a large extent, independent of the LTM constraint after the optimization by DMPC, whereas the LTM constraint methodology without DMPC had a big impact on the result.

V. Conclusions

We have studied Direct Model Predictive Control, both mathematically and experimentally. The approach has various advantages and can in particular mitigate the difficulties of high scale dynamic optimization. High scale action spaces can be handled thanks to polynomial decision making, because the neural network provides the parameters of the decision problem but is not involved in the optimization loop (see Eq. 7).

Non-linear transitions can be mitigated by the use of a non-linear model in simulations (in Eq. 8), as shown by Theorem 3. We can still have an optimal solution. Therefore, we combine polynomial time decision making, and consistency.

Theorem 2 shows that with a limited number of cuts in the representation (Eq. 7) we have approximately optimal policies, outperforming classical MPC. Prior to any optimization, our algorithm is equal to a MPC with, possibly, a handcrafted valorization. It is therefore easy to use, on top of an existing MPC implementation; MPC might be the most efficient usual approach in such contexts, due to its handling of arbitrary complex stochastic processes; we indeed modify it by including a direct policy search layer, so that we preserve the polynomial decision making, while handling non-linear effects.

The main drawback is the heavy computation of θ^* as in Eq. 8. This however can be mitigated by warm starts and parallelization - as noisy optimization is highly parallel. At the cost of a few hours or days of offline optimization, DMPC brings a policy that does not require any linearity or convexity assumption and that is not restrained to a Markovian random process.

References

[1] Sandra Cecilia Astete-Morales, Jialin Liu, and Olivier Teytaud. Log-log Convergence for Noisy Optimization. In Pierrick Legrand, Marc-Michel Corsini, Jin-Kao Hao, Nicolas Monmarché, Evelyne Lutton, and Marc Schoenauer, editors, *Artificial Evolution - 11th International Conference, Evolution Artificielle, EA 2013, Bordeaux, France, October 21-23, 2013. Revised Selected Papers*, volume 8752 of *Lecture Notes in Computer Science*, pages 16 –28, Bordeaux, France, October 2014. Springer.

[2] Maria-Florina Balcan and Kilian Q. Weinberger, editors. *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2016.

[3] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[4] Dimitri P. Bertsekas. Dynamic programming and suboptimal control: A survey from ADP to MPC. *European Journal of Control*, 11(4-5):310–334, 2005.

[5] H.-G. Beyer. *The Theory of Evolution Strategies*. Natural Computing Series. Springer-Verlag Berlin Heidelberg, 2001.

[6] Jean-Joseph Christophe, Jérémie Decock, and Olivier Teytaud. Direct model predictive control. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, Bruges, Belgique, 2014.

[7] Rémi Coulom. CLOP: Confident Local Optimization for Noisy Black-Box Parameter Tuning. In H. Jaap van den Herik and Aske Plaat, editors, *Advances in Computer Games*, number 7168 in Lecture Notes in Computer Science, pages 146–157. Springer Berlin Heidelberg, November 2011. DOI: 10.1007/978-3-642-31866-5_13.

[8] Vaclav Fabian. Stochastic Approximation of Minima with Improved Asymptotic Speed. *Annals of Mathematical statistics*, 38:191–200, 1967.

[9] Antonio Frangioni. Solving nonlinear single-unit commitment problems with ramping constraints. *Operations Research*, 54(4):767–775, 2006.

[10] Verena Heidrich-Meisner and Christian Igel. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 401–408, New York, NY, USA, 2009. ACM.

[11] S. Najafi and Y. Pourjamal. A New Heuristic Algorithm for Unit Commitment Problem. *Energy Procedia*, 14:2005–2011, 2012. 2011 2nd International Conference on Advances in Energy Engineering (ICAEE).

[12] M. V. F. Pereira and L. M. V. G. Pinto. Multi-stage stochastic optimization applied to energy planning. *Math. Program.*, 52(2):359–375, October 1991.

[13] M. Schoenauer and E. Ronald. Neuro-genetic truck backer-upper controller. In *ICEC94*. IEEE Press, June 1994.

[14] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[15] Ohad Shamir. On the complexity of bandit and derivative-free stochastic convex optimization. In *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, pages 3–24, 2013.

[16] Alexander Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63–72, 2011.

[17] J.C. Spall. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE Transactions on Automatic Control*, 45(10):1839–1853, October 2000.

[18] J.C. Spall. Feedback and weighting mechanisms for improving jacobian estimates in the adaptive simultaneous perturbation algorithm. *IEEE Transactions on Automatic Control*, 54(6):1216–1229, June 2009.

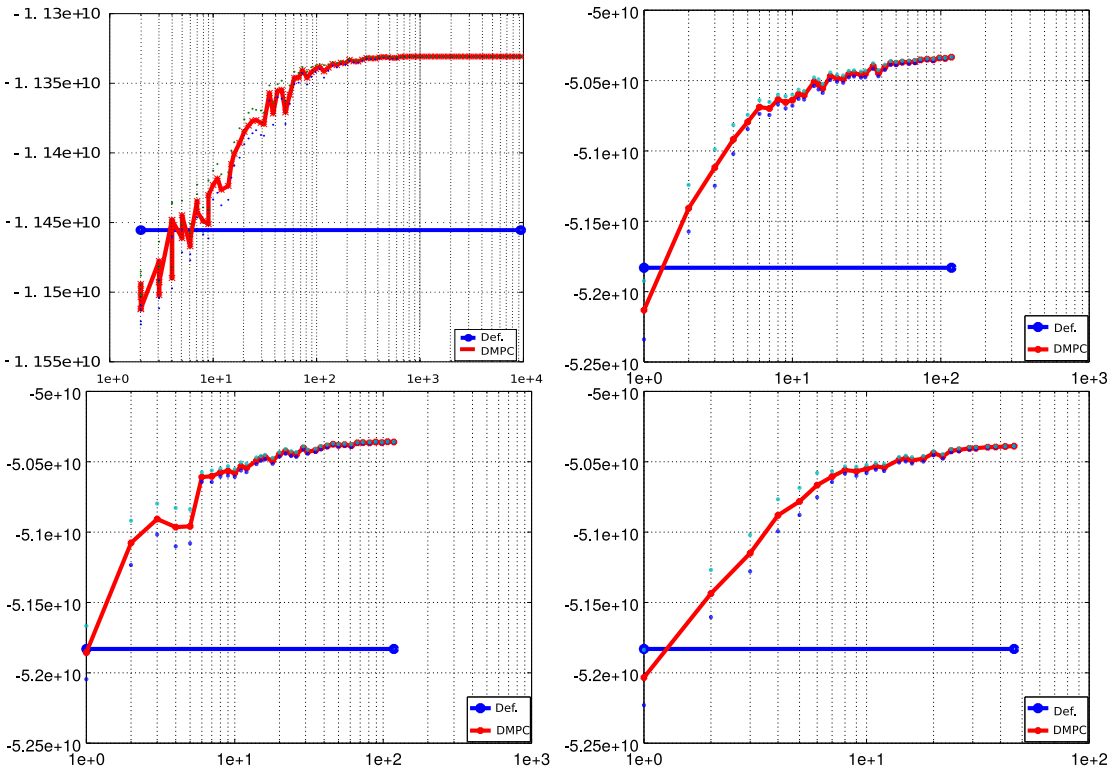

Figure 2. Experiments with 1 stock (top left), and 10 stocks (top right: including the LTM constraint designed offline (see Section IV-B2) by optimization (the higher the better); bottom left: including this constraint on the stocks, but relaxed by 20%; bottom right: constraint completely removed except for the last time step). The horizontal line is, in each case, the performance with the constraint designed offline. The optimization algorithm is a very simple random search, but the structure of the policy is carefully designed (see text). We see that our algorithm has the same result independently of the offline constraint. The X-axis is in all cases the number of evaluations in the optimization of θ. The default methods have no computation of θ; their offline computational costs are larger than time constants here and human expertise is involved in the loop.

[19] S. Virmani, E. C. Adrian, K. Imhof, and S. Mukherjee. Implementation of a Lagrangian relaxation based unit commitment problem. *IEEE Transactions on Power Systems*, 4(4):1373–1380, 1989.

[20] M. Zambelli, S. Soares Filho, A.E. Toscano, E. dos Santos, and D. da Silva Filho. NEWAVE versus ODIN: comparison of stochastic and deterministic models for the long term hydropower scheduling of the interconnected brazilian system. *Sba: Sociedade Brasileira de Automatica*, 22:598 – 609, December 2011.

APPENDIX

Lemma 1: There exist $d+1$ unit vectors $w_1, \ldots, w_{d+1}$ in $\mathbb{R}^d$, and there exists a constant $c > 0$ such that, for any unit vector $u \in \mathbb{R}^d$, there exist $i \in \{1, \ldots, d+1\}$ such that $(u|w_i) > c$.

Proof 4: $(x)_k$ is the k-th coordinate of vector x.

Consider $\mathbb{S} = \{x \in \mathbb{R}^{d+1} \mid \sum_{i=1}^{d+1}(x)_i^2 = d^2 + d \text{ and } \sum_{i=1}^{d+1}(x)_i = 0\}$. $\mathbb{S}$ is the intersection of a d-dimensional sphere of $\mathbb{R}^{d+1}$ and of a hyperplane in dimension $d+1$; it is therefore a $(d-1)$-dimensional sphere of a d-dimensional Euclidean space (which is the hyperplane $H = \{x; \sum_{i=1}^{d+1}(x)_i = 0\}$ of $\mathbb{R}^{d+1}$). We show the result in this d-dimensional Euclidean space.

Let us consider $w_1, \ldots, w_{d+1}$ defined by:

$$\forall i \in \{1, \ldots, d+1\}, \quad w_i \in \mathbb{R}^{d+1}$$

$$\text{and } \forall j \in \{1, \ldots, d+1\}, \quad (w_i)_j = \begin{cases} d & \text{if } j = i, \\ -1 & \text{otherwise.} \end{cases}$$

The w_i are elements of $\mathbb{S}$. They are a regular simplex. For any $x \in \mathbb{S}$, we define $v(x) = \max_{i \in \{1,\ldots,d+1\}} (x|w_i)$. Let us show that for any $x \in \mathbb{S}$, $v(x) > 0$.

Without loss of generality, let us assume that $(x)_1 > (x)_j$, for any $j \in \{2, \ldots, d+1\}$ (otherwise just permute coordinates, the problem is invariant by such permutations).

$$(x|w_1) = (d+1)(x)_1 - \underbrace{\sum_{i=1}^{d+1}(x)_i}_{0, \text{ because } \mathbb{S} \subset H},$$

so for any $x \in \mathbb{S}$, $v(x) = (d+1) \max_{i \in \{1,\ldots,d+1\}} (x)_i$. By definition of $\mathbb{S}$, $\max_{i \in \{1,\ldots,d+1\}} (x)_i > 0$, therefore $v(x) > 0$. Since $\mathbb{S}$ is a compact Hausdorff space, v reaches its lower bound on $\mathbb{S}$: there exists $c > 0$ such that $\forall x \in \mathbb{S}$, $v(x) \geq c$.

We have the above conclusion for the hyperplane $\sum_{i=1}^{d+1}(x)_i = 0$ of $\mathbb{R}^{d+1}$, thus we have the same conclusion for the domain $\mathbb{R}^d$. We have concluded the proof for x such that $\|x\|^2 = d^2 + d$; a fortiori the result holds for x such that $\|x\| = 1$.