Solar home 2020 : enrichissement du benchmark open source de gestion d'énergie avec entrées incertaines

Pierre Haessig, Jesse James Arthur Prince Agbodjan, Romain Bourdais, Hervé Guéguen

Solar home 2020 : enrichissement du benchmark open source de gestion d'énergie avec entrées incertaines

Pierre HAESSIG*, Jesse James PRINCE AGBODJAN*, Romain BOURDAIS*, Hervé GUÉGUEN*
*IETR, CentraleSupélec

RÉSUME – Nous présentons la version 2020 du banc de test Solar home. Cet outil open source permet de comparer de façon simple et objective différentes méthodes de gestion d'énergie en présence d'entrées incertaines. Ce cas test est basé sur les données réelles d'une maison solaire avec production photovoltaïque et stockage dont il faut piloter les flux d'énergie pour minimiser la facture énergétique. Par rapport à la version initiale de 2018, nous avons ajouté de nouvelles méthodes d'optimisation parmi les plus performantes : programmation dynamique stochastique et MPC (commande prédictive) stochastique du type OLFC. Nous discutons également de l'intérêt de la gestion par règles heuristiques optimisées, qui peut se révéler aussi performante que les méthodes "optimales". La conclusion sur le classement des méthodes est cependant remise en question par la sensibilité au scénario d'entrée qui reste largement à étudier.

Mots-clés – Gestion d'énergie, Optimisation dynamique, Commande prédictive, Programmation dynamique stochastique

1. INTRODUCTION

Une abondance de travaux de recherche porte sur le pilotage des systèmes énergétiques pour optimiser leur fonctionnement. Les nombreuses méthodes de gestion d'énergie existantes sont très variées et parfois complexes (prérequis mathématiques, moyens techniques de mise en œuvre…). Ainsi, la personne qui aborde un problème de gestion d'énergie est confrontée à un choix difficile, car il est multicritère et les méthodes possibles à comparer sont nombreuses. Parmi ces critères : quelles méthodes sont les plus performantes ? Mais aussi : quelles sont les plus faciles à prendre en main ? Et à implémenter (besoins logiciels) ? Y a-t-il beaucoup de paramètres à régler ?

Pour faciliter la *comparaison objective* des méthodes de gestion d'énergie, nous avons créé en 2018 le banc de test "Solar home" [1]. Ce cas test, open source, est basé sur les données réelles d'une maison solaire avec production photovoltaïque et stockage dont il faut piloter les flux d'énergie pour minimiser la facture énergétique (figure 1). Il s'agit d'un problème à la fois simple et concret de gestion d'énergie *en présence d'entrées incertaines* (incertitudes sur la production et la consommation futures).

Notre travail de 2018 consistait principalement dans la mise au point du cas test : formalisation, sélection des données et dimensionnement du système PV+stockage. La palette de méthodes de gestion d'énergie que nous avions implémentées et comparées était loin d'être complète : gestion heuristique et commande prédictive déterministe. Dans la version 2020 de Solar home, nous avons ajouté des méthodes plus complexes et a priori plus performantes dans un contexte stochastique, ce qui augmente nettement l'intérêt du banc.

Il existe des travaux comparant des méthodes de gestion d'énergie, sur des exemples réels complexes (barrages hydroélectriques [2], véhicules hybrides [3]). Les travaux récents de Pacaud *et al.* [4] sont même très proches de Solar home. Ce même groupe de recherche a d'ailleurs lancé benchmark open source EMSx [5] qui propose une évaluation de contrôleurs

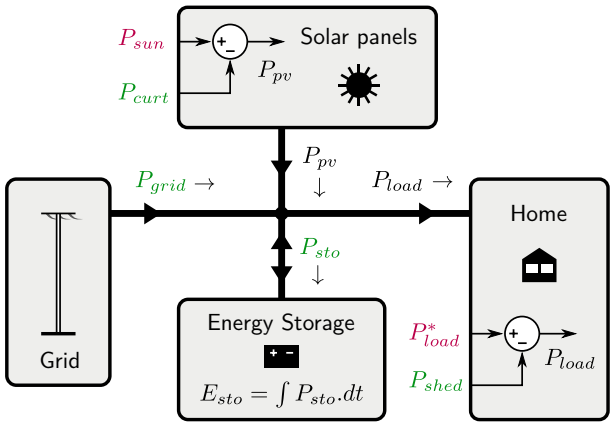

FIG. 1. Modèle en flux d'énergie de la maison solaire. Variables de décision en vert, données externes en rouge (potentiel solaire et consommation souhaitée), variables internes en noir. La consommation du foyer P^*_{load}, imposée, est couverte par 3 sources : le réseau électrique, des panneaux solaires (écrêtables) et un système de stockage. En dernier recours, la consommation de la maison peut être délestée, mais ce n'est pas nécessaire dans cet article.

d'énergie sur un jeu de données particulièrement riche de 70 sites de microréseaux industriels. Notre proposition est complémentaire, car elle porte au moins autant sur la comparaison *fonctionnelle* des méthodes (e.g. analyse des objets et concepts nécessaires pour leur mise en oeuvre) que sur la seule comparaison quantitative des résultats d'optimisation. Nous souhaitons également expliquer quelques "astuces pratiques" pour appliquer ces méthodes. En effet, dans des articles de gestion d'énergie portant sur une seule méthode, la formalisation du problème semble souvent naturelle et ce n'est qu'en formalisant un même problème dans plusieurs cadres différents que les difficultés et les choix subjectifs de modélisation apparaissent. Par ailleurs, comme EMSx, notre proposition est librement accessible à tous (code et données open source). Ainsi, nous espérons que Solar home a un intérêt pédagogique pour des personnes débutant dans le domaine de la gestion d'énergie en général ou sur une méthode d'optimisation particulière.

2. BANC DE TEST SOLAR HOME

Le banc de test est composé d'un modèle du problème et de données qui l'alimentent.

2.1. *Description du problème d'optimisation*

Nous résumons ici le problème d'optimisation qui est décrit plus pédagogiquement dans [1]. Le modèle de la maison solaire est basé sur des flux d'énergie (Fig. 1) avec pour objectif la mi-

nimisation de la facture énergétique cumulée sur n instants :

$$C_{grid} = \sum_{k=1}^{n} c_{grid}(k) P_{grid}(k) \tag{1}$$

Le problème est formulé à temps discret (temps k, pas de temps Δ_t) avec un prix de l'électricité c_{grid} variable, mais connu : heures creuses 0,10 €/kWh de 00h00 à 05h59, heures pleines 0,20 €/kWh de 06h00 à 23h59. La minimisation de la fonction coût (1) fait intervenir trois types de variables temporelles : des signaux extérieurs, des variables de décision et des variables d'état ou internes (cf. couleurs Fig 1).

Les signaux externes sont le productible solaire P_{sun} et la consommation souhaitée P^*_{load}. Les variables de décision (à optimiser) sont la consommation du réseau P_{grid}, l'écrêtage de la puissance solaire excédentaire P_{curt}, la puissance stockée dans la batterie P_{sto} et l'éventuel délestage de la charge P_{shed}. Dans le contexte présent où le réseau est suffisamment puissant (cf. §2.3) et jamais en panne, ce délestage ne sert pas et la puissance consommée est égale à la demande : $P_{load} = P^*_{load}$ (cf. [6] pour une extension du Solar home avec des pannes et la gestion optimale du délestage). Les variables de décisions sont contraintes par la puissance maximale de stockage P^{max}_{sto} et la puissance maximale du réseau P^{max}_{grid}. Enfin, l'écrêtage solaire doit être inférieur au productible. Ces contraintes s'expriment par :

$$-P^{max}_{sto} \leq P_{sto} \leq P^{max}_{sto} \tag{2a}$$

$$0 \leq P_{grid} \leq P^{max}_{grid} \tag{2b}$$

$$0 \leq P_{curt} \leq P_{sun} \tag{2c}$$

L'énergie stockée est une variable d'état, car elle est régie par une équation dynamique (couplage entre instants) :

$$E_{sto}(k+1) = E_{sto}(k) + P_{sto}(k)\Delta_t \tag{3}$$

et cette énergie est limitée par la capacité de stockage E_{rated} :

$$0 \leq E_{sto} \leq E_{rated} \tag{4}$$

La dynamique (3) permet de préciser le sens des variables à temps discret. Les variables de stock comme $E(k)$ représentent leur valeur *à l'instant* k. Par contre, les flux, i.e. toutes les puissances $P_{...}(k)$, représentent une *moyenne* entre les instants k et $k+1$ du signal continu sous-jacent. Cette définition est bien cohérente avec (3). Nous renvoyons à [7] pour l'ajout de pertes au modèle de stockage. Pour finir, les flux sont contraints par la contrainte de conservation de l'énergie :

$$P_{grid} + \underbrace{P_{sun} - P_{curt}}_{P_{pv}} = P_{load} + P_{sto} \tag{5}$$

Le problème de décision semble comprendre 3 variables, mais nous allons voir qu'il n'y a fondamentalement qu'un seul degré de liberté, grâce à des regroupements et à cause de la contrainte algébrique de conservation de l'énergie (5). Nous définissons la charge nette $P_{nl} \triangleq P^*_{load} - P_{sun}$ qui regroupe toutes les données externes et $P_{gc} \triangleq P_{grid} - P_{curt}$ qui résume deux décisions. Cette dernière agrégation est parfaitement inversible, car c'est la différence de deux variables positives qui sont complémentaires (on ne tire pas sur le réseau en même temps qu'on écrête le PV). Ainsi, on retrouve P_{grid} en prenant la partie positive de P_{gc} et P_{curt} avec sa partie négative (i.e. respectivement $\max(0, P_{gc})$ et $\max(0, -P_{gc})$). Avec ce regroupement, (5) se simplifie en :

$$P_{gc} = P_{nl} + P_{sto} \tag{6}$$

La figure 2 illustre l'agrégation et cette équation. Eq. (6) permet de voir P_{gc} comme une conséquence imposée de la puissance stockée P_{sto} qui est donc l'*unique degré de liberté* de ce problème. Le choix inverse (P_{sto} se déduit du choix de P_{gc}) est également possible et cet ordre n'a pas d'importance sauf lorsqu'on considère P_{nl} comme aléatoire et qu'on veut prendre une décision indépendante de sa valeur. (cf. OLFC §3.4).

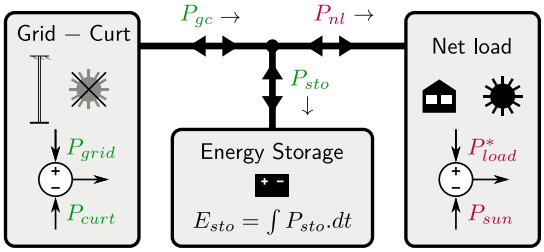

FIG. 2. Modèle en flux d'énergie de la maison solaire avec agrégation des variables qui montre le caractère unidimensionnel du problème décision.

2.2. *Non-anticipativité des décisions*

La description du problème Solar Home serait incomplète si nous ne précisions pas la "règle du jeu" concernant *l'information disponible* pour le contrôleur à chaque prise de décision. En effet, la performance d'une loi de gestion *en présence d'incertitude* dépend, en général, de la quantité d'information disponible. Pour Solar home, connaître les valeurs des signaux externes (i.e. la charge nette) postérieures à l'instant de décision donne un grand avantage alors que c'est irréaliste, car c'est une anticipation du futur. Nous imposons donc que la décision doive être *non anticipative*. Cependant, parce que le problème est formulé à temps discret, il existe deux variantes de la formulation non anticipative :

— Variante 1 : à l'instant de décision k, les entrées incertaines sont connues jusqu'à l'instant k *inclus*
— Variante 2 : à l'instant de décision k, les entrées incertaines sont connues jusqu'à l'instant k *exclus*

Comme les variables de puissance $P_{...}(k)$ représentent une moyenne entre les instants k et $k+1$, la variante 1 est une légère entorse à la non-anticipativité alors que la variante 2 est un peu pessimiste. Cependant, cette différence reste légère (d'autant plus que le pas de temps Δ_t est petit par rapport aux variations des signaux), car, dans tous les cas, les instants postérieurs à $k+1$ sont incertains. Nous utilisons la variante 1 (parfois appelée "wait-and-see" ou "hasard-décision) qui évite des problèmes d'infaisabilité, alors que [4, 8, 5] préfèrent la 2 (parfois appelée "here-and-now" ou "décision-hasard"). In fine, l'essentiel est que toutes les méthodes soient comparées identiquement.

2.3. *Données d'entrée et dimensionnement*

Les données d'entrées proviennent du jeu de données *réel et ouvert* fourni par Ausgrid, l'opérateur réseau de Sydney et sa région en Australie. Il contient 3 années de consommation et production, au pas demi-horaire (Δ_t = 0,5 h), de 300 clients résidentiels disposant de panneaux PV. Ratnam *et al.* [9] donnent plus de détails sur la création du jeu de données.

Pour le banc de test, nous avons sélectionné un client sans charge pilotable [1] et nous avons choisi 30 jours consécutifs de test, du 29/11/2011 au 28/12/2011 (nombres de points : $n = 24 \times 48$). Les premiers jours de test sont représentés figure 3. Avec ces données, un travail d'optimisation du dimensionnement a conduit à choisir une puissance crête photovoltaïque de 4 kW_c associée à une batterie de capacité $E_{rated} = 8$ kWh [1]. Sa puissance P^{max}_{sto} n'est pas limitée. Le réseau a une puissance maximale P^{max}_{grid} = 3 kW, suffisante pour couvrir la charge maximale sans batterie. Avec ce dimensionnement, la production solaire couvre 91% de la consommation sur la période de test (avec bien sûr une forte variabilité journalière) et une capacité de stockage d'environ la moitié de la consommation (ou de la production solaire) journalière.

Par ailleurs, la plupart des méthodes hormis les plus simples nécessitent une *phase d'apprentissage*. Il peut s'agir d'un réglage de paramètres (cf. heuristique optimisée §3.1) ou bien de

1. Client numéroté "12". La description détaillée de ce choix est fournie dans le fichier data/README.md du dépôt.

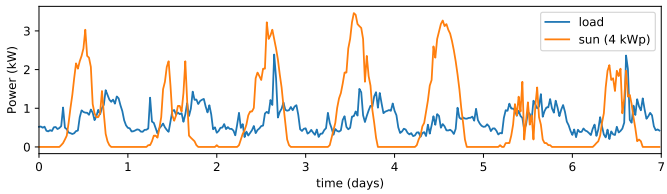

FIG. 3. Consommation et productible solaire durant les 7 premiers jours de test

construire des données statistiques : prévision, scénarios. Pour respecter la non-anticipativité, cet apprentissage doit se faire sur des données passées et nous avons donc réservé 30 jours du mois précédent.

2.4. *Formulation générique du problème d'optimisation dynamique*

Le problème de minimisation du coût (1) avec les contraintes décrites partie §2.1 est un problème *d'optimisation dynamique stochastique*. Il est dynamique par la présence de la batterie qui introduit un *couplage temporel* entre les valeurs successives de l'état d'énergie (3). Le caractère stochastique vient de la règle de non-anticipativité qui considère que la décision à chaque instant ne peut pas dépendre des valeurs futures des données externes.

Pour discuter plus efficacement des différentes méthodes de gestion d'énergie, nous introduisons ici une formulation générique de ce type de problème. Nous utilisons la notation "automaticienne" utilisée en particulier dans le livre de D. Bertsekas [10] avec des variables regroupées dans trois vecteurs : état x, décision u et données exogènes incertaines w. Pour le problème Solar Home, ces variables correspondent a priori respectivement à E_{sto}, P_{sto} et P_{nl}. Cependant, il apparaitra partie 3 que des choix de modélisation ainsi que la méthode de gestion d'énergie peuvent modifier ces associations. On se peut se référer à Powell ([11] ou [12]) pour un tour d'horizon des variantes de formulation de ce problème qui coexistent selon les communautés (Markov Decision Process, Multistage Stochastic Programming…).

La fonction coût à minimiser, généralisation de (1), est le cumul sur l'horizon du problème d'une fonction coût instantané c qui dépend de la valeur courante des variables x, u, w et éventuellement de l'instant k :

$$J = \mathbb{E}\left\{\sum_{k=1}^{n} c_k(x_k, u_k, w_k)\right\} \tag{7}$$

L'espérance est nécessaire pour donner une valeur non aléatoire à la somme lorsque l'incertitude $\{w_k\}_{k=1\dots n}$ est modélisée par un processus aléatoire (i.e. une séquence de variables aléatoires). Si, par contre, on considère que $\{w_k\}$ est une séquence de données, elle disparait.

La dynamique du système, ici celle de la batterie (3), est exprimée avec une fonction dynamique f :

$$x_{k+1} = f(x_k, u_k, w_k) \tag{8}$$

La décision à prendre à chaque instant doit respecter des contraintes qui peuvent dépendre de l'état :

$$u_k \in U(x) \tag{9}$$

Solar home rentre dans bien dans cette écriture, car en combinant (3) et (4), l'intervalle possible pour la décision $P_{sto}(k)$ dépend de l'état $E_{sto}(k)$. Enfin, cette décision doit être *non anticipative* : u_k peut dépendre de l'état présent x_k et des commandes et incertitudes passées, mais pas présentes ou futures. Cependant, nous avons signalé partie 2.1 que, dans Solar home, la décision peut dépendre de la charge nette présente. Pour formaliser ce cas dans notre cadre général, il faut ajouter l'entrée incertaine P_{nl} à l'état et décaler l'entrée comme expliqué §3.2.

Observons que pour le problème Solar home, les fonctions c et f sont *linéaires*. C'est un cas courant en gestion d'énergie et cette propriété est très importante pour l'efficacité de résolution informatique de beaucoup de méthodes de gestion (e.g. commande prédictive 3.3), mais pas toutes (e.g. SDP 3.2.1). On conserve cette efficacité dans le cas un plus général où c est *convexe* (e.g. quadratique) [7, 13].

2.5. *Forme de la solution optimale*

La *nature* de la solution du problème d'optimisation dynamique stochastique (e.g. la gestion d'énergie de Solar home) est très différente de celle des optimisations "classiques" (déterministe). Cette différence est parfois source de confusion. Alors qu'en optimisation classique on recherche des *nombres* (décision $u_1, \dots, u_n$), la présence de l'incertain oblige à chercher des *stratégies* (aussi appelées *loi de gestion*). La forme précise de ces stratégies dépend de la méthode de résolution du problème, mais les stratégies optimales sont souvent une fonction de l'état : $\mu : x \mapsto u$ ("retour d'état" en automatique).

Selon la méthode de gestion d'énergie, cette stratégie est une fonction *explicite*, comme avec les règles heuristiques ("rule based control") ou avec la programmation dynamique (§3.2) qui fournit une table de valeur de μ ("look-up table"), c'est-à-dire la décision $u_i = \mu(x_i)$ pour chaque valeur x_i appartenant à une grille de discrétisation de l'espace d'état.

Cependant, d'autres méthodes comme dans la commande prédictive (MPC §3.3, mais aussi OLFC §3.4) ne fournissent qu'une relation $\mu : x \mapsto u$ *implicite*. Ces méthodes utilisent généralement une optimisation *en ligne* qui à chaque instant k calcule un nombre (la décision u_k, souvent accompagnée de décisions prospectives $u_{k+1} \dots$ non utilisées), mais ce calcul dépend de l'état courant x_k : un tel contrôleur est donc bien une fonction de l'état.

3. MÉTHODES DE GESTION D'ÉNERGIE

Dans la version 2018 du banc de test Solar home, nous avions comparé deux méthodes de gestion d'énergie : une règle heuristique simple et la commande prédictive (MPC : Model Predictive Control) déterministe [1]. Pour la version 2020, nous avons ajouté plusieurs méthodes de gestion d'énergie parmi les plus performantes qui existent pour des problèmes d'optimisation dynamique et stochastique. Nous revenons également sur le MPC déterministe, car nous avons découvert une amélioration importante de sa formulation.

Dans cette partie, nous présentons la formalisation et des conseils de mise en oeuvre de ces méthodes. La comparaison des performances des différentes méthodes est faite §4.

3.1. *Gestion heuristique optimisée*

Dans notre travail de 2018 [1], nous avons présenté une loi de gestion heuristique simple qui permet de fournir une valeur de référence du coût de fonctionnement du système (coût que les autres méthodes cherchent à améliorer). Nous allons voir ici une amélioration substantielle de cette heuristique qui rentre dans la catégorie des "heuristiques optimisées". C'est sans doute l'approche la moins mathématique, ce qui ne l'empêche pas d'être parfois très efficace.

Rappelons d'abord que la loi heuristique simple pour Solar home consiste à utiliser prioritairement la batterie et n'utiliser le réseau ou l'écrêtage solaire qu'en dernier recours. Ainsi, cela signifie que, "tant que possible", batterie se charge avec le surplus solaire net (cas production > consommation) ou bien se décharge pour fournir la consommation nette (cas consommation > production). Formellement, cela peut s'écrire, avec les variables regroupées de la partie 2.1, de la façon suivante :

1. utilisation prioritaire de la batterie : $P_{sto} := -P_{nl}$ "tant

que possible", c'est-à-dire tant qu'elle n'atteint pas une de ses bornes de puissance (2) ou d'énergie (4). Si la batterie sature, on note P_{sto}^{sat} la plus grande puissance que la batterie peut fournir (positive ou négative selon le signe de P_{nl}).

2. utilisation de P_{gc} pour compenser le reste : $P_{gc} := P_{sto}^{sat} + P_{nl}$, c'est-à-dire que
   — on consomme sur le réseau ($P_{gc} > 0$) si la charge nette est positive et la batterie est vide ou en décharge maximale
   — on écrête la production solaire ($P_{gc} < 0$) dans le cas inverse

La performance de cette stratégie simple est donnée en première ligne du tableau 1. La performance (0.563 €/j) est correcte, mais d'autres méthodes font mieux.

3.1.1. *Présentation générale*

Pour améliorer la performance, l'approche "heuristique optimisée" combine deux idées :

1. une heuristique a priori : une intuition, en partie vague, des actions appropriées pour améliorer la performance. Le caractère vague s'exprime par la présence de *paramètres à régler* dans la formulation de la loi de gestion
2. une optimisation a posteriori : les paramètres sont réglés par optimisation d'un critère calculé en simulant la loi de gestion avec un réglage donné.

Trouver une heuristique pertinente est la problématique la plus ouverte de cette approche : où trouver les "bonnes astuces" ? L'expérience du problème de gestion d'énergie à traiter aide bien sûr, à condition d'avoir cette expérience. Une technique assez courante est de s'appuyer sur un formalisme du type *logique floue*, comme par exemple Caux *et al.* pour la gestion d'énergie d'un véhicule à hybride hydrogène+stockage [14]. Utiliser un contrôleur flou permet de fixer heuristiquement les grandes tendances de la gestion d'énergie, tout en faisant apparaitre des paramètres à optimiser (typiquement les bornes des fonctions d'appartenance).

Quelle que soit la méthode employée, il s'agit de construire une *loi de gestion heuristique paramétrique* que l'on peut écrire :

$$\mu_\theta : x, \theta \mapsto u \tag{10}$$

où x contient des variables utilisées pour la prise de décision et θ les paramètres à optimiser. Le problème d'optimisation consiste à minimiser le coût (7), où la décision u est issue de la stratégie μ_θ, c'est-à-dire :

$$\min J(\theta) = \mathbb{E}\left\{\sum_{k=1}^{n} c_k(x_k, \mu_\theta(x_k), w_k)\right\} \tag{11}$$

Pour exprimer l'optimisation des paramètres (11) l'approche la plus courante consiste à empaqueter une *simulation* du système avec la loi à régler dans une fonction coût "boite noire", c'est-à-dire un code de simulation $\theta \mapsto J$ Cela implique d'utiliser une méthode d'optimisation du type *boite noire* (sans gradient) et *globale* (qui peut échapper aux minima locaux). Les méthodes les plus connues sont les métaheuristiques évolutionnaires (algorithmes génétiques, essaims particulaires…) qui ont une part d'aléatoire dans leur recherche de l'optimum. Il existe également des approches déterministes comme DIRECT [15], disponible dans la bibliothèque d'optimisation open source NLopt [16]. Certains algorithmes sont spécialisés pour les fonctions coût coûteuses à évaluer (longue simulation) comme EGO [17] (approche par krigeage de la fonction coût, aussi appelée optimisation bayésienne).

Point méthodologique important, l'optimisation des paramètres de gestion (11) doit impérativement se faire avec des *données d'entrainement* différentes des *données de test* (§2.3) qui servent pour évaluer la performance finale du système (voir par exemple le paragraphe "4.3 Robustness…" de [14] et Figure 4). Ne pas faire cette séparation amène un risque de *surapprentissage* (overfitting), c'est-à-dire d'obtenir un réglage θ très adapté à un profil de mission particulier, mais peu adapté à d'autres.

En résumé, nous avons une méthode qui n'a pas de prérequis en optimisation mathématique (comparativement aux autres méthodes présentées dans cet article), mais qui nécessite une bonne expérience du système pour formuler des heuristiques pertinentes. Ce fait, couplé à l'utilisation de méthodes d'optimisation globale qui n'offrent généralement pas de garantie de convergence, implique que l'on n'est jamais certain d'avoir atteint la "meilleure valeur" du coût à minimiser (7). Cela n'empêche pas les heuristiques optimisées d'atteindre d'excellentes performances dans certains cas, sachant également que les méthodes théoriquement plus optimales incluent souvent des hypothèses simplificatrices qui limitent leur performance réelle (ex. : réduction de la dimension et discrétisation de l'état pour la programmation dynamique §3.2).

Dernier avantage notable : utiliser une heuristique paramétrique de la forme (10) est sans doute l'approche la plus simple pour faire une *cooptimisation* entre dimensionnement et gestion d'énergie du système. Il "suffit" de concaténer les paramètres θ de la loi de gestion aux paramètres de dimensionnement.

3.1.2. *Heuristique optimisée pour Solar home*

Nous avons amélioré la gestion heuristique simple en ajoutant une règle de charge en heure creuse qui amène la batterie à un état de charge E_{sp} à la fin des heures creuses :

$$P_{sto}(k) = \frac{E_{sp} - E_{sto}(k)}{t_{sp} - t(k)} \text{ si } -P_{sto} \leq P_{nl} \tag{12}$$

où t_{sp} correspond à l'instant de fin des heures creuses (6 :00 de la journée courante). Cette règle contient un paramètre à optimiser : E_{sp}. Notons que cette heuristique a été trouvée en analysant les résultats de la programmation dynamique stochastique (§3.2 et Figure 5).

L'optimisation du paramètre E_{sp} doit impérativement se faire avec des données d'entrainement (30 jours précédents les données de test) plutôt que celles de test, pour éviter l'overfitting. Cette optimisation est illustrée Figure 4. Le réglage optimal est $E_{sp}^* = 1.74$ kWh (rond bleu). Appliqué aux données de test, on obtient un coût de 0,512 €/j (diamant rouge et tableau 1). Cependant, si l'on optimise directement sur les données de test, on obtient $E_{sp}^* = 1.30$ kWh et un coût artificiellement bas de 0,508 €/j (rond vert).

La performance obtenue est meilleure que celle de la programmation dynamique (§3.2), mais nous ne savons pas si la différence est statistiquement significative (voir §4).

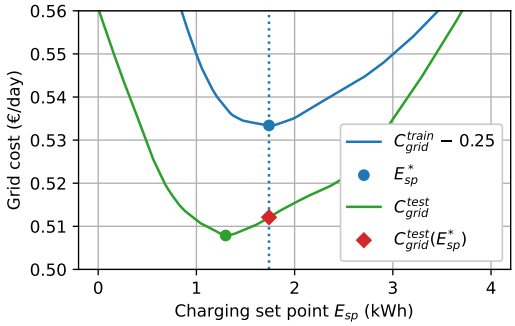

FIG. 4. Optimisation du paramètre de gestion E_{sp} (état de charge en fin d'heures creuses). Le réglage optimal est obtenu en minimisant le coût sur les données d'entrainement, puis il est appliqué aux données de test.

3.2. *Programmation dynamique stochastique (SDP)*

La programmation dynamique, dans sa version stochastique (abrégée ci-après par le sigle anglais SDP), est une méthode d'*analyse théorique* naturelle des problèmes d'optimisation dynamique stochastique du type présenté partie 2.4. Ce terme a été forgé par Bellman dans les années 1950 [18] et cette méthode donné naissance à beaucoup de variantes de formulation, de résolution et d'approximation. Nous relevons en particulier les Markov Decision Processes (MDPs) [19] et l'apprentissage par renforcement (RL en anglais) [20]. Pour comprendre les liens et les divergences entre ces approches, on peut se référer un article de Powell [11]. Une version spécialisée dite SDDP[21] a une certaine popularité en gestion d'énergie (ex. [4]) et nous en rediscutons rapidement plus bas. Ceci étant précisé, nous traitons ici de la SDP "pure", c'est-à-dire avec un minimum d'approximation, basée sur l'approche de Bertsekas [10]. Nous renvoyons à ce livre sur les méthodes d'analyse et de résolution (récursion de Bellman…, algorithme de "policy iteration") ainsi qu'à un précédent article [22] et la thèse [23, Ch 3] où nous présentons le package Python opensource `StoDynProg` qui implémente la méthode de résolution utilisée ici. Dans cet article, nous nous concentrons sur les enjeux de la mise en pratique.

Pour des problèmes où, comme dans Solar home, la durée n est longue, deux approches SDP sont applicables :

— Programmation dynamique à horizon fini : $n < \infty$

— Programmation dynamique à horizon infini : $n \to \infty$

L'approche à horizon fini est la première abordée dans le livre [10], car son traitement théorique est plus simple et elle semble naturelle vu que n est fini. Cependant, supposer que $n \to \infty$ est une approximation raisonnable pour la gestion d'énergie sur le temps long et cela permet de permet d'avoir une loi de gestion *stationnaire*, c'est-à-dire une loi $\mu : x \mapsto u$ optimisée dans un seul calcul hors ligne et valable à tous les instants. À l'inverse, l'approche à horizon fini nécessite, a priori, de calculer une loi de gestion pour chaque instant ($\mu_k : x_k \mapsto u_k$, pour $k = 1...n$, avec $n = 1440$ pour Solar home).

Pour obtenir une formulation à horizon infini, il n'est pas possible de simplement faire tendre $n \to \infty$, car le coût (7) divergerait. Une solution [2] retenue pour la SDP de Solar home est de minimiser un coût *moyen* ("average cost per stage") :

$$J_\infty = \lim_{n\to\infty} \mathbb{E}\left\{ \frac{1}{n} \sum_{k=1}^{n} c(x_k, u_k, w_k) \right\} \tag{13}$$

avec une fonction coût instantané c qui ne dépend *pas* de l'instant (à l'inverse de (7)). Des algorithmes dédiés à la minimisation de (13) existent ("*relative* value iteration" et "policy iteration") et sont implémentées par exemple dans notre package Python `StoDynProg` [22]. Ces algorithmes résolvent itérativement l'équation de Bellman pour le "problème à coût moyen sur un horizon infini"

$$J^* + \tilde{J}(x) = \min_{u\in U(x)} \mathbb{E}_w \Big\{ \underbrace{c(x,u,w)}_{\text{coût instantané}} + \underbrace{\tilde{J}(f(x,u,w))}_{\text{coût relatif du futur}} \Big\} \tag{14}$$

où J^* est le coût moyen optimal et $\tilde{J}(x)$ est le coût relatif du futur qui est une fonction de l'état (aussi appelée "valeur de Bellman"). La résolution génère J^*, $\tilde{J}(x)$ et surtout la loi de gestion optimale $\mu^* : x \mapsto u^*$ qui est l'argmin de (14).

3.2.1. *Applicabilité de la SDP*

La question de savoir si la SDP est applicable à un problème d'optimisation doit être séparée entre les aspects théorique et pratique (numérique).

Pour appliquer la SDP théoriquement, c'est-à-dire comme méthode d'analyse, le caractère additif du coût (7) est essentiel. Par contre la linéarité du coût c est de la dynamique f dans (8) ne sont pas nécessaires. Pour la SDP à horizon infini, ces deux fonctions ne doivent pas dépendre de l'instant (alors que l'horizon fini l'autorise). Enfin, et c'est singulier à la programmation dynamique, la SDP nécessite que (8) définisse un processus aléatoire qui respecte la *propriété de Markov*, à savoir que l'état futur x_{k+1} ne dépend que de x_k, u_k, w_k (sous-entendu pas des incertitudes passées). Pour cela, la série temporelle des incertitudes $\{w_k\}_{k=1...n}$ est généralement supposée être une séquence de variables *indépendantes et identiquement distribuées* (iid). Alors, de façon récursive, l'état x_k "résume" toute l'information sur les incertitudes et les décisions passées $\{(u_l, w_l), l = 1 \dots k-1\}$. Le fait que le processus w soit iid et que les fonctions c et f ne dépendent pas de l'instant fait que le problème est *stationnaire*, ce qui explique que la loi de gestion optimale $\mu*$ le soit aussi.

En plus de ces conditions théoriques, la SDP impose des contraintes pour pouvoir l'utiliser comme méthode de *résolution pratique*. La limite la plus connue est la "malédiction de la dimension" qui concerne la dimension du vecteur d'état x. La dimension maximale est faible, de l'ordre de 4–5, car le besoin en mémoire et en calcul croît exponentiellement avec elle. Seules des approximations (RL) ou des spécialisations (SDDP [3]) permettent d'aller au-delà. Par contre, comme la programmation dynamique est une *décomposition temporelle* du problème (via la "récursion de Bellman"), la durée n du problème n'est pas un facteur limitant. Dans le cas particulier de Solar home, la dimension de l'état est faible (seulement un stockage), ce qui permet de formuler un problème avec 3 variables d'état (voir les sous-parties qui suivent). La mise en pratique de la SDP pour Solar home nécessite deux travaux préalables : reformuler de façon stationnaire la périodicité et modéliser les entrées incertaines.

3.2.2. *Reformulation stationnaire de la périodicité journalière*

Le problème Solar home présente un *cycle jour/nuit*, présent à la fois dans les données d'entrées incertaines (la charge nette P_{nl}, cf. figure 3) ainsi que dans le prix de l'électricité connu $c_{grid}(k)$. Ces variations ne peuvent pas entrer, a priori, dans la formulation (13) qui est un problème stationnaire. Cette non-stationnarité serait intégrable avec l'approche à horizon fini, où c et f peuvent alors dépendre de l'instant, mais nous avons déjà souligné le défaut de cette approche qui génère une loi de gestion différente pour chaque instant du problème.

Lorsque cette non-stationnarité est périodique, comme dans Solar home, il est possible d'*ajouter une variable d'état* qui modélise le cycle temporel. Nous proposons la variable h qui représente l'heure du jour comme un entier entre 1 et 48 (ou de façon générale $24/\Delta_t$). Initialisée à 1, l'heure du jour a une dynamique périodique en "dent de scie" :

$$h_{k+1} = (h_k + 1) \,\%\, 48. \tag{15}$$

En ajoutant h à l'état x, on peut écrire le prix de l'électricité fonction de x pour aboutir à une fonction coût instantané c stationnaire. De plus, nous allons réutiliser h pour modéliser la périodicité de la charge nette. Au final, cette approche augmente le vecteur d'état, mais il semble intuitif que la gestion d'énergie d'un système avec panneau solaire dépende de l'heure du jour.

2. L'autre solution pour faire converger le coût, courante en RL, est d'ajouter un facteur d'amortissement γ^k au coût instantané, avec $0 < \gamma < 1$. Ce facteur a souvent une justification en finance (taux d'intérêt), mais pas en gestion d'énergie (ce qui n'interdit pas d'utiliser quand même un γ choisi empiriquement).

3. Stochastic Dual Dynamic Programming[21]. Notons que cette méthode de résolution spécialisée, basée sur l'approximation de la fonction de Bellman par des tangentes, augmente la dimension des problèmes traitables en échange d'une condition : la *convexité* du problème d'optimisation, ce qui implique un coût c convexe et surtout une dynamique f linéaire. Voir [24] pour une implémentation Julia open source.

3.2.3. *Modélisation de l'incertitude pour la SDP*

Vu les conditions d'applicabilité de la SDP (§3.2.1) la mise en œuvre de la SDP exige un travail préalable : *modéliser les incertitudes par un processus aléatoire*, alors qu'on ne dispose généralement au départ que de données historiques. Qui plus est, ce processus doit être markovien et faire apparaitre un processus w iid.

Ce travail relève des statistiques des séries temporelles [25], mais avec des contraintes spécifiques à la SDP qui oblige à une bonne concertation si l'on souhaite déléguer ce travail à des statisticien·ne·s. En effet, la modélisation des incertitudes, par exemple avec un processus AR (voir plus bas), augmente la dimension du vecteur d'état du problème, ce qui oblige à se restreindre à un modèle d'ordre faible (1 ou 2) là où la modélisation statistique optimale (au sens d'un critère d'information type AIC [25, §9.3]) pourrait sélectionner un ordre bien plus élevé.

Il est courant que les données incertaines d'un problème de gestion d'énergie, ici la charge nette P_{nl}, ne puissent être modélisées directement par le processus w qui doit être iid. Pour Solar home, deux aspects sont à prendre en compte dans la modélisation stochastique de la charge nette : *1)* sa périodicité journalière, *2)* sa persistance temporelle (~autocorrélation), c'est-à-dire que les variations successives autour du motif périodique ne sont pas statistiquement indépendantes.

L'approche de modélisation typique consiste à *augmenter le vecteur d'état* en y ajoutant le signal incertain. Ainsi, nous ajoutons au vecteur x_k la valeur de P_{nl} à l'instant k, puis nous formulons un processus stochastique f_{nl} qui constituera une des composantes de la fonction dynamique f (8) :

$$P_{nl}(k+1) = f_{nl}(P_{nl}(k), h_{k+1}, w_k) \tag{16}$$

Le processus générique (16) peut modéliser :
- le caractère partiellement incertain des valeurs futures de la charge nette grâce à la présence du bruit iid w_k
- la périodicité journalière grâce à la présence de l'heure du jour h_{k+1} (variable exogène).
- la persistance temporelle entre deux instants successifs grâce à la présence de $P_{nl}(k)$ (processus autorégressif)

Détail subtil : c'est la valeur de P_{nl} *à l'instant* k que nous avons ajouté dans l'état, car nous avons formulé le problème Solar home avec la variante 1 ("hasard-décision") de la contrainte de non-anticipativité (cf. §2.2). Cette variante autorise le contrôleur à utiliser la valeur de $P_{nl}(k)$. Comme la SDP produit une stratégie de décision qui dépend de l'état ($\mu_k : x_k \mapsto u_k$), notre choix d'augmentation de l'état est cohérent. Si nous options pour la variante 2, c'est la valeur de P_{nl} *à l'instant* $k-1$ qu'il faudrait ajouter dans l'état (et décaler l'équation (16) en conséquence).

En décrivant le processus (16), nous avons fixé une structure de dépendance, mais, à ce stade, $f_{nl}(...)$ signifie "une expression quelconque des variables ...". Il reste encore à préciser cette expression, c'est-à-dire "fitter" un processus concret aux données.

Si l'on s'en tient au seul phénomène de persistance (sans l'aspect périodique), on choisit typiquement un processus autorégressif linéaire, une structure nommée AR(p), où p indique l'ordre [25]. Le processus (16) devient ainsi un AR(1) :

$$P_{nl}(k+1) = \phi_1 P_{nl}(k) + w_k \tag{17}$$

où le coefficient $\phi_1 \in]-1, 1[$ règle l'autocorrélation à un pas (généralement positive). On se contente de l'ordre 1, car, comme signalé plus haut, il faut distinguer modélisation de l'incertain dans une visée SDP par rapport à une modélisation "statistique généraliste". Pour la SDP :
- on privilégie des modèles d'ordre faible à cause de la malédiction de la dimension
- la performance du contrôleur crée sur la base du modèle statistique prime sur la qualité statistique du modèle

Par exemple dans l'étude EMSx [5], le contrôleur SDP basé sur un AR(1) est presque aussi bon que celui basé sur un AR(2), pour un temps de calcul hors-ligne 10 fois plus court. Une entrée incertaine oscillante comme la houle nous semble un cas exceptionnel qui peut justifier un ordre 2 [22].

Pour la même raison de simplicité, bien que les processus AR appartiennent à la classe plus générale des ARMA cf. [25]), la modélisation à moyenne mobile (le MA de ARMA) nous semble à proscrire. En effet, la partie MA crée des variables d'état non mesurées qui augmentent considérablement la complexité de la programmation dynamique (cf. [10, ch 5]).

In fine, dans l'implémentation actuelle de la SDP pour Solar home, nous avons modélisé la périodicité uniquement (persistance négligée). Parmi l'éventail de choix possibles, voici un processus qui a l'avantage d'utiliser directement les données d'entrainement :

$$P_{nl}(k+1) = P_{nl,train}[w_k, h_{k+1}]. \tag{18}$$

Ce modèle suppose que la charge nette des 30 jours d'entrainements est rangée dans une matrice $P_{nl,train}$ de dimension 30×48, c'est-à-dire avec une journée par ligne. Les crochets $[i, j]$ notent l'indexation de cette matrice par les entiers i, j. L'incertitude w_k est ici un entier aléatoire uniforme entre 1 et 30 qui sélectionne le jour, alors que h_{k+1} sélectionne l'heure du jour adéquate de façon déterministe. Le processus (18) décrit une charge nette totalement incertaine à l'instant $k+1$ (pas de persistance) et qui prend ses valeurs dans l'ensemble des valeurs mesurées à la même heure durant les jours d'entrainement. Nous avons également essayé un modèle de structure similaire, mais gaussien :

$$P_{nl}(k+1) = \mu[h_{k+1}] + \sigma[h_{k+1}].w_k. \tag{19}$$

Ce 2e modèle utilise les vecteurs μ et σ qui contiennent respectivement la moyenne [4] et l'écart-type de la charge nette à chaque heure du jour dans les données d'entrainement. Le bruit w_k est normal (loi gaussienne centrée, de variance unitaire). Cela modélise une charge nette gaussienne dont la moyenne et l'écart-type correspondent statistiquement aux observations de l'ensemble d'entrainement.

Aucun des deux modèles utilisés ne modélise la persistance. Il est envisageable de créer des processus qui incluent la dépendance à $P_{nl}(k)$, mais la forme exacte reste à préciser. La résolution de la SDP ne sera pas plus calculatoire, mais il faudra vérifier que cela améliore bien la performance.

En conclusion générale de cette partie sur la modélisation de l'incertain pour la SDP, nous soulignons qu'on ne peut pas assimiler en général les entrées incertaines du problème (ici charge nette) et la variable incertaine w de la SDP. La relation entre ces variables est à construire au cas par cas dans une démarche de modélisation statistique pour aboutir à un processus stochastique, par exemple du type (16). Cette modélisation des entrées incertaines peut être une *part importante* du temps de mise en oeuvre de la SDP.

3.2.4. *Résolutions et résultats*

Après avoir reformulé le problème Solar home de façon stationnaire (§3.2.2) et modélisé l'incertitude (§3.2.3), le vecteur d'état du système pour la programmation dynamique contient 3 variables : $x = (E_{sto}, P_{nl}, h)'$. C'est une dimension suffisamment faible pour faire une résolution numérique directe, sans approximation autre que la discrétisation classique de l'espace d'état sur une grille. Nous ne détaillons pas la méthode de résolution, qui est faite grâce à notre package Python `StoDynProg` (cf. [22] et [23, Ch 3]). Pour une résolution avec une grille raisonnablement fine de $41 \times 51 \times 48 \approx 100$k points, il faut 6 minutes pour réaliser 5 itérations de l'algorithme Policy Iteration, dans lequel l'étape de Policy Evaluation a été faite sur 3 jours (3×48 pas). Les résultats qui suivent sont obtenus avec le

4. c'est le même vecteur des moyennes qui sert de prévision déterministe à l'approche MPC

modèle d'incertitude gaussien (19), mais l'autre modèle donne des résultats très proches.

La résolution SDP est une optimisation *hors ligne* qui génère la loi de gestion optimale $\mu^*(E_{sto}, P_{nl}, h) \mapsto u^*$ sous forme de table discrète (la décision optimale pour chaque point de grille de l'état). Pour l'utiliser en simulation, en particulier pour évaluer sa performance sur les données de test, il faut une routine d'interpolation. Nous utilisons pour cela la même routine d'interpolation linéaire qui est déjà utilisée dans l'algorithme Policy Iteration (pour interpoler la fonction valeur). La simulation est donc très rapide.

Pour analyser comment fonctionne la loi de gestion optimale SDP $\mu*$, nous superposons figure 5 la trajectoire de l'état d'énergie du stockage simulée sur les 30 jours de test. Cette représentation met en lumière un phénomène intéressant : de minuit à 6 :00 (heures creuses), la loi de gestion amène, via des trajectoires linéaires (c.-à-d. à puissance constante) la batterie vers un état d'énergie de 2,2 kWh à 6 :00. Ainsi, la baisse du coût de fonctionnement du système est due à une recharge optimisée pendant les heures creuses. La SDP, grâce à la prise en compte de l'incertitude, et en particulier du motif journalier, fournit un compromis optimal entre deux options extrêmes :

— ne pas charger la batterie pendant les heures creuses, pour pouvoir absorber au mieux une éventuelle abondance d'énergie solaire gratuite
— charger la batterie pendant les heures creuses pour éviter d'acheter de l'électricité plein tarif en cas de faible production solaire ou de forte consommation

Pour finir, cette observation nous a permis d'intuiter une loi de gestion heuristique optimisée (§3.1).

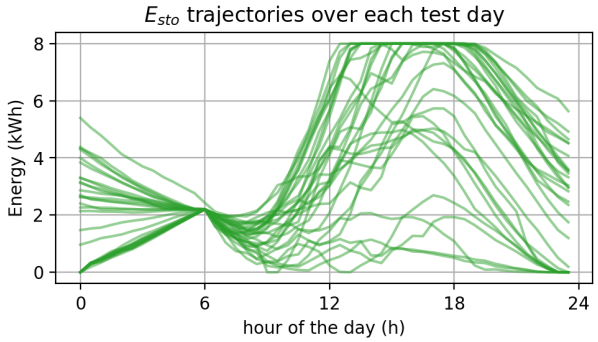

FIG. 5. Gestion d'énergie par Programmation dynamique stochastique. Superposition des trajectoires d'état d'énergie pour chacun des 30 jours de tests. Constat : de minuit à 6 :00 (heures creuses), la loi de gestion amène linéairement la batterie vers un état d'énergie de 2,2 kWh.

3.3. *MPC (commande prédictive) déterministe – le retour*

Notre travail de 2018 [1] incluait une mise en oeuvre de la commande prédictive (MPC, pour Model Predictive Control en anglais). Cette méthode est très utilisée académiquement et dans l'industrie pour la gestion d'énergie et la commande des systèmes en général. Cette méthode se base sur une optimisation en ligne qui, à chaque pas de temps, résout le problème d'optimisation sur un *horizon tronqué* pour en déduire une trajectoire optimale à suivre. La version la plus commune du MPC est dite *déterministe*, car le problème d'optimisation est rendu déterministe en remplaçant les entrées incertaines w par des prévisions $\hat{w}$ (cf. [1] pour la construction de la prévision). Ainsi, à chaque instant de décision ks, la commande u_k est celle qui minimise :

$$J_{MPC}(k) = \sum_{i=k}^{k+H} \tilde{c}(x_i, u_i, \hat{w}_i) \qquad (20)$$

Cette minimisation se fait sur toute la *séquence optimale* de décisions $(u_k, \dots, u_{k+H})$, mais seule la décision présente u_k est appliqué au système, car à l'instant de décision suivant, la commande est réoptimisée sur la période $k+1$ à $k+1+H$ (horizon glissant) avec les informations les plus à jour. Pour une bonne mise en pratique la commande prédictive, il est courant d'exiger que le la minimisation de (20) soit un problème *convexe*, car la convergence est alors assurée. Cela implique que toutes les contraintes du système (9) soient convexes (ex. : l'ensemble $U(x)$ est une boite, un polyèdre ou un ellipsoïde). En particulier, la dynamique du système (8) doit être *linéaire*, ce qui réduit considérablement le type de modèle de batterie utilisable. Dans certains cas, des modèles de stockage d'énergie plus complexes (non linéaires convexes) sont néanmoins possibles [7].

Par rapport à 2018, nous avons modifié notre implémentation pour corriger un défaut passé alors inaperçu. En effet, dans le cas d'un coût linéaire comme (1), la minimisation de (20) a souvent une infinité de solutions. Typiquement, des blocs d'énergie peuvent être échangés entre des heures où le prix est identique. Cependant, ce n'est pas parce que plusieurs trajectoires de commande $(u_k, \dots, u_{k+H})$ ont un coût *prospectif* (20) identique qu'elles ont coût en *boucle fermée* (7) identique. C'est pourquoi nous avons introduit dans (20) un coût instantané $\tilde{c}$ qui peut être différent du coût réel c. Cela permet d'introduire des *heuristiques* qui orientent la décision. Pour le MPC de Solar Home 2020, nous avons modifié le coût pour que le recours au réseau ou à l'écrêtage PV soit le plus tardif possible (à coût réseau identique). Pour cela, nous ajoutons au coût sur l'horizon une petite pénalisation des variables P_{grid} et P_{curt} :

$$\tilde{c}(x_i, u_i, w_i) = \underbrace{c_{grid}(i) P_{grid}(i)}_{c(x_i,u_i,w_i)} + \underbrace{d(i-k)\,(P_{grid}(i) + P_{curt}(i))}_{\text{pénalisation heuristique}} \qquad (21)$$

La fonction coût (21) exprime un problème d'optimisation *lexicographique* où la minimisation du coût économique est l'objectif prioritaire alors que la minimisation de la pénalisation heuristique est un objectif secondaire. La pénalisation heuristique se fait à l'aide de $\{d(n)\}$, une suite positive décroissante, pour repousser l'utilisation des variables pénalisée vers la fin de l'horizon. Nous avons choisi la formule linéaire suivante :

$$d(n) = \varepsilon \times \frac{H-1-n}{H-1}, \quad n = 0 \dots H-1 \qquad (22)$$

avec ε une constante positive et petite devant le prix de l'électricité pour que la pénalisation ne différencie que des solutions dont le coût réel est identique. La valeur $\varepsilon = 10^{-4}$ a été choisie en ce sens.

L'ajout de ce petit terme a un effet important sur la performance, puisque le MPC 2020 est nettement meilleur que le MPC 2018. En effet, à cause du minimum non unique, le résultat de 2018 dépendait implicitement du choix arbitraire d'un minimiseur u^* que trouvait le solver utilisé (`linprog` de Matlab). L'effet est encore plus spectaculaire si on inverse l'heuristique (séquence croissante $d' = 1-d$) en privilégiant le recours immédiat à réseau ou à l'écrêtage : la performance est catastrophique (de 0,6 à 1,5 €/j selon la longueur de l'horizon).

Nous n'avons pas vu dans la littérature la description d'une modification de la fonction coût telle que (21) dans le contexte du MPC pour la gestion d'énergie. Cependant, l'idée générale d'ajuster heuristiquement le coût est présentée comme classique chez Powell [11, 12] (appliqué pour corriger un "coût myope", c'est-à-dire un MPC avec un horizon de $H = 1$). Pour conclure cette partie, relevons que la promesse du MPC, comparativement aux approches heuristiques §3.1, est de pouvoir résoudre un problème de gestion d'énergie grâce à un bon modèle, mais sans nécessiter une connaissance heuristique sur son pilotage. La nécessité de la pénalisation heuristique du coût tronqué (21) vient modérer cette promesse.

3.4. *MPC (commande prédictive) stochastique*

Le MPC déterministe §3.3 nécessite une prévision $\hat{w}$ qui est souvent entachée d'erreur [5]. La stratégie MPC déterministe est certes en partie robuste à ces erreurs grâce au principe de l'horizon glissant, mais le problème d'optimisation (20) ne prend pas en compte *intrinsèquement* le caractère incertain de $\boldsymbol{w}$ (dans cette partie, nous utilisons le gras pour noter les variables aléatoires, car la distinction aléatoire vs déterministe sera importante).

Pour prendre en compte l'incertain, il existe une alternative à la programmation dynamique stochastique (§3.2). Il est possible d'étendre le MPC en s'appuyant sur les outils de la programmation stochastique (SP en anglais) [26, 27] pour forger un *MPC stochastique*. Le terme MPC stochastique recouvre plusieurs variantes, avec en commun le fait d'avoir une fonction coût tronquée sur un horizon H, avec en entrée un processus aléatoire $\boldsymbol{w_k}$ ou bien des réalisations de processus, généralement appelées *scénarios*. Ce qui peut apparaitre comme une idée simple pose en fait une série de questions pour aboutir à un problème bien posé :

1. Q1 : Quel choix de structure de décision ? En particulier, quelle information sur l'incertitude est disponible pour prendre les décisions ?
2. Q2 : Comment optimiser un coût qui est une variable aléatoire ?
3. Q3 : Comment intégrer les contraintes qui font intervenir des variables aléatoires ?

Q1 : La première question peut sembler abstraite, mais elle est essentielle. Il s'agit de décrire mathématiquement la non-anticipativité des décisions : à l'instant de décision k, la commande u_k peut dépendre des incertitudes passées $\boldsymbol{w_l}$, $l \leq k$, mais pas futures ($l > k$), cf. §2.2. Plus précisément dans un contexte de commande prédictive, il s'agit d'*extrapoler* cette contrainte sur l'horizon de prédiction et plusieurs choix sont possibles. Nous allons voir que ce choix aboutit à un compromis entre optimalité et complexité du problème d'optimisation à résoudre.

L'extrapolation exacte de la règle de non-anticipativité donne la structure de décision suivante : à l'instant de décision sur l'horizon $i = k + h$ ($h = 0...H$), u_i peut dépendre des $\boldsymbol{w_j}$, $j \leq k + h$, mais pas $j > k + h$. Si elle a l'avantage d'être exacte, cette extrapolation présente un défaut majeur : sa potentielle *complexité calculatoire*. En effet, l'expression "u_i dépend des $\boldsymbol{w_j}$, pour $j \leq i$" signifie que la variable d'optimisation n'est plus un scalaire, mais une fonction des incertitudes : $u_i = \mu(\boldsymbol{w_k}...\boldsymbol{w_i})$. Cette formulation s'appelle "multistage stochastic program" (MSP). Dans sa déclinaison classique par *scénarios*, où les incertitudes sont supposées *discrètes*, cela conduit à considérer plusieurs valeurs de la commande u à l'instant i. Comme la commande dépend de toute la séquence des incertitudes passées, l'ensemble des commandes à optimiser $(u_k...u_{k+H})$ est un *arbre de décision* comme représenté figure 6 [28]. Dans dans le cas du problème MPC déterministe (20), la séquence de commande est un simple vecteur. La taille de l'arbre croit *exponentiellement* avec la longueur de l'horizon H. Par exemple, si l'incertitude $\boldsymbol{w_j}$ ne peut prendre à chaque instant que 2 valeurs, alors au dernier instant de l'horizon il y a $S = 2^{H-1}$ trajectoires de réalisation des incertitudes et donc la commande du dernier instant u_H est un vecteur de S éléments à optimiser.

Pour contenir la taille du problème d'optimisation, une solution possible est l'approche "open-loop feedback control" (OLFC) mise en avant par Bertsekas [10, §6.2]. L'application à la gestion d'énergie a été faite par *Rigaut et al.* [8]. Il s'agit d'utiliser une structure de décision plus simple : la séquence de commande sur l'horizon $(u_k, ..., u_{k+H})$ est supposée ne dépendre d'*aucune* des incertitudes $\boldsymbol{w_j}$, $j = k...k + H$. La séquence de commande est alors un vecteur, identique au MPC déterministe, plutôt qu'un arbre. C'est une structure de décision plus restrictive que la réalité, car elle suppose qu'à l'instant de décision k la séquence de commande est fixe (déterministe, non aléatoire), c'est-à-dire incapable de réagir aux aléas, alors que l'horizon glissant du MPC permet justement, à l'instant de décision suivant $k + 1$, de changer la valeur de la commande u_{k+1} et suivantes. C'est le sens de l'appellation OLFC : la stratégie de contrôle est réellement boucle fermée ("feedback") grâce à l'horizon glissant, mais le contrôleur résout à chaque instant un problème d'optimisation "open loop".

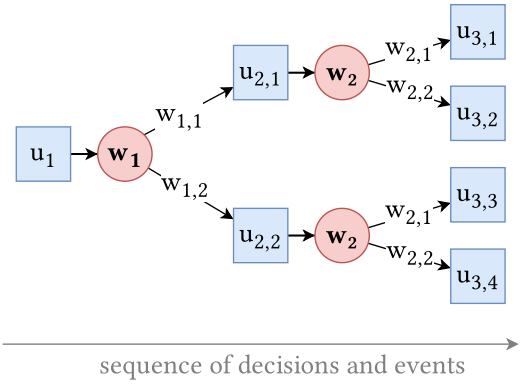

FIG. 6. Décisions u_i définies sur l'arbre de scénario généré par les séquences de réalisation des incertitudes $\boldsymbol{w_j}$. Le 1er indice correspond aux instants de l'horizon et le 2e à la numérotation des séquences de réalisation d'incertitude.

Q2 : Un coût aléatoire ne peut pas être directement minimisé, car il est impossible de comparer sans ambiguïté les variables aléatoires entre elles. Mathématiquement, il n'existe pas de relations d'ordre totales, seulement partielles. L'optimisation en présence d'aléas nécessite d'exprimer un choix subjectif d'*appétence au risque* [26, §2.5][27, §4]. Concrètement, le coût aléatoire doit être transformé en coût déterministe via un opérateur statistique comme l'espérance ou un quantile (aussi appelé Value-at-Risk en finance). Ce choix dépend de l'appétence au risque. Pour Solar home, nous utilisons un coût en espérance (7), c'est-à-dire un opérateur "risk-neutral" (les aléas positifs et négatifs se compensent), car nous supposons que la durée du problème n est suffisamment grande pour que la loi des grands nombres fasse que les aléas se compensent effectivement.

Q3 : Les contraintes avec des variables aléatoires peuvent également poser problème, mais cela dépend de la structure de décision choisie (Q1). Pour simplifier la discussion, nous prenons dans ce paragraphe l'exemple d'une contrainte simple $u \leq \boldsymbol{w}$, sachant que la contrainte qui pose véritablement problème dans Solar home est l'équilibre offre-demande (5) ou (6).

Si la structure de décision autorise la commande u à dépendre de l'incertain $\boldsymbol{w}$ (par ex. l'approche MSP avec les arbres de scénarios), alors il est possible de transformer la contrainte stochastique en un ensemble de contraintes déterministes. Pour chacune des M réalisations (discrètes) de l'incertain w_m ($m = 1...M$), la valeur correspondante de la variable de décision est contrainte par cette réalisation : $u_m \leq w_m$. La prise en compte de la contrainte stochastique est donc simple sur le principe, mais si la dimension du problème d'optimisation augmente.

Par contre, si la décision ne peut pas dépendre de l'incertain (par ex. avec l'OLFC), se pose alors la question du risque : comme la valeur de u doit être unique, car indépendante de $\boldsymbol{w}$, peut-on tolérer des non-respects ponctuels, par exemple pour des valeurs extrêmes de l'aléa ?

5. Certains travaux académiques supposent malgré tout une prévision parfaite, c'est-à-dire une anticipation des données futures qui n'est pas implémentable. La performance réelle peut être très décevante, cf. tableau 1.

Si l'on ne tolère aucun non-respect, on obtient une formulation *robuste* : $u \leq w_m \, \forall m$. Cette formulation est souvent *conservative*, car elle prend en compte le pire cas. Plus grave, dans le cas d'une contrainte égalité comme l'équilibre offre-demande (5), cela revient à chercher $u_i = w_m \, \forall m$, ce qui n'a aucune solution si les w_m sont différents !

Ainsi, on considère souvent qu'une contrainte stochastique peut-être partiellement relaxée. Certains travaux utilisent des *contraintes en probabilité* ("chance constraints") qui autorise une fraction de non-respect, par ex. $\mathbb{P}\{u \leq \boldsymbol{w}\} \geq 0,99$. L'alternative qui nous semble plus adaptée à la gestion d'énergie consiste à tolérer un dépassement par une relaxation de contrainte, puis à *pénaliser ce dépassement* : on introduit $\boldsymbol{y} \geq 0$, avec la contrainte $\boldsymbol{y} \geq u - \boldsymbol{w}$ et on pénalise $\mathbb{E}\{\boldsymbol{y}\}$. Dans le vocabulaire de la programmation stochastique, cette formulation s'appelle un problème "two-stage". La décision u est la décision "first-stage" et la nouvelle variable $\boldsymbol{y}$ est une décision "second-stage", aussi appelée "recourse". Cela signifie que u est une variable déterministe, car elle est choisie sans connaitre la valeur de l'incertitude $\boldsymbol{w}$. À l'inverse, le recours $\boldsymbol{y}$ est choisi en réaction à $\boldsymbol{w}$.

Selon les cas d'étude, la variable de recours $\boldsymbol{y}$ sera soit une variable physique du problème soit une variable de relaxation artificielle. Si c'est une variable physique, il faut s'assurer qu'il raisonnable de supposer que sa valeur peut être ajustée tardivement. La pénalisation de $\mathbb{E}\{\boldsymbol{y}\}$ n'est alors pas nécessaire, car cette variable est sans doute déjà pénalisée dans la fonction coût (7). Si par contre le recours est artificiel, il faut encore trouver empiriquement une bonne pondération de sa pénalisation.

3.4.1. *OLFC appliqué à Solar home*

Pour Solar home 2020, nous avons implémenté un MPC stochastique de type OLFC. Nous n'avons pas encore implémenté l'approche alternative MSP. Théoriquement plus optimale, elle est cependant plus calculatoire et plus complexe à implémenter, car il faut modéliser les incertitudes sous forme d'*arbres* de scénarios (fig 6) et formuler les variables de décisions sur ce même arbre [28].

Comme décrit plus haut, l'OLFC a l'avantage d'une complexité guère supérieur au MPC déterministe, mais il théoriquement sous-optimal. Autre avantage pratique, l'OLFC se formule presque comme un MPC déterministe, mais où l'unique signal de prévision d'incertitude $\{\hat{w}_i\}_{i=k\ldots k+H}$ est remplacé par un ensemble de S trajectoires $\{w_i^s\}_{i=k\ldots k+H}^{s=1\ldots S}$. La fonction coût à optimiser à chaque instant de décision par le contrôleur OLFC est une variante de celle du MPC (20) où l'on ajoute le calcul d'espérance, un incertain aléatoire, ainsi que les variables de recours :

$$J_{OLFC}(k) = \mathbb{E}\left\{ \sum_{i=k}^{k+H} \tilde{c}(x_i, u_i, \boldsymbol{y_i}, \boldsymbol{w_i}) \right\} \tag{23}$$

En pratique, on considère S trajectoires d'incertitude $\{w_i^s\}_{i=k\ldots k+H}^{s=1\ldots S}$ où chaque trajectoire a une probabilité π^s. Le coût s'exprime alors par une expression dite "déterministe équivalente" :

$$J_{OLFC}(k) = \sum_{s=1}^{S} \pi^s \sum_{i=k}^{k+H} c(x_i, u_i, y_i^s, w_i^s) \tag{24}$$

où y_i^s correspond à la valeur du recours $\boldsymbol{y_i}$ pour la trajectoire d'incertitude s. Cette expression a la même complexité mathématique que celle MPC. Ainsi, elle peut être traitée par le même solver d'optimisation linéaire. L'expression a une taille S fois plus grande.

L'incertitude considérée est la charge nette P_{nl}, modélisée par $S = 30$ trajectoires tirées des 30 jours de la période d'entrainement (§2.3). Par principe de l'OLFC, presque toutes les variables de décisions sont modélisées par une unique séquence déterministe $\{u_i\}_{i=k\ldots k+H}$, comme pour le MPC déterministe. Cependant, la théorie de Bertsekas [10, §6.2] ne fait pas mention des contraintes telles que l'équilibre offre-demande (6). Il faut donc choisir quelles variables sont fixes et lesquelles sont des recours.

Choix du recours Dans l'équation (6) où P_{nl} est incertain, nous supposons que P_{sto} est déterministe. Alors P_{gc} est nécessairement la variable de recours. Il semble en effet physiquement raisonnable de pouvoir ajuster la consommation du réseau ou l'écrêtage solaire au dernier moment, fonction de la charge nette. Ainsi, nous considérons 30 trajectoires possibles de P_{gc}, fonction de la charge nette correspondante.

Un choix inverse est de supposer P_{gc} déterministe et P_{sto} dépendant des scénarios d'incertitude. Cela se propage dans la dynamique de la batterie, et on doit alors prendre en compte S trajectoires de l'état d'énergie. Il semble que Rigaut [8] ait fait ce choix alternatif. Nous n'avons pas encore de test comparatif entre les deux choix possibles.

Notre implémentation du contrôleur OLFC prend 150 ms pour résoudre à chaque instant le problème d'optimisation (24) avec un horizon de 24 h ($H = 48$) et $S = 30$ scénarios. C'est à comparer aux 2 ms nécessaires pour résoudre le problème déterministe (20). L'augmentation du temps de calcul est donc empiriquement de l'ordre de $2S$, ce qui est raisonnable. La performance du contrôleur est bonne, mais comme expliqué ci-après, nous ne sommes pas en mesure de conclure précisément.

4. COMPARAISON ET SENSIBILITÉ DES RÉSULTATS

Le tableau 1 contient les principaux indices de performance des méthodes de gestions d'énergie présentées dans cet article, plus certaines présentées en 2018 (heuristique, optimisation anticipative) [1]. Le principal critère est le coût minimisé (1), mais nous l'accompagnons par deux indices énergétiques partiellement corrélés : la consommation d'énergie réseau $\int P_{grid}$ et l'énergie solaire gâchée $\int P_{curt}$. Toutes ces intégrales sont calculées sur les 30 jours de test et exprimées en moyenne par jour.

Une partie du classement correspond à celui de 2018, avec toujours aux deux extrémités l'heuristique simple comme limite haute (0,563 €/j) et l'optimisation anticipative (connaissant la consommation future) comme limite basse (0,354 €/j), très en deçà de toutes les autres méthodes qui sont non anticipatives.

Les nouvelles méthodes présentées (heuristique optimisée, programmation dynamique et OLFC), qui tiennent compte de l'incertain, sont très proches, entre 0,51 et 0,52 €/j. Ces résultats confirment nos résultats de 2018 : le caractère incertain de la charge nette *augmente nettement le coût de fonctionnement du système*. Dit autrement, il y a une grande valeur à disposer d'un mécanisme de prévision plus puissant que de simples statistiques (par ex. avec un modèle météo pour la production solaire). En utilisant le vocabulaire de la programmation stochastique, l'"Expected Value of Perfect Information" (EVPI) [26, §1.1 et §4] pour Solar home est assez grande, de l'ordre de $0,51 - 0,354 \approx 0,16$ €/j.

Plus surprenant, le MPC corrigé en 2020 rejoint le groupe des meilleurs et se place même en dessous. Cela contredit l'intuition : la meilleure méthode devrait être la programmation dynamique (et l'heuristique optimisée qui s'en inspire). Le MPC, qui prend théoriquement assez mal en compte le caractère incertain des données d'entrée était attendu un peu au-dessus, mais ce n'est plus le cas avec la correction 2020.

Ce résultat contre-intuitif pose la question de la sensibilité de ces résultats au scénario d'entrée, à savoir les 30 jours de test. En effet, on peut considérer que la trajectoire de la consommation et la production solaire sont la réalisation particulière d'un processus aléatoire inconnu. Dès lors, les indices de performance issus du test Solar Home deviennent des variables aléatoires et les nombres du tableau 1 n'en sont qu'une réalisation, entourée d'incertitude.

TABLEAU 1. Performance des différentes méthodes de gestion d'énergie, par coût décroissant (attention à la sensibilité de ces résultats au scénario d'entrée)

Méthode	Moyennes journalières (kWh/j, €/j)		
	P_{curt}	P_{grid}	$c_{grid}.P_{grid}$
Heuristique	1.94	3.38	0.563
MPC 24 h 2018	2.94	4.38	0.543
Prog. dyn. sto.	2.61	4.04	0.523
OLFC 24 h	1.97	3.41	0.522
Heurist. optim.	2.35	3.79	0.512
MPC 24 h 2020	2.14	3.58	0.509
Optim. anticip.	1.97	3.38	0.354

Une façon d'évaluer cette incertitude, sans chercher à modéliser les séries temporelles par un processus particulier, est d'utiliser la technique du *bootstrap* [29], en prenant garde au fait que les données sont temporelles, avec une forte saisonnalité journalière et une corrélation potentielle entre les jours. Cela requiert de faire du *seasonal block bootstrap* [30], c'est-à-dire de synthétiser des séries temporelles en concaténant des blocs piochés aléatoirement dans la série originale, mais en respectant une taille de bloc multiple de 24 heures.

De résultats préliminaires nous ont permis de voir que l'incertitude est assez forte (cf. notebook Benchmark_Sensitivity.ipynb). En particulier, la resimulation de la gestion heuristique sur 400 scénarios bootstrappés donne un intervalle de confiance à 95% du coût réseau de [0.35, 0.74] €/j et l'énergie consommée du réseau de [2.4, 4.4] kWh/j. Ces intervalles sont nettement plus larges que les variations dues à la méthode dans le tableau 1. Cependant, on ne peut pas conclure directement que ces différences sont toutes statistiquement non significatives. En effet, il est très probable que les performances des méthodes soient très corrélées entre elles, c'est-à-dire qu'un scénario qui est cher pour la gestion heuristique est sans doute également cher pour les autres. Le travail de resimulation de chaque loi de gestion pour ces quelques centaines de scénarios aléatoires reste à faire pour conclure réellement.

Si l'on prend l'exemple de Pacaud *et al.* [4], 1000 scénarios ont été utilisés pour montrer que la programmation dynamique est meilleure que le MPC *la plupart du temps* et *en moyenne*, mais le MPC était meilleur dans 7% des scénarios. Il est possible que l'étonnante inversion visible dans le tableau 1 relève de ce type de variabilité.

Au final, les résultats de cette étude de sensibilité nous pousseront peut-être à allonger la longueur du test Solar home. L'inconvénient est qu'au-delà de 60 jours (données d'entrainement + test), l'hypothèse que les données sont statistiquement stationnaires ne tient plus vraiment (changement de saison) ce qui amènerait à devoir faire des lois de gestion adaptatives.

5. CONCLUSIONS

Le banc de test Solar home a été enrichi de plusieurs méthodes de gestion d'énergie performantes pour renforcer sa couverture des méthodes existantes. Il reste cependant encore du travail sur le MPC stochastique basé sur des arbres de scénarios [28]. Les méthodes d'apprentissage par renforcement [20], pour lesquelles nous sommes moins familiers, sont également manquantes.

Comme décrit partie 4, l'analyse de sensibilité des résultats au scénario d'entrée, basée sur un bootstrap de plusieurs centaines de scénarios de synthèse, reste à faire. Enfin, nous souhaitons également utiliser ces variations de scénarios pour conduire des expériences numériques montrant l'importance de bien séparer les données d'entrainement (optimisation de la loi de gestion) et les données de test (évaluation de la performance de la loi de gestion) pour éviter le phénomène de surapprentissage (overfitting) qui est connu en machine learning, mais pas forcément en gestion d'énergie.

6. RÉFÉRENCES

[1] P. Haessig, J. J. Prince Agbodjan, R. Bourdais, and H. Guéguen, "Gestion d'énergie avec entrées incertaines : quel algorithme choisir ? Benchmark open source sur une maison solaire," in *SGE 2018, Nancy, France*, Jul. 2018.

[2] 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 Revista Controle & Automação*, vol. 22, pp. 598–609, Dec. 2011.

[3] Q. Jiang, F. Ossart, and C. Marchand, "Comparative Study of Real-Time HEV Energy Management Strategies," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10 875–10 888, Dec 2017.

[4] F. Pacaud, P. Carpentier, J.-P. Chancelier, and M. D. Lara, "Stochastic optimal control of a domestic microgrid equipped with solar panel and battery," Jan. 2018, working paper hal-01688666.

[5] A. Le Franc, P. Carpentier, J.-P. Chancelier, and M. De Lara, "EMSx : an Energy Management System numerical benchmark," May 2020. [Online]. Available : https://hal.archives-ouvertes.fr/hal-02425913

[6] J.-J. Prince Agbodjan, P. Haessig, R. Bourdais, and H. Guéguen, "Stochastic modelled grid outage effect on home Energy Management," in *2020 IEEE Conference on Control Technology and Applications (CCTA)*, Aug 2020, pp. 1080–1085.

[7] P. Haessig, "Convex Storage Loss Modeling for Optimal Energy Management," in *IEEE PowerTech 2021, Madrid*, 2021. [Online]. Available : https://hal.archives-ouvertes.fr/hal-03032241

[8] T. Rigaut, P. Carpentier, J.-P. Chancelier, M. D. Lara, and J. Waeytens, "Stochastic Optimization of Braking Energy Storage and Ventilation in a Subway Station," *IEEE Transactions on Power Systems*, vol. 34, no. 2, pp. 1256–1263, March 2019.

[9] E. L. Ratnam, S. R. Weller, C. M. Kellett, and A. T. Murray, "Residential load and rooftop PV generation : an Australian distribution network dataset," *International Journal of Sustainable Energy*, vol. 36, no. 8, pp. 787–806, 2017.

[10] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Athena Scientific, 2005.

[11] W. B. Powell, "Perspectives of approximate dynamic programming," *Annals of Operations Research*, pp. 1–38, 2012.

[12] W. B. Powell and S. Meisel, "Tutorial on Stochastic Optimization in Energy - Part I : Modeling and Policies," *IEEE Transactions on Power Systems*, vol. 31, no. 2, pp. 1459–1467, March 2016.

[13] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[14] S. Caux, W. Hankache, M. Fadel, and D. Hissel, "On-line fuzzy energy management for hybrid fuel cell systems," *International Journal of Hydrogen Energy*, vol. 35, no. 5, pp. 2134–2143, 2010.

[15] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, "Lipschitzian optimization without the Lipschitz constant," *Journal of Optimization Theory and Applications*, vol. 79, no. 1, pp. 157–181, Oct. 1993, found jan 2014 (DIRECT algo used by Kim :2007 :JoPS).

[16] S. G. Johnson, "The NLopt nonlinear-optimization package." [Online]. Available : http://github.com/stevengj/nlopt

[17] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998, found jun 2014 (EGO algo used by Sareni).

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

[19] M. L. Puterman, *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2005.

[20] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*, 1st ed. Boca Raton : CRC Press, 2010.

[21] M. Pereira and L. Pinto, "Multi-stage stochastic optimization applied to energy planning," *Mathematical Programming*, vol. 52, no. 1-3, pp. 359–375, 1991.

[22] P. Haessig, T. Kovaltchouk, B. Multon, H. Ben Ahmed, and S. Lascaud, "Computing an Optimal Control Policy for an Energy Storage," in *6th European Conference on Python in Science (EuroSciPy 2013), Brussels, Belgium*, Aug. 2013, pp. 51–58.

[23] P. Haessig, "Dimensionnement & gestion d'un stockage d'énergie pour l'atténuation des incertitudes de production éolienne," Ph.D. dissertation, ENS Cachan, Jul. 2014.

[24] O. Dowson and L. Kapelevich, "SDDP.jl : A Julia Package for Stochastic Dual Dynamic Programming," *INFORMS Journal on Computing*, vol. 0, no. 0, p. null, 2020, available online https ://doi.org/10.1287/ijoc.2020.0987.

[25] P. J. Brockwell and R. A. Davis, *Time Series : Theory and Methods*, second edition ed., ser. Springer Series in Statistics. Springer New York, 1991.

[26] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*, ser. Springer Series in Operations Research and Financial Engineering. New York, NY : Springer, 2011.

[27] A. Shapiro and A. Philpott, "A Tutorial on Stochastic Programming," 2007. [Online]. Available : https://sites.gatech.edu/alexander-shapiro/publications/

[28] S. Lucia, J. A. E. Andersson, H. Brandt, M. Diehl, and S. Engell, "Handling uncertainty in economic nonlinear model predictive control : A comparative case study," *Journal of Process Control*, vol. 24, no. 8, pp. 1247–1259, 2014, read nov 2019 (found for μG dyn sizing NL SP, but also interesting for JJ's OLFC).

[29] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, ser. Monographs on Statistics & Applied Probability. Chapman & Hall/CRC, 1994.

[30] R. Hyndman and S. Fan, "Density Forecasting for Long-Term Peak Electricity Demand," *IEEE Trans. Power Syst.*, vol. 25, no. 2, pp. 1142–1153, May 2010.