Time decomposition methods for optimal management of energy storage under stochasticity

Tristan Rigaut

UNIVERSITÉ PARIS-EST

ÉCOLE DOCTORALE : MATHÉMATIQUES ET SCIENCES
ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

SPÉCIALITÉ : MATHÉMATIQUES APPLIQUÉES

présentée par

Tristan RIGAUT

Time Decomposition Methods For Optimal Management Of Energy Storage Under Stochasticity

---

Thèse préparée à EFFICACITY
au LISIS (COSYS), IFSTTAR
et au CERMICS, École des Ponts ParisTech

Soutenue le 16 Mai 2019 devant le Jury composé de :

JURY

Rapporteurs :	Nadia Oudjane	Électricité de France (EDF) R&D
	Alois Pichler	Technische Universität, Chemnitz
Directeurs :	Frédéric Bourquin	IFSTTAR
	Jean-Philippe Chancelier	École des Ponts ParisTech
Examinateurs :	Pierre Carpentier	ENSTA ParisTech
	Pierre Haessig	CentraleSupélec
	Andy Philpott	University of Auckland
	Julien Waeytens	IFSTTAR

Abstract

The development of energy storage paves the way to innovative methods to manage energy at a local scale. Micro grids are a novel kind of electrical grids with local production (renewable and waste energy), local demand, local storage and an Energy Management System (EMS). A wide literature already studies EMS implementations in micro grids but the produced methods are not exhaustively framed and compared. One of the main difficulty in micro grids energy management is to handle the different dynamics of electrical devices. Current variations are lighting fast, solar power changes quickly, different kind of storage react at different paces and batteries ageing is a slow process. We study a mathematical framework and algorithms, based on multistage stochastic optimization theory and Dynamic Programming, to model and solve energy management problems in micro grids with time decomposition methods.

In the first part of this thesis, *Contributions to time decomposition in multistage stochastic optimization*, we present a general framework to decompose temporally large scale stochastic optimization problems into smaller subproblems. We then classify multiple existing resolution methods inside this framework.

In the second part, *Stochastic optimization of energy storage for management of micro grids*, we compare different methods presented in the first part on realistic applications. First we control a battery and a ventilation in a subway station recovering subways braking energy with four different algorithms. Then we present how these results could be implemented on a real micro grid. We implement a fast online control method to stabilize the voltage in a simulated islanded DC micro grid connecting solar panels, an electrical load and two sorts of energy storage: a battery and a supercapacitor. Finally we apply our time decomposition framework to a problem of long term aging and energy management of a storage in a micro grid. This last chapter introduces a framework to model time decomposition of micro grids hierarchical control architectures, as well as two algorithms to solve temporally large scale stochastic optimization problems.

In the third part, *Softwares and experiments*, we present DynOpt.jl, a Julia language package developed to produce all the results of this thesis and more. Then we study an application of this software to the control of a real test bed: the energy aware temperature regulation of a real house in the equipment named “Sense City".

Remerciements

Cette page conclut mes trois années les plus riches en terme d'éveil scientifique, écologique, professionnel et amical. Pour respecter la flèche du temps je remercie d'abord mes incroyables parents que j'ai rencontré il y a 29 ans et sans qui je n'aurais pas de doigts pour taper des remerciements. Je remercie également ma sœur que je vais pouvoir voir plus souvent je l'espère.

Je remercie Alexandre Nassiopoulos, André Pény et mon directeur Frédéric Bourquin. Sans eux je n'aurais pas intégré Efficacity et son écosystème passionnant un an avant ma thèse. Avec eux j'ai appris les bases du contrôle, comment freinent les métros et comment stocker de l'énergie dans des mûrs.

Merci à Pierre Carpentier, Michel De Lara et mon autre directeur Jean-Philippe Chancelier pour leur encadrement de choc et leur phénoménal sens du partage des sciences, du code, des blagues de matheux et du voyage.

Merci à Julien Waeytens pour sa bonne humeur communicative et pour m'avoir aidé à faire atterrir les maths et le code jusque dans un chalet de banlieue. Merci aussi à Alessio Iovine même s'il m'a obligé à faire du matlab. Merci à Mathieu Aveline qui me rappelle souvent que ça peut peut être aller dans le bon sens.

Mes plus vifs remerciements aux membres du jury de cette thèse, Nadia Oudjane, Alois Pichler, Pierre Haessig et Andy Philpott pour leurs remarques et leur présence le 16 Mai.

Un grand merci à tous mes collègues d'Efficacity, du CERMICS et mes amis qui étaient présents le jour J, ou non (comme Deleu) mais à tellement d'autres soirées aussi.

Je remercie chaudement Débiche, Billon, Alexou, Nicobear et c'est granulaire qui parviennent à mettre l'ambiance n'importe où, même dans leurs quartiers.

Une pensée particulière pour mon cobureau Frapy qui a été une grande source d'inspiration, jusque dans cette phrase, et avec qui je partage mes meilleurs souvenirs de thésard comme les caïpirinhas sur une plage d'Ilha Grande ou les heures à coder en Juju, au choix.

Et je finirai par un gros bisous à Anne So qui m'a particulièrement soutenu surtout dans la dernière ligne droite.

Contents

Contents (detailed)

Résumé en français

Résumé général : Méthodes de décomposition temporelle pour la gestion optimale de stockage d'énergie sous incertitudes

L'évolution du stockage d'énergie permet de développer des méthodes innovantes de gestion de l'énergie à une échelle locale. Les micro réseaux électriques sont une forme émergente de petits réseaux électriques munis de production locale (en majorité des énergies renouvelables), de stockage d'énergie et en particulier d'un système de gestion de l'énergie (EMS pour Energy Management System). De nombreuses études et recherches scientifiques ont été menées pour proposer diverses stratégies d'implémentations de ces EMS. Néanmoins il n'existe pas à ce jour d'articulation claire et formelle de ces méthodes permettant leur comparaison. L'une des principales difficultés que les EMS ont a affronté, est la gestion des dynamiques radicalement différentes des systèmes énergétiques. Les variations de courant vont à la vitesse de l'électron, la production d'énergie solaire photovoltaïque varie au gré des nuages et différentes technologies de stockages peuvent réagir plus ou moins vite à ces phénomènes imprévisibles. Nous étudions dans ce manuscrit, un formalisme mathématique et des algorithmes basés sur la théorie de l'optimisation stochastique multi-étapes et la Programmation Dynamique. Ce formalisme permet de modéliser et de résoudre des problèmes de décisions inter-temporelles en présence d'incertitudes, à l'aide de méthodes de décomposition temporelle que nous appliquons à des problèmes de gestion de l'énergie.

Dans la première partie de cette thèse *Contributions à la décomposition temporelle en optimisation stochastique multi-étapes* nous présentons le formalisme général que nous utilisons pour décomposer, en temps, les problèmes d'optimisation stochastique avec un grand nombre de pas de temps. Nous classifions ensuite différentes méthodes de contrôle optimal au sein de ce formalisme.

Dans la seconde partie *optimisation stochastique de stockage d'énergie pour la gestion des micro réseaux*, nous comparons différentes méthodes, introduites dans la première partie, sur des cas d'application réels. Dans un premier temps nous contrôlons avec quatre algorithmes différents une batterie ainsi que des ventilations dans une station de métro récupérant de l'énergie de freinage des trains. Dans un second temps nous montrons comment ces algorithmes pourraient être implémentés sur un système réel à l'aide d'une architecture de contrôle hiérarchique d'un micro réseau électrique en courant continu. Le micro réseau étudié connecte cette fois ci de l'énergie photovoltaïque à une batterie, une super-capacité et une charge électrique. Enfin nous appliquons le formalisme

de décomposition par blocs temporels présenté dans la première partie pour traiter un problème de gestion de charge de batterie mais aussi de son vieillissement long terme. Ce dernier chapitre introduit deux algorithmes, basés sur la décomposition par blocs temporels, qui pourraient être utilisés pour le contrôle hiérarchique de micro grids ou les problèmes d'optimisation stochastique présentant un grand nombre de pas de temps.

Dans la troisième et dernière partie, *Logiciels et expériences* nous présentons DynOpt.jl un paquet développé en langage Julia qui a permis de développer toutes les applications de cette thèse et bien d'autres. Nous étudions enfin l'utilisation de ce paquet dans un cas de pilotage réel de système énergétique : la gestion intelligente de la température dans une maison de l'équipement d'excellence Sense City.

Motivations, contexte et structure de la thèse

Motivations : gestion locale de l'énergie

Les secteurs de l'électricité et de la chaleur représentaient à eux deux 42% des émissions CO2 planétaires en 2016[1]. Les unités de génération au charbon et au fuel sont les principales responsables des émissions de gaz à effet de serre du secteur de la production électrique. Les centrales nucléaires et certaines énergies renouvelables permettent de diminuer ces émissions de CO2, mais seules ces dernières sont considérées comme des énergies vertes. La production électrique assurée par les énergies renouvelables est intermittente et imprévisible, elle évolue au gré du vent, qui ne souffle pas constamment, et de l'ensoleillement qui peut être obscurci par les nuages. Pour assurer en permanence l'équilibre électrique entre production et demande, il est nécessaire de recourir à des technologies de stockage. De nombreux pays ont intégré des unités de stockage de grande échelle dans leur mix énergétique. Mentionnons notamment les barrages hydroélectriques ou les vallées hydrauliques permettant de pomper de l'eau lorsque la production excède la demande pour la turbiner plus tard dans le cas inverse.

Ces stockages de grande échelle nécessitent de prendre des décisions de quantité d'eau pompée/turbinée en temps réel, sans connaître précisément la production ou la demande future à l'échelle régionale ou nationale. L'optimisation stochastique est la théorie mathématique qui permet de modéliser ces problèmes de décisions dans l'incertain. Les barrages hydroélectriques sont l'un des cas pratiques historiques des praticiens de l'optimisation stochastique. Mais ces stockages de grande échelle ne sont pas les seuls moyens pour équilibrer la production et la demande dans le mix énergétique. Les batteries, volants d'inerties, systèmes à air comprimé ou beaucoup d'autres technologies innovantes existent pour stocker l'énergie. Dans ce manuscrit nous nous concentrons sur les batteries qui sont la plupart du temps de capacité bien plus réduite que des barrages. Elles sont plus souvent utilisées à une échelle locale : le bâtiment, le quartier, un micro réseau électrique... Ces unités de stockage permettent la gestion locale de l'énergie, autrement dit à petite échelle. Mais ils sont pour le moment coûteux et leur construction est rarement écologique. La gestion locale de l'énergie permet également d'utiliser d'autres moyens de stockage de l'énergie déjà existants mais sous exploités pour le moment. La chaleur dans les maisons ou les ballons d'eau chaude sont des exemples de stocks d'énergie sous

[1] https://www.iea.org/statistics/co2emissions/

exploités car non optimisés.

Dans cette thèse nous étudions des méthodes de gestion en temps réel de stocks énergétiques dans des petits réseaux électriques que nous appelons micro grids. Ces micro grids contiennent de la production locale (souvent renouvelable ou fatale), une demande électrique et différentes sortes de stockage : batteries, super capacités ou la chaleur des bâtiments. Nous développons des méthodes de gestion basées sur l'optimisation stochastique. Nous résolvons des problèmes de gestion de l'énergie et des stockages en utilisant cette théorie et nous proposons des extensions permettant de traiter des problèmes de gestion long terme des équipements en prenant en compte des aspects investissement, vieillissement et maintenance. La plupart des résultats théoriques de cette thèse font l'objet d'expérimentations numériques (et/ou réelles) à l'aide d'une bibliothèque générique d'optimisation stochastique, DynOpt.jl, que nous avons développée.

Contexte

Ce doctorat a été financé par Efficacity[2], l'institut de R&D pour la transition énergétique des villes. Il a été dirigé par Frédéric Bourquin de l'IFSTTAR[3] et Jean Philippe Chancelier du CERMICS[4], laboratoire de L'École des Ponts ParisTech. Un complément de supervision a été apporté par Pierre Carpentier de l'ENSTA[5], Michel De Lara du CERMICS et Julien Waeytens de l'IFSTTAR.

Structure

Ce manuscrit est divisé en trois grandes parties, elles-mêmes subdivisées en chapitres.

La première partie est dédiée à la formalisation des méthodes de décomposition en temps des problèmes d'optimisation stochastique. Cela qualifie les méthodes consistants à décomposer un problème d'optimisation stochastique multi pas de temps en plusieurs sous problèmes contenant un nombre inférieur de pas de temps. Elles s'articulent autour de l'équation de Bellman [1]. Le formalisme de décomposition temporelle que nous développons est ensuite utilisé pour construire un modèle générique pour les politique de contrôle en ligne de systèmes stochastiques dynamiques.

La seconde partie applique le formalise de la première à des cas d'usages réels de gestion de l'énergie. Le premier cas étudié est celui d'une station de métro où est récupérée l'énergie de freinage des trains à l'aide d'une batterie et où la ventilation est contrôlée pour minimiser sa consommation d'énergie en assurant une bonne qualité de l'air. Le second cas concerne la stabilisation en tension d'un micro réseau électrique en courant continu à l'aide d'un algorithme de contrôle prédictif. Le troisième cas étudie la gestion long terme et temps réel d'une batterie dans un micro réseau électrique.

La troisième partie est divisée en deux chapitres. Le premier présente la bibliothèque d'optimisation stochastique, DynOpt.jl, développée à Efficacity par François Pacaud et T.R., et utilisée pour toutes les applications de la seconde partie. Cette bibliothèque reprend le formalisme de la première partie dans un cadre plus opérationnel. Le second

[2]https://www.efficacity.com
[3]https://www.ifsttar.fr
[4]https://cermics-lab.enpc.fr
[5]https://uma.ensta-paristech.fr/

et dernier chapitre présente un cas de pilotage énergétique réel de la thermique dans un chalet avec DynOpt. Ce bâtiment, présenté Figure 1, fait partie de l'équipement d'excellence Sense City.

Figure 1: Équipement Sense City.[9] Source[6]

L'optimisation pour la gestion de l'énergie à l'échelle locale

Energy management systems pour micro grids

Dans ce manuscrit nous appelons micro grid un réseau électrique contenant une production locale incertaine (photovoltaïque, éolien, énergie de freinage...), une demande électrique incertaine, et un ou plusieurs stockages. Ce micro réseau électrique peut être connecté au réseau national ou non. Dans ce dernier cas on le dit en îlotage. Lorsque le micro réseau est connecté à la grille nationale, il est possible de payer pour importer de l'électricité. Cela rend plus facile l'équilibrage du réseau lorsque la production ou le stockage ne sont pas suffisants pour couvrir la demande. L'objectif est alors de minimiser la facture énergétique du micro réseau. Dans ce manuscrit nous étudions majoritairement des micro réseaux connectés à la grille nationale, excepté dans le Chapitre 4 où nous étudions un problème de stabilisation de micro réseau en îlotage.

Contrôle hiérarchique Dans le Chapitre 4 de cette thèse, nous étudions un micro grid en courant continu (DC). Ces micro réseaux permettent de connecter différents équipements en courant continu, tels que des panneaux solaires, des LEDs ou des bornes de recharge de véhicules électriques, sans dispositifs de conversion courant alternatif/continu. Les micro grids sont en général gérés à l'aide d'une architecture de contrôle hiérarchique [12] pour prendre en compte les différentes dynamiques des multiples équipements. La philosophie de cette implémentation est la suivante : un niveau supérieur calcule des

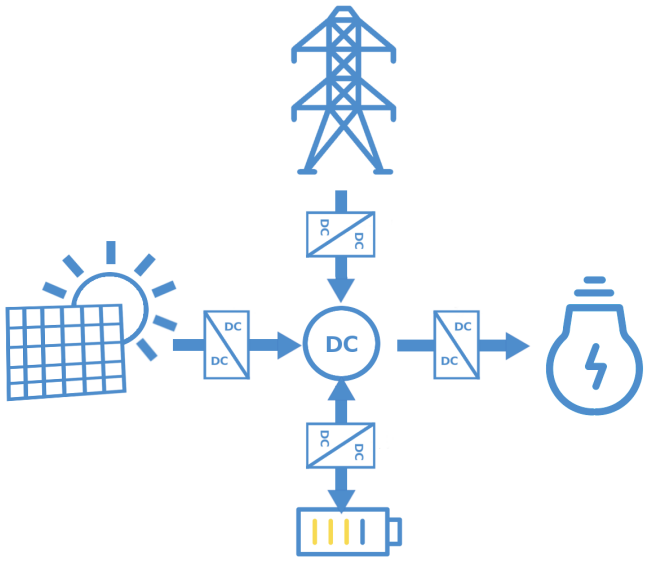

Figure 2: Micro grid DC simple

cibles ou consignes pour un niveau inférieur qui est en charge de suivre ces cibles ou consignes. Dans ce manuscrit nous divisons cette architecture en trois niveaux qui répondent à trois questions formulées simplement.

- Troisième niveau (pas de temps de la minute) : quelle quantité d'énergie doit stocker la batterie toutes les minutes pour minimiser la facture énergétique sur la journée?
- Deuxième niveau (pas de temps de la seconde) : quelle puissance doit on imposer à la batterie, gâcher de la production ou consommer sur le réseau national pour assurer la stabilité en tension du micro réseau tout en atteignant la cible d'énergie du troisième niveau?
- Premier niveau (pas de temps de la milliseconde) : quel courant doit on soutirer des équipements toutes les millisecondes pour assurer la consigne du second niveau?

Dans le Chapitre 5 nous ajoutons un quatrième niveau, au pas de temps de la journée, pour gérer le vieillissement des équipements.

Que contrôle t'on dans un micro réseau électrique? Le premier niveau est souvent ignoré par les praticiens de l'optimisation stochastique qui considèrent qu'il sera implémenté par des ingénieurs dont c'est la spécialité. Cependant il est important de comprendre comment chaque équipement d'un micro grid est effectivement piloté pour modéliser au mieux le problème décisionnel qui se pose pour le gestionnaire du réseau. Un optimiseur considérera en général que la charge/décharge d'une batterie est directement contrôlée, ce qui est imprécis. Les équipements contrôlés dans le réseau représenté Figure 2 sont en général les convertisseurs DC/DC (rectangles de la figure) [11]. De manière basique ils contiennent des interrupteurs ON/OFF qui sont activés/désactivés

à intervalles réguliers de manière à produire le courant requis par le premier niveau de contrôle. En contrôlant ces convertisseurs il est possible de produire la bonne énergie de charge/décharge en sortie de la batterie par exemple. Cela permet également de contrôler l'énergie consommée sur le réseau national ou gâchée des panneaux solaires, si trop d'énergie est produite par rapport à la demande. Il est aussi possible de ne pas satisfaire une partie de la demande, mais ce type d'action est en général à utiliser en dernier recours pour assurer la stabilité du réseau. Le cercle central contenant l'appellation DC représente le réseau en courant continu qui connecte tous les équipements entre eux. Il peut être considéré comme un très petit stockage d'énergie, tellement petit qu'il est ignoré au troisième niveau et remplacé par un équilibre des puissances entre les différents équipements, une loi de Kirchoff.

Optimisation stochastique multi-étapes

Selon les mots, traduits en français, de R. Tyrell Rockafellar[7], les problèmes d'optimisation stochastique sont caractérisés par la nécessité de prendre des décisions sans connaître précisément leur impact à l'avance[8]. Cette incertitude peut venir du caractère aléatoire de phénomènes physiques, financiers, comportementaux ou sociaux qui ne peuvent être prédits à l'avance.

Programmation Dynamique Stochastique. La Programmation Dynamique Stochastique [17, 2, 6] est une méthode mathématique permettant de résoudre certains problèmes d'optimisation stochastique multi-étapes. Elle se base sur le principe d'optimalité de Bellman [1] qui permet de décomposer un problème multi-étapes en une multitude de problèmes à 2 étapes imbriqués. Nous présentons donc l'équation de Bellman comme une méthode de décomposition temporelle des problèmes d'optimisation stochastique. L'un de nos travaux consiste à présenter l'équation de Bellman comme une méthode de décomposition par blocs temporels dans le formalisme mathématique le plus général possible. Nous utilisons ce formalisme pour construire un modèle permettant d'agencer les méthodes existantes de contrôle en ligne de systèmes stochastiques dynamiques.

La malédiction de la dimension. La complexité de l'algorithme de programmation dynamique stochastique standard, basé sur la discrétisation des espaces des états et des contrôles puis sur des recherches exhaustives imbriquées (boucles for), est exponentielle en le nombre de variables d'état, contrôle et d'aléa. C'est la fameuse malédiction de la dimension introduite par Bellman. De multiples méthodes ont été mises au point pour affronter cette malédiction de la dimension et calculer des solutions admissibles aux problèmes d'optimisation stochastique. Citons notamment la programmation stochastique sur arbres de scénarios (SP) [19], le contrôle prédictif (stochastique) (MPC) [3], la programmation dynamique approchée (ADP) [4], l'algorithme Stochastic Dual Dynamic Programming (SDDP) [13] ou l'apprentissage par renforcement (RL) [5, 16, 20]. Dans [3] et [15] les auteurs proposent une classification de toutes ces méthodes montrant qu'elles sont toutes basées sur l'équation de Bellman et différentes sortes d'approximation.

[7] `http://sites.math.washington.edu/~rtr/mypage.html`

[8] `http://sites.math.washington.edu/~rtr/uncertainty.pdf`

L'optimisation dans les Energy Management Systems des micro grids

Les Energy Management Systems (EMS) des micro réseaux requièrent de prendre des décisions en peu de temps pour contrôler des systèmes aux dynamiques rapides. Nous présentons dans le Chapitre 4 un exemple où des décisions de puissances d'équipements sont à prendre toutes les secondes, en moins d'une seconde. Une stratégie de contrôle, ou politique de contrôle, est dite "en ligne" parce qu'elle prend en compte l'état courant d'un système en temps réel pour calculer une décision, ou un contrôle.

Nous distinguons deux grandes classes de méthodes pour produire des politiques en ligne.

1. Totalement en ligne : certaines méthodes nécessitent uniquement de résoudre un problème d'optimisation mathématique en ligne pour calculer une décision connaissant l'état du système. C'est le cas par exemple de SP ou MPC qui remplacent les aléas par un ou plusieurs scénarios, produits d'une quelconque manière, afin de poser un problème de programmation mathématique.

2. Hors ligne - En ligne : au contraire d'autres méthodes consistent à pré-calculer, hors ligne, une certaine quantité de données pour les utiliser par la suite, en ligne, afin de produire une décision connaissant l'état du système en très peu de temps. C'est le cas par exemple de SDP, SDDP, ADP, RL...Ces méthodes ont en général l'avantage de simplifier la résolution du problème en ligne.

Toutes les méthodes que nous avons introduites ont déjà été appliquées, dans de multiples versions, à des problèmes de gestion de micro réseaux. Peu d'articles comparent ces méthodes entre elles. Nous présentons un petit état de l'art sur le sujet au Chapitre 3.

Contributions

Ce manuscrit est organisé comme une collection d'articles. Pour organiser la présentation ces articles sont groupés par parties. Parmi ces articles, deux sont publiés [18, 10], deux sont envoyés à des revues [8, 7] et les trois derniers sont finalisés comme chapitres de thèse mais pas en tant qu'articles de revue. Dans cette section nous présentons un résumé de chaque article.

Décomposition temporelle en contrôle optimal stochastique

Dans cette partie nous présentons un formalisme général de décomposition temporelle en optimisation stochastique.

Chapitre 1 Les problèmes d'optimisation stochastique multi-étapes sont complexes par essence car leurs solutions sont paramétrées par les étapes (pas de temps) et les incertitudes (scénarios). Leur caractère grande échelle incite à l'utilisation de méthodes de décomposition. Les méthodes les plus standards sont la décomposition temporelle — comme la programmation dynamique stochastique en contrôle optimal stochastique — et

la décomposition par scénarios — comme le recouvrement progressif en programmation stochastique. Nous présentons une méthode générale de décomposition des problèmes d'optimisation stochastique multi-étapes par blocs temporels, ce qui peut permettre de combiner la programmation stochastique et la programmation dynamique stochastique. Nous présentons une équation de programmation dynamique avec des fonctions valeurs définies sur l'espace des histoires (une histoire est une séquence d'incertitudes et de décisions). Nous présentons ensuite des conditions permettant la construction d'une variable d'état par réduction de l'histoire. Cette réduction est effectuée par blocs temporels, c'est à dire à des pas de temps qui ne sont pas forcément le pas de temps unitaire du problème. Cela permet de construire une équation de programmation dynamique réduite. Nous appliquons cette méthode de réduction à des problèmes à *deux échelles de temps* et à une nouvelle classe de problèmes que nous appelons *décision-hasard-décision* et qui permet de modéliser un grand nombre de problèmes réels, notamment dans la gestion de stocks. La méthode de décomposition par blocs temporels suit le schéma suivant : nous appliquons la programmation dynamique sur des blocs contenant un nombre suffisamment grand de pas de temps pour supposer l'indépendance entre blocs des aléas du problème. Nous produisons des fonctions de Bellman par blocs, liées récursivement par l'équation de Bellman par blocs. Ces fonctions sont toutes fonction valeur d'un problème d'optimisation stochastique multi-étapes, plus petit que le problème original, qui peut être résolu par la méthode de notre choix, programmation dynamique ou stochastique par exemple. Cette méthode générale permet de ne pas forcément faire d'hypothèse Markovienne à petit pas de temps mais uniquement entre plusieurs blocs, par exemple entre deux journées mais pas entre deux minutes.

Chapitre 2 Les solutions de problèmes d'optimisation stochastique multi-étapes sont des politiques, c'est à dire des applications de l'information passée vers l'espace des décisions, à chaque étape. Les praticiens calculent rarement les politiques complètes mais les évaluent en ligne à une étape donnée et une information passé donnée. Nous proposons dans ce chapitre un modèle théorique général permettant de construire des politiques de contrôle en ligne pour les problèmes d'optimisation stochastique multi-étapes. Notre approche met en avant le rôle des structures d'information dans la construction des politiques. Nous utilisons le formalisme du Chapitre 1 pour construire notre modèle théorique de politiques en ligne. Nous classifions dans ce chapitre, de multiples méthodes de contrôle existantes (SDP, SP, MPC) au sein de ce modèle théorique.

Méthodes de gestion de stockages pour des applications réelles

Dans cette partie nous présentons trois problèmes réels de gestion de l'énergie dans différents systèmes : une station de métro, un micro grid DC avec stockage hybride et un micro grid contenant un stockage pour lequel nous optimisons le vieillissement long terme.

Chapitre 3 Les stations représentent un tiers de la consommation électrique du réseau métro Parisien. Dans ces stations, la ventilation est l'un des dispositifs les plus énergivores; elle est en générale à débit maximal toute la journée pour assurer une bonne qualité de l'air et une grande réactivité en cas d'incendies. Nous présentons dans ce chapitre, un concept de gestion de l'énergie en station permettant de maintenir une qualité de l'air

comparable en diminuant la facture énergétique de la station. Ce système comprend une batterie permettant de récupérer de l'énergie de freinage des métros, une source d'énergie particulièrement erratique et imprévisible. Nous proposons un EMS qui contrôle les flux énergétique et la ventilation dans cette station, à petit pas de temps. Nous développons des algorithmes en mesure de satisfaire en permanence l'équilibre offre/demande tout en minimisant la facture énergétique. Nous avons donc développé des algorithmes permettant de gérer la variabilité de l'énergie de freinage. Ils sont basés sur le modèle présenté dans le Chapitre 2. Nous comparons équitablement des méthodes basées sur la programmation dynamique stochastique à d'autres basées sur le contrôle prédictif stochastique. Notre premier résultat est que tous nos algorithmes permettent de diminuer la facture de la station d'environ 30% en assurant une qualité de l'air comparable. Notre second résultat montre que dans ce cas, les méthodes basées sur la programmation dynamique sont plus performantes de quelques pourcents par rapport à celles basées sur le contrôle prédictif, avec un temps de calcul en ligne plus adapté aux phénomènes rapides.

Chapitre 4 Nous présentons un système de gestion de puissance d'un micro grid DC en îlotage contenant des panneaux photovoltaïques, des unités de stockage et une charge électrique. Ce système assure l'équilibre des puissances et la stabilité du réseau même lorsque la puissance de certains équipements n'est pas directement contrôlable et que la charge électrique varie de manière imprévisible au cours du temps. L'équilibre des puissances et la consigne en tension du réseau sont considérées comme des contraintes à satisfaire. Des simulations et un dispositif expérimental *Hardware In the Loop* sont présentés pour montrer l'efficacité du système développé.

Chapitre 5 Dans ce chapitre, nous appliquons la méthode théorique générale présentée au Chapitre 1 pour un cas réaliste de gestion de stockage énergétique dans un micro grid en prenant le vieillissement de l'équipement en plus de son comportement temps réel. Les dispositifs de stockages sont d'une importance majeure pour intégrer les énergies renouvelables intermittentes dans le mix énergétique. Malheureusement ces équipements restent pour le moment coûteux même si le développement du marché des véhicules électriques et des batteries tend à diminuer leur coût. Nous présentons un modèle d'optimisation stochastique qui vise à minimiser la facture énergétique d'un micro grid par un pilotage optimal du stockage, en prenant en compte l'investissement dans le stockage et son vieillissement. Pour une capacité de batterie donnée, il est nécessaire de calculer et d'évaluer sa politique de contrôle optimal pour minimiser la facture du réseau tout en assurant un bon vieillissement de l'équipement. Le vieillissement des batteries est un phénomène lent, sa dynamique est bien moins rapide que la dynamique de charge/décharge. Nous avons donc recours à des algorithmes multi-échelles de temps afin de prendre en compte ces dynamiques fondamentalement différentes. Ces algorithmes sont basés sur le formalisme du Chapitre 1 et combinent différentes méthodes d'optimisation stochastique sur un seul problème. Nous les appliquons sur trois exemples numériques différents. Le premier consiste à minimiser la facture énergétique long terme d'une maison équipée de photovoltaïque et d'un stockage. Le second consiste à gérer le vieillissement sur une semaine d'un équipement tout en minimisant la facture du réseau. Dans ce second cas, le problème présente une unique échelle de temps mais un grand nombre de pas de temps. Nous comparons nos résultat à une application directe de SDP et SDDP pour démontrer

que nos méthodes permettent de décomposer les problèmes comportant un grand nombre de pas de temps efficacement. Finalement nous montrons que l'un de nos algorithmes permet un dimensionnement rapide de systèmes de stockage en prenant en compte leur stratégie de pilotage optimal.

La boîte à outils DynOpt

Dans cette partie, nous présentons dans le Chapitre 6 la bibliothèque générique d'optimisation stochastique développée dans le cadre de cette thèse par François Pacaud[9] et T.R. Le Chapitre 7 présente une application de cette bibliothèque sur une expérimentation réelle : le contrôle de la température dans un chalet.

Chapitre 6 Nous présentons dans ce chapitre une boite à outils d'optimisation stochastique dynamique appelée DynOpt. Une interface de programmation permet à un utilisateur de modéliser un problème d'optimisation stochastique, puis de le résoudre avec l'un des multiples algorithmes implémentés. A partir d'une instance modélisée, un utilisateur est en mesure de développer une autre interface de programmation, pour un problème de gestion de l'énergie bien spécifique, qui peut être déployée pour des applications réelles. Par exemple, il est possible de construire un programme de dimensionnement et de pilotage optimal de batterie ou un dispositif de pilotage de température dans une maison, comme présenté dans le Chapitre 7. Cette bibliothèque s'articule autour du formalisme présenté dans le Chapitre 2 de cette thèse dans un cadre simplifié de contrôle optimal stochastique. Nous présentons dans un premier temps ce cadre simplifié, puis nous introduisons les différents objets qu'un utilisateur ou développeur potentiel sera amené à utiliser. Pour finir, nous montrons l'application de DynOpt sur un problème de pilotage de batterie dans un micro grid. L'efficacité de DynOpt repose fortement sur le concept de dispatche multiple, l'une des grandes forces du langage Julia.

Chapitre 7 Nous présentons une implémentation sur un cas de pilotage réel de DynOpt. Nous pilotons la température dans un chalet de l'équipement d'excellence de Sense City en minimisant la facture du chauffage électrique. Ce dispositif de pilotage utilise un modèle de la thermique du chalet basé sur une analogie électrique, un modèle RC. Nous présentons une méthode de calibration de ce modèle RC ne nécessitant aucune intervention d'un utilisateur. Puis nous modélisons le problème avec DynOpt et déployons une API conteneurisée dans un serveur pour piloter le chauffage dans le chalet. Nous présentons les résultats de ce pilotage sur une semaine.

Perspectives par chapitres

Nous présentons les perspectives de cette thèse par chapitres.

Chapitre 1 Le formalisme de décomposition temporelle développé dans ce chapitre permet de mélanger différentes méthodes de contrôle optimal stochastique pour la résolution de problèmes présentant un grand nombre de pas de temps. Nous présentons une

[9]https://cermics.enpc.fr/ pacaudf/

application dans le Chapitre 5 en mélangeant plusieurs méthodes basées sur la programmation dynamique stochastique. Il serait intéressant d'effectuer le même travail avec des méthodes de programmation stochastique, en particulier le recouvrement progressif, qui ne nécessitent pas d'hypothèse markovienne pour résoudre un problème.

Chapitre 2 Nous avons classifié différentes méthodes existantes de contrôle en ligne de systèmes stochastiques dynamiques à l'aide de notre modèle théorique. Il est possible de continuer à intégrer d'autres méthodes, en particulier l'apprentissage par renforcement, et de faire la lumière sur leur utilisation formelle de l'information en ligne.

Chapitre 3 Nous avons comparé différentes méthodes de contrôle pour la minimisation de la facture énergétique d'une station sur une journée. Il reste à appliquer les méthodes du Chapitre 5 pour dimensionner au mieux le stockage d'énergie pour ce genre d'applications.

Chapitre 4 Nous avons présenté une implémentation de contrôleur en puissance de réseau DC basé sur du contrôle prédictif permettant de suivre une consigne fournie par un niveau de contrôle supérieur. Il serait intéressant de comparer cette approche classique, mais heuristique, à une méthode multi-échelles de temps utilisant les algorithmes développés dans le Chapitre 5.

Chapitre 5 Nous avons présenté des résultats prometteurs concernant la performance de nos algorithmes de décomposition temporelle sur les problèmes présentant un grand nombre de pas de temps. De nombreuses méthodes de résolution ont une efficacité ou une complexité dépendant fortement du nombre de pas de temps. En particulier la programmation stochastique sur arbres de scénarios a une complexité exponentielle en le nombre de pas de temps et des algorithmes comme SDDP ont une convergence et une qualité de solution dépendant du nombre de pas de temps. Il serait intéressant de généraliser l'application de nos algorithmes pour améliorer l'efficacité de ces méthodes.

Chapitre 6 Nous avons présenté notre bibliothèque d'optimisation stochastique DynOpt, qui contient les méthodes présentées dans le Chapitre 2 de cette thèse. Tout comme pour le Chapitre 2 nous prévoyons d'ajouter des méthodes d'apprentissage par renforcement à l'arsenal algorithmique de DynOpt.

Chapitre 7 Nous avons appliqué une méthode de contrôle basée sur SDDP pour le contrôle de la thermique d'une maison et pour montrer la faisabilité d'un déploiement de DynOpt. Il nous reste à la comparer à un contrôle plus classique, de type PID, et à effectuer cette comparaison sur une typologie de bâtiments pour déterminer dans quels cas la méthode est intéressante.

Introduction

Motivations, context and structure

Motivations: local energy management

In 2016, electricity and heat sector represented 42% of the world global CO2 emissions [10]. Coal and fuel based generation units are the main responsible for the greenhouse gases emissions of electricity. Only nuclear power plants and renewable energies can help to lower these emissions. But only the latter are considered as green energies. As the wind does not always blow and the sun is sometimes hidden behind clouds, the integration of renewable energies requires energy storage. Many countries have already integrated large scale energy storage facilities such as water dams or even hydro valleys. These are used to balance electricity production and demand at a regional or national scale. As wind or solar power production is not perfectly predictable, the integration of renewable energies introduces uncertainty regarding electricity production in the energy mix.

Stochastic optimizers have long worked on water dams management to mitigate this uncertainty. But dams are not the only way to balance stochastic production and demand. Batteries, flywheels, compressed air or many other technologies are innovative ways to storage energy. In this thesis we focus on batteries which are most often significantly smaller than water dams, in terms of the amount of energy they can store. They are rarely used at a national or regional scale but rather at a building or neighborhood scale. Such systems then bring the concept of local energy management, that is energy management at a small scale. These devices are costly and their construction is rarely eco-friendly. Local energy management systems bring other opportunities to mitigate renewables uncertainty at a small scale. Heat in houses or in water heaters, for instance, is already existing and a not fully exploited energy storage.

In this thesis we study methods to implement energy management systems in small electrical grids that we call micro grids. These micro grids contain renewable production, or energy recovery, uncertain electrical demand and different kind of storages: batteries, supercapacitors or temperature in houses. The developed methods are based on Stochastic Optimization theory. We frame energy management problems as well as storage management methods using this theory and we propose extensions to solve long term problems so as to improve the economic profitability of costly energy storage systems. Most of the theoretical results presented in this thesis are implemented in a software library (called DynOpt.jl) that is used to solve real local energy management problems.

[10] https://www.iea.org/statistics/co2emissions/

Context

This PhD thesis has been financed by Efficacity[11] a French Research and Development institute dedicated to urban energy transition. This PhD thesis has been directed by Frédéric Bourquin and Jean Philippe Chancelier. It has been supervised as well by Michel De Lara, Pierre Carpentier, researchers at CERMICS[12] and UMA[13], for the stochastic optimization part. It was also supervised by Julien Waeytens, an IFSTTAR[14] researcher, for the experimental part, that is, the implementation of stochastic optimal control strategies to a real system.

Structure

Part 1 of this manuscript is dedicated to contributions to a formalization of time decomposition methods in stochastic optimization problems. By that, we mean methods that decompose a multistage stochastic optimization problem into smaller problems displaying a fewer number of time stages. It articulates around the well know Bellman equation [1]. This time decomposition framework is then used to build a template for online control policies of stochastic dynamical systems. A contribution of the thesis is to classify well known control methods using this template.

Part 2 uses the Part 1 formalism to model and solve applied energy management problems. We present different realistic cases where we apply stochastic optimization methods to improve the energy efficiency of different micro grids. The first realistic case studied is a subway station with a battery recovering trains braking energy and intelligent ventilation. The second is an islanded DC micro grid with solar panels, an electrical load and a hybrid storage system to stabilize the power of the grid and recover as much renewable energy as possible. The third application is the optimal long term management of an energy storage system in a micro grid connected to a national grid with a special focus on the aging of the storage.

Part 3 is divided in two chapters. In Chapter 6 we describe the numerical library called DynOpt, developed at Efficacity by François Pacaud and T.R., designed according to the formalism described in Part 1. Note that all the simulations of Part 2 have been performed using DynOpt. In Chater 7, the mini city demonstrator Sense City, displayed Figure 3, is presented. A contribution of the thesis is a real life application consisting of controlling temperature in a building of Sense City demonstrator with the help of DynOpt.

[11] https://www.efficacity.com
[12] https://cermics-lab.enpc.fr
[13] https://uma.ensta-paristech.fr/
[14] https://www.ifsttar.fr

Figure 3: Sense City demonstrator.[9] Source[15]

Background in microgrids energy management and optimization

Microgrids energy management

In the sequel we call a micro grid a small electrical grid with an uncertain local production, e.g regenerative braking or solar panels, an uncertain load, e.g a subway station or an house electrical demand and an electrical storage. This micro grid may or may not be connected to a national grid. The latter is called islanded mode. When it is connected to the national grid it is possible to pay electricity to an energy supplier if local production or storage is not enough to cover demand. In this PhD thesis we mostly study micro grids connected to a national grid except in Chapter 4 to demonstrate how to stabilize an islanded micro grid.

Hierarchical control architecture In Chapter 4 of this manuscript we study DC micro grids, that is, continuous current micro grids. DC micro grids offer the ability to connect DC systems easily such as solar panels, LEDs or electrical vehicles. Micro grids are controlled with a hierarchical Energy Management System to manage different physical phenomena that occur at different paces [12]. The philosophy is that an upper level computes targets or set-points and sends them to a lower level that aims at respecting these targets. We divide a micro grid control architecture in three levels answering three simply formulated questions.

- Third level (minutes): How much energy should we store in the storage every minutes to minimize national grid energy consumption over the day?

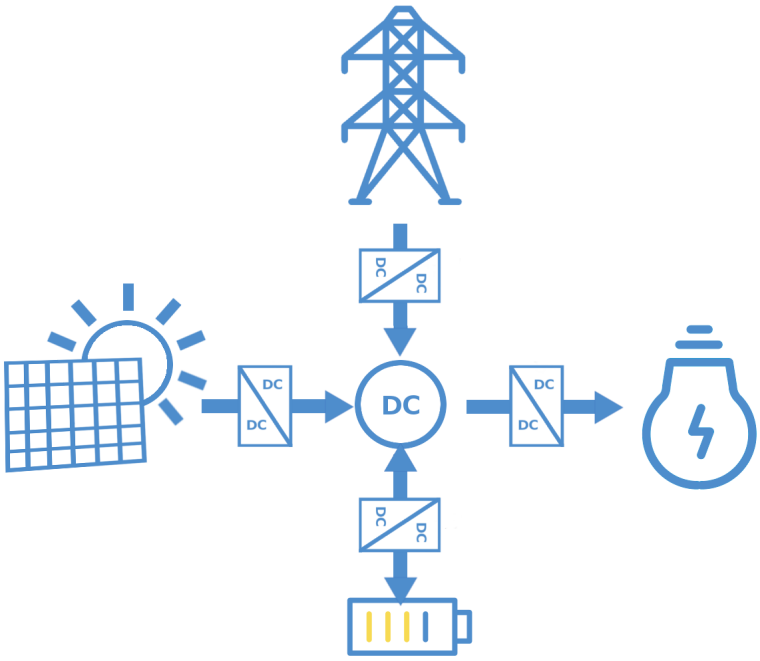

Figure 4: A simple DC micro grid

- Second level (seconds): How much power should we charge/discharge in the storage, curtail from the load, waste from the power source, consume on the national grid, to ensure grid voltage stability every seconds?
- First level (micro seconds): Which current should we draw from the equipment every micro-second to follow the set-points of the second level?

In Chapter 5 we add a fourth level with the time step of a day or multiple days to manage aging of the equipment.

What is controlled in a microgrid? The first level is often ignored by optimizers assuming it will be handled by electrical engineers. However it is important to understand how the equipment in a grid is controlled to model upper levels optimization problems properly. While an optimizer could consider the battery charge/discharge as a control, it is not totally accurate. The devices that are controlled in the grid presented Figure 4 are the DC/DC converters (rectangles on the figure) [11]. Basically they contain on/off switches that can be managed so as to let a prescribed current flows from the equipment to the grid and vice-versa. Controlling these switches, it is then possible to manage the charge/discharge power of the storage. It also enables to control the amount of power consumed on the national grid as well as the solar panels. As a consequence it makes it possible to decide to waste solar power if too much energy is produced. Likewise it is possible to curtail load from demand, that is provoking an outage on the demand side. These last two situations are to be avoided at all costs, however when the grid is islanded, it might be necessary to waste or curtail power to ensure the stability of the grid. Finally

the central DC node can be modeled as a small storage, so small that it is ignored at the third level and replaced by a node with a Kirchoff law, that is an energy supply/demand balance.

Multistage stochastic optimization

To quote R. Tyrrell Rockafellar [16], stochastic optimization problems "are characterized by the necessity of making decisions without knowing what their full effects will be"[17]. That uncertainty originates from random physical, financial or even behavioral phenomena that cannot be fully determined in advance.

Stochastic Dynamic programming. Stochastic Dynamic Programming [17, 2, 6] is a mathematical method to solve multistage stochastic optimization problems. It is built upon the so called Bellman's principle of optimality [1] that allows to decompose a multistage problem into multiple nested two-stages problems. Therefore we see Bellman equation as a time decomposition method of stochastic optimization problems. One of our work consists in presenting the Bellman equation as a time blocks decomposition method in the most general stochastic optimization formalism. We use this framework to build a template of online control policies that makes it possible to exhibit how classical control methods relate.

Tackling the curse of dimensionality. The complexity of the most classical Stochastic Dynamic Programming algorithm, based on discretization of search spaces and nested exhaustive searches (for loops), grows exponentially with the number of state variables and decision variables. This is the so called curse of dimensionality. Multiple methods exist to approximately solve multistage stochastic optimization problems, namely: Stochastic Programming (SP) [19], (Stochastic) Model Predictive Control (MPC) [3], Approximate Dynamic Programming (ADP) [4], Stochastic Dual Dynamic Programming (SDDP) [13], Reinforcement Learning (RL) [5, 16, 20]... In [3] and [15] the authors enforce to classify all these methods showing that all of them are a certain way to solve approximately the Bellman Equation.

Optimization methods for micro grids energy management systems

Energy management systems (EMS) for micro grids sometimes require to make control decisions quickly to handle the fast dynamics of the grid. We present in Chapter 4 an example where we need to compute decisions in less than one second. The control strategy, also called policy, is said to be "online" as it takes the current state of the system, in real time, to compute a control decision.

We distinguish two main classes of methods to produce a policy

[16] `http://sites.math.washington.edu/~rtr/mypage.html`

[17] `http://sites.math.washington.edu/~rtr/uncertainty.pdf`

- Fully online: some methods only require to solve an optimization problem online once the state of the system is observed. This is the idea of SP or MPC methods. They model future uncertainties using scenario trees or forecasts, then they formulate and solve a deterministic optimization problem using mathematical programming methods.
- Offline-online: on the contrary some methods require to make some calculations offline to produce a policy that will be used online. For example SDP, SDDP or Mixed Integer Dynamic Approximation Scheme (MIDAS) [14] compute functions offline to produce a policy that will be evaluated online to compute a control. This policy remains a function of the state, but the offline computation speeds up the online problem resolution.

All the methods we introduced have already been implemented in an Energy Management System for micro grids. However few papers compare the performance of different methods on a given system. We present some of these papers in Chapter 3.

Contributions

This thesis is organised as a collection of articles. To ease the presentation, the articles are regrouped in parts. Among these articles two are accepted [18, 10], two are submitted [8, 7], the last three are finalized as thesis chapters but not completely polished as papers. In this section we detail further the contents of each article.

Time decomposition in stochastic optimal control

In this part we present two chapters within a general formalism for time decomposition of multistage stochastic optimization problems.

Chapter 1 Multistage stochastic optimization problems are, by essence, complex because their solutions are indexed both by stages (time) and by uncertainties (scenarios). Their large scale nature makes decomposition methods appealing. The most common approaches are time decomposition — and state-based resolution methods, like stochastic dynamic programming, in stochastic optimal control — and scenario decomposition — like progressive hedging in stochastic programming. We present a method to decompose multistage stochastic optimization problems by time blocks, which covers both stochastic programming and stochastic dynamic programming. Once established a dynamic programming equation with value functions defined on the history space (a history is a sequence of uncertainties and controls), we provide conditions to reduce the history using a compressed "state" variable. This reduction is done by time blocks, that is, at stages that are not necessarily all the original unit stages, and we prove a reduced dynamic programming equation. Then, we apply the reduction method by time blocks to *two time-scales* stochastic optimization problems and to a novel class of so-called *decision-hazard-decision* problems, arising in many practical situations, like in stock management. The *time blocks decomposition* scheme is as follows: we use dynamic programming at slow time scale where the slow time scale noises are supposed to be stagewise independent, and

we produce slow time scale Bellman functions; then, it remains to solve short time scale problems (by stochastic programming or dynamic programming), within two consecutive slow time steps, with the final short time scale cost given by the slow time scale Bellman functions, and without assuming stagewise independence for the short time scale noises.

Chapter 2. The solutions of multistage stochastic optimization problems are policies, that is, mappings from past information into the current decision set, at each stage. In practice, one does not compute the whole policy but produces the value of the policy for the current argument and stage. We propose a general template to design online control policies for multistage stochastic optimization problems. Our approach stresses the role of information structures in the design of online policies. We use Chapter 1 formalism to build the general template of online policies. Then, we frame well known methods (Stochastic Dynamic Programming, Stochastic Programming, Stochastic Model Predictive Control) to produce online control policies within this template.

Energy storage management strategies for realistic applications

This part contains three articles that all focus on realistic examples: the energy management of a subway station, the control of an islanded DC micro grid and the long term management of energy storage in a micro grid.

Chapter 3 In the Paris subway system, stations represent about one third of the overall energy consumption. Within stations, ventilation is among the top consuming devices; it is operated at maximum airflow all day long, for air quality reasons. We present a concept of energy system that displays comparable air quality while consuming much less energy. The system comprises a battery that makes it possible to recover the trains braking energy, arriving under the form of erratic and strong peaks. We propose an energy management system (EMS) that, at short time scale, controls energy flows and ventilation airflow. By using proper optimization algorithms, we manage to match supply with demand, while minimizing energy daily costs. For this purpose, we have designed algorithms that take into account the braking variability. They are based on the so-called Stochastic Dynamic Programming (SDP) mathematical framework. We fairly compare SDP based algorithms with the widespread Model Predictive Control (MPC) ones. First, both SDP and MPC yield energy/money operating savings of the order of one third, compared to the current management without battery (our figure does not include the cost of the battery). Second, depending on the specific design, we observe that SDP outperforms MPC by a few percent, with an easier online numerical implementation.

Chapter 4 A power management controller for a DC micro grid containing renewable energy sources, storage elements and loads is presented. The controller ensures power balance and grid stability even when some devices are not controllable in terms of their power output, and environmental conditions and load vary in time. Power balance and desired voltage level for the DC micro grid are considered as constraints for the controller. Simulations and an experimental setup are implemented to show the effectiveness of the proposed control action.

Chapter 5 In this chapter, we apply Chapter 1 formalism to design algorithms for two time scales stochastic optimization problems arising from long term storage management. Energy storage devices are of major importance to integrate more renewable energies and demand-side management in a new energy mix. However batteries remain costly even if recent market developments in the field of electrical vehicles and stationary storage tend to decrease their cost. We present a stochastic optimization model aiming at minimizing the investment and maintenance costs of batteries for a house with solar panels. For any given capacity of battery it is necessary to compute a charge/discharge strategy as well as maintenance to maximize revenues provided by intraday energy arbitrage while ensuring an optimal long term aging of the storage devices. Long term aging is a slow process while charge/discharge control of a storage handles fast dynamics. For this purpose, we have designed algorithms that take into account this two time scales aspect in the decision making process. They are based on Chapter 1 time decomposition framework. These algorithms are applied to three numerical experiments. First one of them is used to control charge/discharge, aging and renewal of batteries for a house. Results show that it is economically significant to control aging. Second we apply and compare our algorithms on a simple charge/discharge and aging problem, that is a multistage stochastic optimization problem with many time steps. We compare our algorithms to SDP and Stochastic Dual Dynamic Programming and we observe that they are less computationally costly while displaying similar performances on the control of a storage. Finally we show how one of our algorithm can be used for the optimal sizing of a storage taking into account charge/discharge strategy as well as aging.

The DynOpt toolbox

In this part, in Chapter 6 we present a stochastic optimization toolbox developed by François Pacaud [18] and T.R. In Chapter 7, its application to a mini city test bed called Sense City is presented.

Chapter 6 We present in this chapter a stochastic dynamic optimization toolbox called DynOpt. A user interacts with DynOpt toolbox though an API which enables to build a stochastic optimization problem and solve it using dedicated algorithms. Moreover the user is able to build a specialized API for specific energy management problems she wants to solve. For instance, a user can build a battery sizing utility out of DynOpt or a house temperature controller as described in Chapter 7. This library is articulated around the concept of control policies that are distinguished by how they use online information to compute an optimal control for a stochastic dynamical system. We present the mathematical background that led to develop DynOpt. Then, we introduce the main objects for a potential user or a potential developer. Finally, we apply DynOpt on an energy storage toy problem and discuss the opportunities that it brings for real energy management applications. It is implemented as a Julia package, leveraging Julia multiple dispatch design.

[18] https://cermics.enpc.fr/ pacaudf/

Chapter 7 We present the implementation of a control strategy for the temperature of a real house, minimizing the energy consumption with stochastic thermal gains. We present a calibration method of an RC model, which models a building thermal behavior. This calibration method allows to determine some house thermal parameters but also to build a stochastic model of the thermal gains that are not generated by electrical heaters but originates from computers or bodies. These calibrated RC model and stochastic model of the thermal gains are used to build a control policy to minimize the energy consumption of the house while maintaining a comfortable temperature. This policy is computed using the SDDP algorithm using DynOpt, the toolbox introduced Chapter 6. We build a software architecture based on containerized APIs to calibrate the model, compute the control policy and apply the control policy online from a server or in the cloud without requiring a human input. We implement this architecture to a real house, controlling two electrical heaters in two rooms using smart plugs.

Perspectives by chapters

We present the perspectives of this thesis by chapters.

Chapter 1 The time blocks decomposition framework presented in this chapter makes it possible to mix stochastic programming and dynamic programming techniques to solve multistage stochastic optimization problems with an important number of time steps. SDP requires an independence assumption and has a complexity exponential in the number of state variables while stochastic programming has a complexity exponential in the number of time stages. Decomposition by time blocks makes it possible to assume randomness stage-wise independence on multiple time steps and not between consecutive time steps.

Chapter 2 We presented classical control methods in the framework of Chapter 1 with an online policies template based on Bellman equation. It would be interesting to integrate reinforcement learning techniques in this framework as they are deeply related to Bellman equation as well and proved efficient in numerous energy management applications.

Chapter 3 We compared four stochastic optimal control strategies to control energy storage in a subway station to minimize the energy consumption. The next step of this study is to apply the two time scales methods presented in Chapter 5 for the optimal aging and maintenance management of the battery as well as the ventilation.

Chapter 4 We presented a MPC controller to stabilize an islanded DC micro grid. In this chapter we assume that set-points are provided by an upper level for the state of charge of the battery. The next step is to apply the two time scales methods developed in Chapter 5 for the hierarchical control of micro grids. It would distinguish from classical control architectures of the literature as information between time steps would be exchanged through value functions, weights or targets instead of set-points. It could make

more efficient the MPC controller implementation and should produce better results as the control architecture would be based on a theoretical framework instead of a heuristic.

Chapter 5 We presented promising results regarding the mix of different stochastic dynamic programming techniques to solve multistage stochastic optimization problems with a large number of time steps. The two algorithms developed essentially made it possible to parallelize the resolution of a large problem over time. The same kind of approach could be applied to stochastic programming techniques that have a complexity exponential in the number of time steps. The weight algorithms presented uses an exhaustive search over weights to produce daily value functions, it could be more efficient to use a gradient based optimization method.

Chapter 6 We presented DynOpt, a toolbox to solve stochastic optimization problems using the methods introduced in Chapter 2. As in Chapter 2 our next step is to add reinforcement learning techniques to the toolbox as they are efficient for some energy management problems.

Chapter 7 We implemented a control strategy on a small house to manage the temperature and minimize the energy consumption. The next step is to generalize the strategy by applying it to a wide variety of buildings and multiple buildings at the same time.

Introduction bibliography

[1] R. Bellman. *Dynamic Programming*. Princeton University Press, New Jersey, 1957.

[2] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, third edition, 2005.

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

[4] D. P. Bertsekas. *Dynamic Programming and Optimal Control: Approximate Dynamic Programming*. Athena Scientific, fourth edition, 2012.

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

[6] P. Carpentier, J.-P. Chancelier, G. Cohen, and M. De Lara. *Stochastic Multi-Stage Optimization. At the Crossroads between Discrete Time Stochastic Control and Stochastic Programming*. Springer-Verlag, Berlin, 2015.

[7] P. Carpentier, J.-P. Chancelier, M. De Lara, F. Pacaud, and T. Rigaut. A template to design online policies for multistage stochastic optimization problems. working paper, Jan. 2019.

[8] P. Carpentier, J.-P. Chancelier, M. De Lara, and T. Rigaut. Time blocks decomposition of multistage stochastic optimization problem. 2018.

[9] F. Derkx, B. Lebental, T. Bourouina, F. Bourquin, C.-S. Cojocaru, E. Robine, and H. Van Damme. The Sense-City project. In *XVIIIth Symposium on Vibrations, Shocks and Noise*, page 9p, France, July 2012.

[10] A. Iovine, T. Rigaut, G. Damm, E. D. Santis, and M. D. D. Benedetto. Power management for a dc microgrid integrating renewables and storages. *Control Engineering Practice*, 85:59 – 79, 2019.

[11] A. Iovine, S. B. Siad, G. Damm, E. D. Santis, and M. D. D. Benedetto. Nonlinear control of a dc microgrid for the integration of photovoltaic panels. *IEEE Transactions on Automation Science and Engineering*, 14(2):524–535, April 2017.

[12] D. E. Olivares, A. Mehrizi-Sani, A. H. Etemadi, C. A. Canizares, R. Iravani, M. Kazerani, A. H. Hajimiragha, O. Gomis-Bellmunt, M. Saeedifard, and R. e. a. Palma-Behnke. Trends in Microgrid Control. *IEEE Trans. Smart Grid*, 5(4):1905–1919, 2014.

[13] M. V. Pereira and L. M. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1-3):359–375, 1991.

[14] A. Philpott, F. Wahid, and F. Bonnans. MIDAS: A Mixed Integer Dynamic Approximation Scheme. Research report, Inria Saclay Ile de France, June 2016.

[15] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[16] W. B. Powell. Clearing the jungle of stochastic optimization. *Informs*, pages 109–137, 2014.

[17] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

[18] T. Rigaut, P. Carpentier, J. Chancelier, M. D. Lara, and J. Waeytens. Stochastic optimization of braking energy storage and ventilation in a subway station. *IEEE Transactions on Power Systems*, pages 1–1, 2018.

[19] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[20] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Part I

Contributions to time decomposition in multistage stochastic optimization

Chapter 1

Time blocks decomposition of multistage stochastic optimization problems

This is a joint work with Pierre Carpentier, Jean-Philippe Chancelier and Michel De Lara.

Chapter Abstract

Multistage stochastic optimization problems are, by essence, complex because their solutions are indexed both by stages (time) and by uncertainties (scenarios). Their large scale nature makes decomposition methods appealing. The most common approaches are time decomposition — and state-based resolution methods, like stochastic dynamic programming, in stochastic optimal control — and scenario decomposition — like progressive hedging in stochastic programming. We present a method to decompose multistage stochastic optimization problems by time blocks, which covers both stochastic programming and stochastic dynamic programming. Once established a dynamic programming equation with value functions defined on the history space (a history is a sequence of uncertainties and controls), we provide conditions to reduce the history using a compressed "state" variable. This reduction is done by time blocks, that is, at stages that are not necessarily all the original unit stages, and we prove a reduced dynamic programming equation. Then, we apply the reduction method by time blocks to *two time-scales* stochastic optimization problems and to a novel class of so-called *decision-hazard-decision* problems, arising in many practical situations, like in stock management. The *time blocks decomposition* scheme is as follows: we use dynamic programming at slow time scale where the slow time scale noises are supposed to be stagewise independent, and we produce slow time scale Bellman functions; then, it remains to solve short time scale problems (by stochastic programming or dynamic programming), within two consecutive slow time steps, with the final short time scale cost given by the slow time scale Bellman functions, and without assuming stagewise independence for the short time scale noises.

Contents

1.1 Introduction

Multistage stochastic optimization problems are, by essence, complex because their solutions are indexed both by stages (time) and by uncertainties. Their large scale nature makes decomposition methods appealing. The most common approaches are time decomposition — and state-based resolution methods, like stochastic dynamic programming, in stochastic optimal control — and scenario decomposition — like progressive hedging in stochastic programming.

On the one hand, stochastic programming deals with an underlying random process taking a finite number of values, called scenarios [30]. Solutions are indexed by a scenario tree, the size of which explodes with the number of stages, hence generally few in practice. However, to overcome this obstacle, stochastic programming takes advantage of scenario decomposition methods (progressive hedging [29]). On the other hand, stochastic control

deals with a state model driven by a white noise, that is, the noise is made of a sequence of independent random variables. Under such assumptions, stochastic dynamic programming is able to handle many stages, as it offers reduction of the search for a solution among state feedbacks (instead of functions of the past noise) [22, 28].

In a word, dynamic programming is good at handling multiple stages — but at the price of assuming that noises are stagewise independent — whereas stochastic programming does not require such assumption, but can only handle a few stages. Could we take advantage of both methods? Is there a way to apply stochastic dynamic programming at a slow time scale — a scale at which noise would be statistically independent — crossing over short time scale optimization problems where independence would not hold? This question is one of the motivations of this paper.

We will provide a method to decompose multistage stochastic optimization problems by time blocks. In Sect. 1.2, we present a mathematical framework that covers both stochastic programming and stochastic dynamic programming. First, in §1.2.1, we sketch the literature in stochastic dynamic programming, in order to locate our contribution. Second, in §1.2.2, we formulate multistage stochastic optimization problems over a so-called history space, and we obtain a general dynamic programming equation. Then, we lay out the basic brick of time blocks decomposition, by revisiting the notion of "state" in Sect. 1.3. We lay out conditions under which we can reduce the history using a compressed "state" variable, but with a reduction done by time blocks, that is, at stages that are not necessarily all the original unit stages. We prove a reduced dynamic programming equation, and apply it to two classes of problems in Sect. 1.4. In §1.4.1, we detail the case of two time-scales stochastic optimization problems. In §1.4.2, we apply the reduction method by time blocks to a novel class consisting of decision-hazard-decision models. In the appendix, we relegate technical results, as well as the specific case of optimization with noise process.

1.2 Stochastic Dynamic Programming with Histories

We recall the standard approaches used to deal with a stochastic optimal control problem formulated in discrete time, and we highlight the differences with the framework used in this paper.

1.2.1 Background on Stochastic Dynamic Programming

We first recall the notion of stochastic kernel, used in the modeling of stochastic control problems. Let $(\mathbb{X}, \mathcal{X})$ and $(\mathbb{Y}, \mathcal{Y})$ be two measurable spaces. A *stochastic kernel* from $(\mathbb{X}, \mathcal{X})$ to $(\mathbb{Y}, \mathcal{Y})$ is a mapping $\rho : \mathbb{X} \times \mathcal{Y} \to [0, 1]$ such that

- for any $Y \in \mathcal{Y}$, $\rho(\cdot, Y)$ is $\mathcal{X}$-measurable;
- for any $x \in \mathbb{X}$, $\rho(x, \cdot)$ is a probability measure on $\mathcal{Y}$.

By a slight abuse of notation, a stochastic kernel is also denoted as a mapping $\rho : \mathbb{X} \to \Delta(\mathbb{Y})$ from the measurable space $(\mathbb{X}, \mathcal{X})$ towards the space $\Delta(\mathbb{Y})$ of probability measures over $(\mathbb{Y}, \mathcal{Y})$, with the property that the function $x \in \mathbb{X} \mapsto \int_Y \rho(x, \mathrm{d}y)$ is measurable for any $Y \in \mathcal{Y}$.

We now sketch the most classical frameworks for stochastic dynamic programming.

Witsenhausen Approach. The most general stochastic dynamic programming principle is sketched by Witsenhausen in [32]. However, we do not detail it as its formalism is too far from the following ones. We present here what Witsenhausen calls an optimal stochastic control problem in *standard form* (see [31]). The ingredients are the following:

1. time $t = t_0, t_0+1, \ldots, T-1, T$ is discrete, with integers $t_0 < T$;
2. $(\mathbb{X}_{t_0}, \mathcal{X}_{t_0})$, ..., $(\mathbb{X}_T, \mathcal{X}_T)$ are measurable spaces ("state" spaces);
3. $(\mathbb{U}_{t_0}, \mathcal{U}_{t_0})$,..., $(\mathbb{U}_{T-1}, \mathcal{U}_{T-1})$ are measurable spaces (decision spaces);
4. $\mathcal{I}_t$ is a subfield of $\mathcal{X}_t$, for $t = t_0, \ldots, T-1$ (information);
5. $f_t : (\mathbb{X}_t \times \mathbb{U}_t, \mathcal{X}_t \otimes \mathcal{U}_t) \to (\mathbb{X}_{t+1}, \mathcal{X}_{t+1})$ is measurable, for $t = t_0, \ldots, T-1$ (dynamics);
6. π_{t_0} is a probability on $(\mathbb{X}_{t_0}, \mathcal{X}_{t_0})$;
7. $j : (\mathbb{X}_T, \mathcal{X}_T) \to \mathbb{R}$ is a measurable function (criterion).

With these ingredients, Witsenhausen formulates a stochastic optimization problem, whose solutions are to be searched among adapted feedbacks, namely $\lambda_t : (\mathbb{X}_t, \mathcal{X}_t) \to (\mathbb{U}_t, \mathcal{U}_t)$ with the property that $\lambda_t^{-1}(\mathcal{U}_t) \subset \mathcal{I}_t$ for all $t = t_0, \ldots, T-1$. Then, he establishes a dynamic programming equation, where the Bellman functions are function of the (unconditional) distribution of the original state $x_t \in \mathbb{X}_t$, and where the minimization is done over adapted feedbacks.

The main objective of Witsenhausen is to establish a dynamic programming equation for nonclassical information patterns.

Evstigneev Approach. The ingredients of the approach developed in [26] are the following:

1. time $t = t_0, t_0+1, \ldots, T-1$ is discrete, with integers $t_0 < T$;
2. $(\mathbb{U}_{t_0}, \mathcal{U}_{t_0})$,..., $(\mathbb{U}_{T-1}, \mathcal{U}_{T-1})$ are measurable spaces (decision spaces);
3. $(\Omega, \mathcal{F})$ is a measurable space (Nature);
4. $\{\mathcal{F}_t\}_{t_0,\ldots,T-1}$ is a filtration of $\mathcal{F}$ (information);
5. $\mathbb{P}$ is a probability on $(\Omega, \mathcal{F})$;
6. $j : (\prod_{t=t_0,\ldots,T-1} \mathbb{U}_t \times \Omega, \bigotimes_{t=t_0,\ldots,T-1} \mathcal{U}_t \otimes \mathcal{F}) \to \mathbb{R}$ is a measurable function (criterion).

With these ingredients, Evstigneev formulates a stochastic optimization problem, whose solutions are to be searched among adapted processes, namely random processes with values in $\prod_{t=t_0,\ldots,T-1} \mathbb{U}_t$ and adapted to the filtration $\{\mathcal{F}_t\}_{t_0,\ldots,T-1}$. Then, he establishes a dynamic programming equation, where the Bellman function at time t is an $\mathcal{F}_t$-integrand depending on decisions up to time t (random variables) and where the minimization is done over $\mathcal{F}_t$-measurable random variables at time t.

The main objective of Evstigneev is to establish an existence theorem for an optimal adapted process (under proper technical assumptions, especially on the function j, that we do not detail here).

Bertsekas and Shreve Approach. The ingredients of the approach developed in [23] are the following:

1. time $t = t_0, t_0 + 1, \ldots, T-1, T$ is discrete, with integers $t_0 < T$;
2. $(\mathbb{X}_{t_0}, \mathcal{X}_{t_0})$, ..., $(\mathbb{X}_T, \mathcal{X}_T)$ are measurable spaces (state spaces);
3. $(\mathbb{U}_{t_0}, \mathcal{U}_{t_0})$,..., $(\mathbb{U}_{T-1}, \mathcal{U}_{T-1})$ are measurable spaces (decision spaces);
4. $(\mathbb{W}_{t_0}, \mathcal{W}_{t_0})$,..., $(\mathbb{W}_T, \mathcal{W}_T)$ are measurable spaces (Nature);
5. $f_t : (\mathbb{X}_t \times \mathbb{U}_t \times \mathbb{W}_t, \mathcal{X}_t \otimes \mathcal{U}_t \otimes \mathcal{W}_t) \to (\mathbb{X}_{t+1}, \mathcal{X}_{t+1})$ is a measurable mapping, for $t = t_0, \ldots, T-1$ (dynamics);
6. $\rho_{t-1:t} : \mathbb{X}_{t-1} \times \mathbb{U}_{t-1} \to \Delta(\mathbb{W}_t)$ is a stochastic kernel, for $t = t_0, \ldots, T-1$;
7. $L_t : \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1} \to \mathbb{R}$, for $t = t_0, \ldots, T-1$ and $K : \mathbb{X}_T \to \mathbb{R}$, measurable functions (instantaneous and final costs).

With these ingredients, Bertsekas and Shreve formulate a stochastic optimization problem with time additive additive cost function over given state spaces, action spaces and uncertainty spaces (note that state and action spaces are assumed to be of fixed sizes when time varies, thus a "state" is a priori given). They introduce the notion of history at time t which consists in the states and the actions prior to t and study optimization problems whose solutions (policies) are to be searched among history feedbacks (or relaxed history feedbacks), namely sequences of mappings $\mathbb{X}_{t_0} \times \prod_{s=t_0}^{t-1} (\mathbb{U}_s \times \mathbb{X}_{s+1}) \to \mathbb{U}_t$. They identify cases where no loss of optimality results from reducing the search to (relaxed) Markovian feedbacks $\mathbb{X}_t \to \mathbb{U}_t$. Then, they establish a dynamic programming equation, where the Bellman functions are function of the state $x_t \in \mathbb{X}_t$, and where the minimization is done over controls $u_t \in \mathbb{U}_t$. For finite horizon problems, the mathematical challenge is to set up a mathematical framework (the Borel assumptions) for which optimal policies exists.

The main objective of Bertsekas and Shreve is to state conditions under which the dynamic programming equation is mathematically sound, namely with universally measurable Bellman functions and with universally measurable relaxed control strategies in the context of Borel spaces. The interested reader will find all the subtleties about Borel spaces and universally measurable concepts in [23, Chapter 7].

Puterman Approach. The ingredients of the approach developed in [28] are the following:

1. time $t = t_0, t_0 + 1, \ldots, T-1, T$ is discrete, with integers $t_0 < T$;
2. $(\mathbb{X}_{t_0}, \mathcal{X}_{t_0})$, ..., $(\mathbb{X}_T, \mathcal{X}_T)$ are measurable spaces (state spaces);
3. $(\mathbb{U}_{t_0}, \mathcal{U}_{t_0})$,..., $(\mathbb{U}_{T-1}, \mathcal{U}_{T-1})$ are measurable spaces (decision spaces);
4. $\rho_{t-1:t} : \mathbb{X}_{t-1} \times \mathbb{U}_{t-1} \to \Delta(\mathbb{X}_t)$ is a stochastic kernel, for $t = t_0, \ldots, T-1$;
5. $L_t : \mathbb{X}_t \times \mathbb{U}_t \to \mathbb{R}$, for $t = t_0, \ldots, T-1$ and $K : \mathbb{X}_T \to \mathbb{R}$, measurable functions (instantaneous and final costs).

Puterman shares most of his ingredients with Bertsekas and Shreve, but he does not require uncertainty sets and dynamics, as he directly considers state transition stochastic kernels. With these ingredients, Puterman formulates a stochastic optimization problem, whose solutions are to be searched among history feedbacks, namely sequences of mappings $\mathbb{X}_{t_0} \times \prod_{s=t_0}^{t-1}(\mathbb{U}_s \times \mathbb{X}_{s+1}) \to \mathbb{U}_t$. Then, he establishes a dynamic programming equation, where the Bellman functions are function of the history $h_t \in \mathbb{X}_{t_0} \times \prod_{s=t_0}^{t-1}(\mathbb{U}_s \times \mathbb{X}_{s+1})$. He identifies cases where no loss of optimality results from reducing the search to Markovian feedbacks $\mathbb{X}_t \to \mathbb{U}_t$. In such cases, the Bellman functions are function of the state $x_t \in \mathbb{X}_t$, and the minimization in the dynamic programming equation is done over controls $u_t \in \mathbb{U}_t$.

The main objective of Puterman is to explore infinite horizon criteria, average reward criteria, the continuous time case, and to present many examples.

Approach in this Paper. The ingredients that we will use are the following:

1. time $t = t_0, t_0 + 1, \ldots, T-1, T$ is discrete, with integers $t_0 < T$;
2. $(\mathbb{U}_{t_0}, \mathcal{U}_{t_0})$,..., $(\mathbb{U}_{T-1}, \mathcal{U}_{T-1})$ are measurable spaces (decision spaces);
3. $(\mathbb{W}_{t_0}, \mathcal{W}_{t_0})$,..., $(\mathbb{W}_T, \mathcal{W}_T)$ are measurable spaces (Nature);
4. $\rho_{t-1:t} : \mathbb{W}_0 \times \prod_{s=0}^{t-1}(\mathbb{U}_s \times \mathbb{W}_{s+1}) \to \Delta(\mathbb{W}_t)$ is a stochastic kernel, for $t = t_0, \ldots, T-1$,
5. $j : (\mathbb{W}_0 \times \prod_{s=0}^{T-1}(\mathbb{U}_s \times \mathbb{W}_{s+1}), \mathcal{W}_0 \otimes \bigotimes_{s=0}^{T-1}(\mathcal{U}_s \otimes \mathcal{W}_{s+1})) \to \mathbb{R}$ is a measurable function (criterion).

The main features of the framework developed in this paper are the following: the history at time t consists of all uncertainties and actions prior to time t (rather than states and actions); the cost is a unique function depending on the whole history, from initial time t_0 to the horizon T; the probability distribution of uncertainty at time t depends on the history up to time $t-1$. We will state a dynamic programming equation, where the Bellman functions are function of the history $h_t \in \mathbb{W}_0 \times \prod_{s=0}^{t}(\mathbb{U}_s \times \mathbb{W}_{s+1})$ and where the minimization is done over controls $u_t \in \mathbb{U}_t$.

Our main objective is to establish a dynamic programming equation with a state, not at any time $t \in \{0, \ldots, T\}$, but at some specified instants $0 = t_0 < t_1 < \cdots < t_N = T$. The state spaces are not given a priori, but introduced a posteriori as image sets of history reduction mappings. With this, we can mix dynamic programming and stochastic programming.

Our framework is rather distant with the one of Evstigneev in [26]. It falls in the general framework developed by Witsenhausen (see [31] and [24, § 4.5.4]), *except* for the stochastic kernels (we are more general) and for the information structure (we are less general). Finally, our framework is closest to the one found in Bertsekas and Shreve [23] and Puterman [28], *except* for the state spaces, not given a priori, and for the criterion, function of the whole history.

1.2.2 Stochastic Dynamic Programming with History Feedbacks

We now present a framework that is adapted to both stochastic programming and stochastic dynamic programming. Time is discrete and runs among the integers $t = 0, 1, 2 \ldots, T-1, T$, where $T \in \mathbb{N}^*$. For $0 \leqslant r \leqslant s \leqslant T$, we introduce the interval $(r\,{:}\,s) = \{t \in \mathbb{N} \mid r \leqslant t \leqslant s\}$.

Histories and Feedbacks

We first define the basic and the composite spaces that we need to formulate multistage stochastic optimization problems. Then, we introduce a class of solutions called history feedbacks.

Histories and History Spaces. For each time $t = 0, 1, 2 \ldots, T-1$, the decision u_t takes its values in a measurable set $\mathbb{U}_t$ equipped with a σ-field $\mathcal{U}_t$. For each time $t = 0, 1, 2 \ldots, T$, the uncertainty w_t takes its values in a measurable set $\mathbb{W}_t$ equipped with a σ-field $\mathcal{W}_t$.

For $t = 0, 1, 2 \ldots, T$, we define the *history space* $\mathbb{H}_t$ equipped with the *history field* $\mathcal{H}_t$ by

$$\mathbb{H}_t = \mathbb{W}_0 \times \prod_{s=0}^{t-1} (\mathbb{U}_s \times \mathbb{W}_{s+1}) \text{ and } \mathcal{H}_t = \mathcal{W}_0 \otimes \bigotimes_{s=0}^{t-1} (\mathcal{U}_s \otimes \mathcal{W}_{s+1}) \;,\; t = 0, 1, 2 \ldots, T \;, \quad (1.1)$$

with the particular case $\mathbb{H}_0 = \mathbb{W}_0$, $\mathcal{H}_0 = \mathcal{W}_0$. A generic element $h_t \in \mathbb{H}_t$ is called a *history*:

$$h_t = (w_0, (u_s, w_{s+1})_{s=0,\ldots,t-1}) = (w_0, u_0, w_1, u_1, w_2, \ldots, u_{t-2}, w_{t-1}, u_{t-1}, w_t) \in \mathbb{H}_t \;.$$

For $1 \leqslant r \leqslant s \leqslant t$, we introduce the $(r\,{:}\,s)$-*history subpart*

$$h_{r:s} = (u_{r-1}, w_r, \ldots, u_{s-1}, w_s) \;,$$

so that we have $h_t = (h_{r-1}, h_{r:t})$.

History Feedbacks. When $0 \leqslant r \leqslant t \leqslant T-1$, we define a $(r\,{:}\,t)$-*history feedback* as a sequence $\{\gamma_s\}_{s=r,\ldots,t}$ of measurable mappings

$$\gamma_s : (\mathbb{H}_s, \mathcal{H}_s) \to (\mathbb{U}_s, \mathcal{U}_s) \;.$$

We call $\Gamma_{r:t}$ the set of $(r\,{:}\,t)$-history feedbacks.

The history feedbacks reflect the following information structure. At the end of the time interval $[t-1, t[$, an uncertainty variable w_t is produced. Then, at the beginning of the time interval $[t, t+1[$, a decision-maker takes a decision u_t, as follows

$$w_0 \rightsquigarrow u_0 \rightsquigarrow w_1 \rightsquigarrow u_1 \rightsquigarrow \quad \ldots \quad \rightsquigarrow w_{T-1} \rightsquigarrow u_{T-1} \rightsquigarrow w_T \;. \quad (1.2)$$

Optimization with Stochastic Kernels

We introduce a family of optimization problems with stochastic kernels. Then, we show how such problems can be solved by stochastic dynamic programming.

In what follows, we say that a function is *numerical* if it takes its values in $[-\infty, +\infty]$ (also called *extended* or *extended real-valued* function).

Family of Optimization Problems with Stochastic Kernels. To build a family of optimization problems over the time span $\{0, \dots, T-1\}$, we require two ingredients:

- a family $\{\rho_{s-1:s}\}_{1 \leqslant s \leqslant T}$ of stochastic kernels

$$\rho_{s-1:s} : (\mathbb{H}_{s-1}, \mathcal{H}_{s-1}) \to \Delta(\mathbb{W}_s) \ , \quad s = 1, \dots, T \ , \tag{1.3}$$

  that represents the distribution of the next uncertainty w_s parameterized by past history h_{s-1} (see the chronology in (1.2)),

- a numerical function, playing the role of a cost to be minimized,

$$j : (\mathbb{H}_T, \mathcal{H}_T) \to [0, +\infty] \ , \tag{1.4}$$

  assumed to be nonnegative[1] and measurable with respect to the field $\mathcal{H}_T$.

We define, for any $\{\gamma_s\}_{s=t,\dots,T-1} \in \Gamma_{t:T-1}$, a new family of stochastic kernels

$$\rho^{\gamma}_{t:T} : (\mathbb{H}_t, \mathcal{H}_t) \to \Delta(\mathbb{H}_T) \ ,$$

that capture the transitions between histories when the dynamics $h_{s+1} = \big(h_s, u_s, w_{s+1}\big)$ is driven by $u_s = \gamma_s(h_s)$ for $s = t, \dots, T-1$ (see Definition 11 in §1.6.2 for the detailed construction of $\rho^{\gamma}_{r:t}$; note that $\rho^{\gamma}_{t:T}$ generates a probability distribution on the space $\mathbb{H}_T$ of histories over the whole horizon $\{0, \dots, T\}$).

We consider the family of optimization problems, indexed by $t = 0, \dots, T-1$ and parameterized by the history $h_t \in \mathbb{H}_t$:

$$\inf_{\gamma_{t:T-1} \in \Gamma_{t:T-1}} \int_{\mathbb{H}_T} j(h'_T) \rho^{\gamma}_{t:T}(h_t, \mathrm{d}h'_T) \ , \quad \forall h_t \in \mathbb{H}_t \ , \tag{1.5}$$

the integral in the right-hand side of the above equation corresponding to the cost induced by the feedback $\gamma_{t:T-1}$ when starting at time t with a given history h_t. For all $t = 0, \dots, T-1$, we define the minimum value of Problem (1.5) by

$$V_t(h_t) = \inf_{\gamma_{t:T-1} \in \Gamma_{t:T-1}} \int_{\mathbb{H}_T} j(h'_T) \rho^{\gamma}_{t:T}(h_t, \mathrm{d}h'_T) \ , \quad \forall h_t \in \mathbb{H}_t \ , \tag{1.6a}$$

and we also define

$$V_T(h_T) = j(h_T) \ , \quad \forall h_T \in \mathbb{H}_T \ . \tag{1.6b}$$

The numerical function $V_t : \mathbb{H}_t \to [0, +\infty]$ is called the *value function* at time t.

[1] We could also consider any $j : \mathbb{H}_t \to \mathbb{R}$, measurable bounded function, or measurable and uniformly bounded below function. However, for the sake of simplicity, we will deal in the sequel with measurable nonnegative numerical functions. When $j(h_T) = +\infty$, this materializes joint constraints between uncertainties and controls.

Bellman Operators and Dynamic Programming. We show that the value functions in (1.6) are *Bellman functions*, in that they are solution of the Bellman or dynamic programming equation.

For $t = 0, \dots, T$, let $\mathbb{L}^0_+(\mathbb{H}_t, \mathcal{H}_t)$ be the space of universally measurable nonnegative numerical functions over $\mathbb{H}_t$ (see [23] for further details). For $t = 0, \dots, T-1$, we define the *Bellman operator* by, for all $\varphi \in \mathbb{L}^0_+(\mathbb{H}_{t+1}, \mathcal{H}_{t+1})$ and for all $h_t \in \mathbb{H}_t$,

$$\big(\mathcal{B}_{t+1:t}\varphi\big)(h_t) = \inf_{u_t \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \varphi(h_t, u_t, w_{t+1}) \rho_{t:t+1}(h_t, dw_{t+1}) \; . \tag{1.7}$$

Since $\varphi \in \mathbb{L}^0_+(\mathbb{H}_{t+1}, \mathcal{H}_{t+1})$, we have that $\mathcal{B}_{t+1:t}\varphi$ is a well defined nonnegative numerical function.

The proof of the following theorem is inspired by [23], and given in §1.6.3.

Theorem 1. *Assume that all the spaces introduced in §1.2.2 are Borel spaces, the stochastic kernels in (1.3) are Borel-measurable, and that the criterion j in (1.4) is a nonnegative lower semianalytic function.*

Then, the Bellman operators in (1.7) map $\mathbb{L}^0_+(\mathbb{H}_{t+1}, \mathcal{H}_{t+1})$ into $\mathbb{L}^0_+(\mathbb{H}_t, \mathcal{H}_t)$

$$\mathcal{B}_{t+1:t} : \mathbb{L}^0_+(\mathbb{H}_{t+1}, \mathcal{H}_{t+1}) \to \mathbb{L}^0_+(\mathbb{H}_t, \mathcal{H}_t) \; ,$$

and the value functions V_t defined in (1.6) are universally measurable and satisfy the Bellman equation*, or* (stochastic) dynamic programming equation*,*

$$V_T = j \; , \tag{1.8a}$$

$$V_t = \mathcal{B}_{t+1:t} V_{t+1} \; , \quad \textit{for } t = T-1, \dots, 1, 0 \; . \tag{1.8b}$$

This theorem is mainly inspired by [23], with the feature that the state x_t is in our case the history h_t, with the dynamics:

$$h_{t+1} = \big(h_t, u_t, w_{t+1}\big) \; . \tag{1.9}$$

This very general dynamic programming result will be the basis of all future developments in this paper. In the sequel, we assume that all the assumptions of Theorem 1 are fulfilled, that is,

- all the spaces (like the ones introduced in §1.2.2) will be supposed to be Borel spaces,
- all the stochastic kernels (like the ones introduced in (1.3)) will be supposed to be Borel-measurable,
- all the criteria (like the one introduced in (1.4)) will be supposed to be nonnegative lower semianalytic functions.

1.3 State Reduction by Time Blocks and Dynamic Programming

In this section, we consider the question of reducing the history using a compressed "state" variable. Differing with traditional practice, such a variable may be not available at any time $t \in \{0, \dots, T\}$, but at some specified instants $0 = t_0 < t_1 < \cdots < t_N = T$. We have see in the previous section that the history h_t is itself a canonical state variable in our framework with associated dynamics (1.9). However the size of this canonical state increases with t, which is a nasty feature for dynamic programming.

1.3.1 State Reduction on a Single Time Block

We first present the case where the reduction only occurs at two instants denoted by r and t:

$$0 \leqslant r < t \leqslant T \; .$$

Definition 2. *Let $(\mathbb{X}_r, \mathcal{X}_r)$ and $(\mathbb{X}_t, \mathcal{X}_t)$ be two measurable* state spaces*, θ_r and θ_t be two measurable* reduction mappings

$$\theta_r : \mathbb{H}_r \to \mathbb{X}_r \; , \quad \theta_t : \mathbb{H}_t \to \mathbb{X}_t \; , \tag{1.10a}$$

and $f_{r:t}$ be a measurable dynamics

$$f_{r:t} : \mathbb{X}_r \times \mathbb{H}_{r+1:t} \to \mathbb{X}_t \; . \tag{1.10b}$$

The triplet $(\theta_r, \theta_t, f_{r:t})$ is called a state reduction across $(r\,{:}\,t)$ *if we have*

$$\theta_t\big((h_r, h_{r+1:t})\big) = f_{r:t}\big(\theta_r(h_r), h_{r+1:t}\big) \; , \quad \forall h_t \in \mathbb{H}_t \; . \tag{1.10c}$$

The state reduction $(\theta_r, \theta_t, f_{r:t})$ is said to be compatible *with the family $\{\rho_{s-1:s}\}_{r+1 \leqslant s \leqslant t}$ of stochastic kernels (1.3) if*

- *there exists a* reduced stochastic kernel

$$\widetilde{\rho}_{r:r+1} : \mathbb{X}_r \to \Delta(\mathbb{W}_{r+1}) \; , \tag{1.11a}$$

  such that the stochastic kernel $\rho_{r:r+1}$ in (1.3) can be factored as

$$\rho_{r:r+1}(h_r, \mathrm{d}w_{r+1}) = \widetilde{\rho}_{r:r+1}\big(\theta_r(h_r), \mathrm{d}w_{r+1}\big) \; , \quad \forall h_r \in \mathbb{H}_r \; , \tag{1.11b}$$

- *for all $s = r+2, \dots, t$, there exists a* reduced stochastic kernel

$$\widetilde{\rho}_{s-1:s} : \mathbb{X}_r \times \mathbb{H}_{r+1:s-1} \to \Delta(\mathbb{W}_s) \; , \tag{1.11c}$$

  such that the stochastic kernel $\rho_{s-1:s}$ can be factored as

$$\rho_{s-1:s}\big((h_r, h_{r+1:s-1}), \mathrm{d}w_s\big) = \widetilde{\rho}_{s-1:s}\Big(\big(\theta_r(h_r), h_{r+1:s-1}\big), \mathrm{d}w_s\Big) \; , \quad \forall h_{s-1} \in \mathbb{H}_{s-1} \; . \tag{1.11d}$$

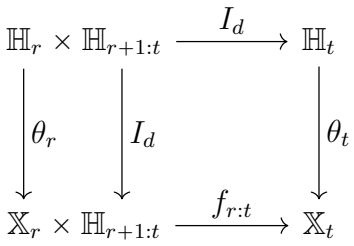

Figure 1.1: Commutative diagram in case of state reduction $(\theta_r, \theta_t, f_{r:t})$

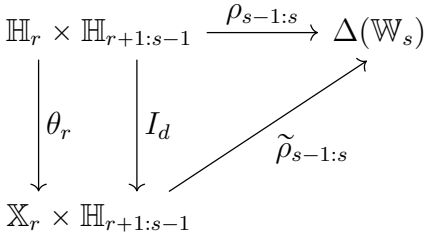

Figure 1.2: Commutative diagram in case of state reduction $(\theta_r, \theta_t, f_{r:t})$ compatible with the family $\{\rho_{s-1:s}\}_{r+1 \leqslant s \leqslant t}$

According to this definition, the triplet $(\theta_r, \theta_t, f_{r:t})$ is a state reduction across $(r\,{:}\,t)$ if and only if the diagram in Figure 1.1 is commutative; it is compatible if and only if the diagram in Figure 1.2 is commutative.

We define the *Bellman operator across* $(t\,{:}\,r)$ $\mathcal{B}_{t:r} : \mathbb{L}^0_+(\mathbb{H}_t, \mathcal{H}_t) \to \mathbb{L}^0_+(\mathbb{H}_r, \mathcal{H}_r)$ by

$$\mathcal{B}_{t:r} = \mathcal{B}_{r+1:r} \circ \cdots \circ \mathcal{B}_{t:t-1} \ , \tag{1.12}$$

where the one time step operators $\mathcal{B}_{s:s-1}$, for $r+1 \leqslant s \leqslant t$ are defined in (1.7).

The following proposition, whose proof is given in §1.6.3, is the key ingredient to formulate dynamic programming equations with a reduced state.

Proposition 3. *Suppose that there exists a state reduction $(\theta_r, \theta_t, f_{r:t})$ that is compatible with the family $\{\rho_{s-1:s}\}_{r+1 \leqslant s \leqslant t}$ of stochastic kernels (1.3) (see Definition 2). Then, there exists a* reduced Bellman operator across $(t\,{:}\,r)$

$$\widetilde{\mathcal{B}}_{t:r} : \mathbb{L}^0_+(\mathbb{X}_t, \mathcal{X}_t) \to \mathbb{L}^0_+(\mathbb{X}_r, \mathcal{X}_r) \ , \tag{1.13}$$

such that, for all $\widetilde{\varphi}_t \in \mathbb{L}^0_+(\mathbb{X}_t, \mathcal{X}_t)$, we have that

$$\big(\widetilde{\mathcal{B}}_{t:r}\widetilde{\varphi}_t\big) \circ \theta_r = \mathcal{B}_{t:r}(\widetilde{\varphi}_t \circ \theta_t) \ . \tag{1.14}$$

For all measurable nonnegative numerical function $\widetilde{\varphi}_t : \mathbb{X}_t \to [0, +\infty]$ and for all $x_r \in \mathbb{X}_r$,

we have that

$$\big(\widetilde{\mathcal{B}}_{t:r}\widetilde{\varphi}_t\big)(x_r) = \inf_{u_r\in\mathbb{U}_r}\int_{\mathbb{W}_{r+1}}\widetilde{\rho}_{r:r+1}(x_r,\mathrm{d}w_{r+1})$$
$$\inf_{u_{r+1}\in\mathbb{U}_{r+1}}\int_{\mathbb{W}_{r+2}}\widetilde{\rho}_{r+1:r+2}(x_r,u_r,w_{r+1},\mathrm{d}w_{r+2})\dots$$
$$\inf_{u_{t-1}\in\mathbb{U}_{t-1}}\int_{\mathbb{W}_t}\widetilde{\varphi}_t\big(f_{r:t}(x_r,u_r,w_{r+1},\dots,u_{t-1},w_t)\big)$$
$$\widetilde{\rho}_{t-1:t}(x_r,u_r,w_{r+1},\dots,u_{t-2},w_{t-1},\mathrm{d}w_t)\ . \quad (1.15)$$

Proposition 3 can be interpreted as follows. Denoting by $\theta_t^\star : \mathbb{L}_+^0(\mathbb{X}_t,\mathcal{X}_t)\to\mathbb{L}_+^0(\mathbb{H}_t,\mathcal{H}_t)$ the operator defined by

$$\theta_t^\star(\widetilde{\varphi}_t) = \widetilde{\varphi}_t\circ\theta_t\ ,\ \ \forall\widetilde{\varphi}_t\in\mathbb{L}_+^0(\mathbb{X}_t,\mathcal{X}_t)\ ,$$

the relation (1.14) rewrites

$$\theta_r^\star\circ\widetilde{\mathcal{B}}_{t:r} = \mathcal{B}_{t:r}\circ\theta_t^\star\ ,$$

that is, Proposition 3 states that the diagram in Figure 1.3 is commutative.

$$\begin{array}{ccc}
\mathbb{L}_+^0(\mathbb{H}_t,\mathcal{H}_t) & \xrightarrow{\mathcal{B}_{t:r}} & \mathbb{L}_+^0(\mathbb{H}_r,\mathcal{H}_r)\\
\uparrow{\scriptstyle\theta_t^\star} & & \uparrow{\scriptstyle\theta_r^\star}\\
\mathbb{L}_+^0(\mathbb{X}_t,\mathcal{X}_t) & \xrightarrow{\widetilde{\mathcal{B}}_{t:r}} & \mathbb{L}_+^0(\mathbb{X}_r,\mathcal{X}_r)
\end{array}$$

Figure 1.3: Commutative diagram for Bellman operators in case of a compatible state reduction $(\theta_r,\theta_t,f_{r:t})$

1.3.2 State Reduction on Multiple Consecutive Time Blocks and Dynamic Programming Equations

Proposition 3 can easily be extended to the case of multiple consecutive time blocks $[t_i,t_{i+1}]$, $i=0,\dots,N-1$, where

$$0 = t_0 < t_1 < \dots < t_N = T\ . \quad (1.16)$$

Definition 4. *Let $\{(\mathbb{X}_{t_i},\mathcal{X}_{t_i})\}_{i=0,\dots,N}$ be a family of measurable state spaces, $\{\theta_{t_i}\}_{i=0,\dots,N}$ be a family of measurable reduction mappings $\theta_{t_i}:\mathbb{H}_{t_i}\to\mathbb{X}_{t_i}$, and $\big\{f_{t_i:t_{i+1}}\big\}_{i=0,\dots,N-1}$ be a family of measurable dynamics $f_{t_i:t_{i+1}}:\mathbb{X}_{t_i}\times\mathbb{H}_{t_i+1:t_{i+1}}\to\mathbb{X}_{t_{i+1}}$.*

The triplet $(\{\mathbb{X}_{t_i}\}_{i=0,\dots,N}, \{\theta_{t_i}\}_{i=0,\dots,N}, \{f_{t_i:t_{i+1}}\}_{i=0,\dots,N-1})$ *is called a* state reduction across the consecutive time blocks $[t_i, t_{i+1}]$, $i = 0, \dots, N-1$ *if every triplet* $(\theta_{t_i}, \theta_{t_{i+1}}, f_{t_i:t_{i+1}})$ *is a state reduction, for* $i = 0, \dots, N-1$.

The state reduction across the consecutive time blocks $[t_i, t_{i+1}]$ *is said to be* compatible *with the family* $\{\rho_{s-1:s}\}_{1 \leqslant s \leqslant T}$ *of stochastic kernels given in (1.3) if every triplet* $(\theta_{t_i}, \theta_{t_{i+1}}, f_{t_i:t_{i+1}})$ *is compatible with the family* $\{\rho_{s-1:s}\}_{t_i+1 \leqslant s \leqslant t_{i+1}}$, *for* $i = 0, \dots, N-1$.

Assuming the existence of a state reduction across the consecutive time blocks $[t_i, t_{i+1}]$ compatible with the family of stochastic kernels (1.3), we obtain the existence of a *family of reduced Bellman operators* across the consecutive $(t_{i+1}:t_i)$ as an immediate consequence of multiple applications of Proposition 3, that is,

$$\widetilde{\mathcal{B}}_{t_{i+1}:t_i} : \mathbb{L}^0_+(\mathbb{X}_{t_{i+1}}, \mathcal{X}_{t_{i+1}}) \to \mathbb{L}^0_+(\mathbb{X}_{t_i}, \mathcal{X}_{t_i}) \ , \quad i = 0, \dots, N-1 \ ,$$

such that, for any function $\widetilde{\varphi}_{t_{i+1}} \in \mathbb{L}^0_+(\mathbb{X}_{t_{i+1}}, \mathcal{X}_{t_{i+1}})$, we have that

$$\big(\widetilde{\mathcal{B}}_{t_{i+1}:t_i} \widetilde{\varphi}_{t_{i+1}}\big) \circ \theta_{t_i} = \mathcal{B}_{t_{i+1}:t_i}(\widetilde{\varphi}_{t_{i+1}} \circ \theta_{t_{i+1}}) \ .$$

We now consider the family of optimization problems (1.5) and the associated value functions (1.6). Thanks to the state reductions, we are able to state the following theorem which establishes dynamic programming equations *across* consecutive time blocks. Its proof is an immediate consequence of multiple applications of Theorem 1 and Proposition 3.

Theorem 5. *Suppose that a state reduction* $(\{\mathbb{X}_{t_i}\}_{i=0,\dots,N}, \{\theta_{t_i}\}_{i=0,\dots,N}, \{f_{t_i:t_{i+1}}\}_{i=0,\dots,N-1})$ *exists across the consecutive time blocks* $[t_i, t_{i+1}]$, $i = 0, \dots, N-1$ *as in (1.16), that is compatible with the family* $\{\rho_{s-1:s}\}_{1 \leqslant s \leqslant T}$ *of stochastic kernels given in (1.3).*

Assume that there exists a reduced criterion

$$\widetilde{j} : \mathbb{X}_T \to [0, +\infty] \ ,$$

such that the cost function j *in (1.4) can be factored as*

$$j = \widetilde{j} \circ \theta_{t_N} \ .$$

We define the family of reduced value functions $\{\widetilde{V}_{t_i}\}_{i=0,\dots,N}$ *by*

$$\widetilde{V}_{t_N} = \widetilde{j} \ , \tag{1.18a}$$

$$\widetilde{V}_{t_i} = \widetilde{\mathcal{B}}_{t_{i+1}:t_i} \widetilde{V}_{t_{i+1}} \ , \quad \text{for } i = N-1, \dots, 0 \ . \tag{1.18b}$$

Then, the family $\{V_{t_i}\}_{i=0,\dots,N}$ *in (1.6) satisfies*

$$V_{t_i} = \widetilde{V}_{t_i} \circ \theta_{t_i} \ , \quad i = 0, \dots, N \ . \tag{1.18c}$$

To obtain such a dynamic programming equation across time blocks, we needed the detour of Sect. 1.2, with a dynamic programming equation over the history space. Thus equipped, it is now possible to propose a decomposition scheme for optimization problems with multiple time scales, using both stochastic programming and stochastic dynamic programming. We detail applications of this scheme in Sect. 1.4.

1.4 Applications of Time Blocks Dynamic Programming

We present in this section two applications of the state reduction result stated in Theorem 5.

The first one corresponds to a *two time-scales* optimization problem. A typical instance of such a problem is to optimize long-term investment decisions (slow time-scale) — for example the renewal of batteries in an energy system — but the optimal long-term decisions highly depend on short-term operating decisions (fast time-scale) — for example the way the battery is operated in real-time.

The second application corresponds to a class of stochastic multistage optimization problems arising often in practice, especially when managing stocks (dams for instance). The decision-maker takes two decisions at each time step t: at the beginning of the time interval $[t, t+1[$, the first decision (quantity of water to be turbinated to produce electricity for instance) is taken without knowing the uncertainty that will occur during the time step (decision-hazard framework); at the end of the time interval $[t, t+1[$, an uncertainty variable w_{t+1} is produced and the second decision (quantity of water to be released to avoid dam overflow for instance) is taken once the uncertainty at time step t is revealed (hazard-decision framework). This new class of problems is called *decison-hazard-decision* optimization problems.

1.4.1 Two Time-Scales Multistage Optimization Problems

In this class of problems, each time index t is represented by a couple (d, m) of indices, with $d \in \{0, \dots, D+1\}$ and $m \in \{0, \dots, M\}$: we can think of the index d as an index of days (slow time-scale), and m as an index of minutes (fast time-scale). The corresponding set of time indices is thus

$$\mathbb{T} = \{0, \dots, D\} \times \{0, \dots, M\} \cup \{(D+1, 0)\} \, . \tag{1.19}$$

At the end of every minute $m-1$ of every day d, that is, at the end of the time interval $\big[(d, m-1), (d, m)\big)$, $0 \leqslant d \leqslant D$ and $1 \leqslant m \leqslant M$, an uncertainty variable $w_{d,m}$ becomes available. Then, at the beginning of the minute m, a decision-maker takes a decision $u_{d,m}$. Moreover, at the beginning of every day d, an uncertainty variable $w_{d,0}$ is produced, followed by a decision $u_{d,0}$. The interplay between uncertainties and decision is thus as follows (compare the chronology with the one in (1.2)):

$$\begin{aligned} w_{0,0} \rightsquigarrow u_{0,0} \rightsquigarrow w_{0,1} \rightsquigarrow u_{0,1} \rightsquigarrow \cdots \\ \cdots \rightsquigarrow w_{0,M-1} \rightsquigarrow u_{0,M-1} \rightsquigarrow w_{0,M} \rightsquigarrow u_{0,M} \rightsquigarrow w_{1,0} \rightsquigarrow u_{1,0} \rightsquigarrow w_{1,1} \cdots \\ \cdots \rightsquigarrow w_{D,M} \rightsquigarrow u_{D,M} \rightsquigarrow w_{D+1,0} \, . \end{aligned}$$

We assume that a state reduction (as in Definition 4) is available at the beginning of each day d, so that it becomes possible to write dynamic programming equations by time blocks as stated by Theorem 5. Such state reductions will be for example available when the noises of the different days are stochastically independent.

We present the mathematical formalism to handle such type of problems. In this application, the difficulty to apply Theorem 5 is mainly notational.

Time Span. We consider the set $\mathbb{T}$ equipped with the *lexicographical order*

$$(0,0) < (0,1) < \cdots < (d,M) < (d+1,0) < \cdots < (D,M-1) < (D,M) < (D+1,0) \; . \tag{1.20a}$$

The set $\mathbb{T}$ of couples in (1.19) is in one to one correspondence with the (linear) *time span* $\{0,\dots,T\}$, where

$$T = (D+1) \times (M+1) + 1 \; , \tag{1.20b}$$

by the *lexicographic mapping* τ

$$\tau : \{0,\dots,T\} \to \mathbb{T} \tag{1.20c}$$

$$t \mapsto \tau(t) = (d,m) \; . \tag{1.20d}$$

In the sequel, we will denote by $(d,m) \in \mathbb{T}$ the element of $\{0,\dots,T\}$ given by $\tau^{-1}(d,m) = d \times (M+1) + m$:

$$\mathbb{T} \ni (d,m) \leftrightarrow \tau^{-1}(d,m) = d \times (M+1) + m \in \{0,\dots,T\} \; . \tag{1.20e}$$

For $(d,m) \leqslant (d',m')$, as ordered by the lexicographical order (1.20a), we introduce the time interval $((d,m):(d',m')) = \{(d'',m'') \in \mathbb{T} \mid (d,m) \leqslant (d'',m'') \leqslant (d',m')\}$.

History Spaces. For all $(d,m) \in \{0,\dots,D\} \times \{0,\dots,M\}$, the decision $u_{d,m}$ takes its values in a measurable set $\mathbb{U}_{d,m}$ equipped with a σ-field $\mathcal{U}_{d,m}$. For all $(d,m) \in \{0,\dots,D\} \times \{0,\dots,M\} \cup \{(D+1,0)\}$, the uncertainty $w_{d,m}$ takes its values in a measurable set $\mathbb{W}_{d,m}$ equipped with a σ-field $\mathcal{W}_{d,m}$.

With the identification (1.20e), for all $(d,m) \in \mathbb{T}$, we define the *history space* $\mathbb{H}_{(d,m)}$

$$\mathbb{H}_{(d,m)} = \mathbb{W}_{0,0} \times \mathbb{U}_{0,0} \times \mathbb{W}_{0,1} \times \cdots \times \mathbb{U}_{d,m-1} \times \mathbb{W}_{d,m} \; , \tag{1.21a}$$

equipped with the *history field* $\mathcal{H}_{(d,m)}$ as in (1.1). For all $d \in \{0,\dots,D+1\}$, we define the *slow scale history* h_d element of the *slow scale history space* $\mathbb{H}_d$

$$h_d = h_{(d,0)} \in \mathbb{H}_d = \mathbb{H}_{(d,0)} \; , \tag{1.21b}$$

equipped with the *slow scale history field* $\mathcal{H}_d = \mathcal{H}_{(d,0)}$. For all $d \in \{1,\dots,D\}$, we define the *slow scale partial history space* $\mathbb{H}_{d:d+1}$

$$\mathbb{H}_{d:d+1} = \mathbb{H}_{(d,1):(d+1,0)} = \mathbb{U}_{d,0} \times \mathbb{W}_{d,1} \times \cdots \times \mathbb{U}_{d,M-1} \times \mathbb{W}_{d,M} \times \mathbb{U}_{d,M} \times \mathbb{W}_{d+1,0} \; , \tag{1.21c}$$

equipped with the associated *slow scale partial history field* $\mathcal{H}_{d:d+1}$, the case $d = 0$ being

$$\mathbb{H}_{0:1} = \mathbb{H}_{(1,0)} = \mathbb{W}_{0,0} \times \mathbb{U}_{0,0} \times \mathbb{W}_{0,1} \times \cdots \times \mathbb{U}_{0,M-1} \times \mathbb{W}_{0,M} \times \mathbb{U}_{0,M} \times \mathbb{W}_{1,0} \; . \tag{1.21d}$$

Stochastic Kernels. Because of the jump from one day to the next, we introduce two families of stochastic kernels[2]:

- a family $\left\{\rho_{(d,M):(d+1,0)}\right\}_{0 \leqslant d \leqslant D}$ of stochastic kernels *across* consecutive slow scale steps

$$\rho_{(d,M):(d+1,0)} : \mathbb{H}_{(d,M)} \to \Delta(\mathbb{W}_{d+1,0}) \; , \;\; d = 0,\dots,D \; , \tag{1.22a}$$

- a family $\left\{\rho_{(d,m-1):(d,m)}\right\}_{0 \leqslant d \leqslant D, 1 \leqslant m \leqslant M}$ of stochastic kernels *within* consecutive slow scale steps

$$\rho_{(d,m-1):(d,m)} : \mathbb{H}_{(d,m-1)} \to \Delta(\mathbb{W}_{d,m}) \; , \;\; d = 0,\dots,D \; , \;\; m = 1,\dots,M \; . \tag{1.22b}$$

[2]These families are defined over the time span $\{0,\dots,T\} \equiv \mathbb{T}$ by the identification (1.20e) in such a way that the notation is consistent with the notation (1.3).

History Feedbacks. A history feedback at index $(d,m) \in \mathbb{T}$ is a measurable mapping

$$\gamma_{(d,m)} : \mathbb{H}_{(d,m)} \to \mathbb{U}_{(d,m)} \ .$$

For $(d,m) \leqslant (d',m')$, as ordered by the lexicographical order (1.20a), we denote by $\Gamma_{(d,m):(d',m')}$ the set of $((d,m):(d',m'))$-history feedbacks.

Slow Scale Value Functions. We suppose given a nonnegative numerical function

$$j : \mathbb{H}_{D+1} \to [0,+\infty] \ , \tag{1.23}$$

assumed to be measurable with respect to the field $\mathcal{H}_{D+1}$ associated to $\mathbb{H}_{D+1}$.

For $d = 0, \ldots, D$, we build the new stochastic kernels $\rho^{\gamma}_{(d,0):(D+1,0)} : \mathbb{H}_d \to \Delta(\mathbb{H}_{D+1})$ (see Definition 11 in §1.6.2 for their construction), and we define the *slow scale value functions*

$$V_d(h_d) = \inf_{\gamma \in \Gamma_{(d,0):(D,M)}} \int_{\mathbb{H}_{D+1}} j(h'_{D+1}) \rho^{\gamma}_{(d,0):(D+1,0)}(h_d, \mathrm{d}h'_{D+1}) \ , \quad \forall h_d \in \mathbb{H}_d \ , \tag{1.24a}$$

$$V_{D+1} = j \ . \tag{1.24b}$$

For $d = 0, \ldots, D$, we define a *family of slow scale Bellman operators across* $(d+1:d)$

$$\mathcal{B}_{d+1:d} : \mathbb{L}^0_+(\mathbb{H}_{d+1}, \mathcal{H}_{d+1}) \to \mathbb{L}^0_+(\mathbb{H}_d, \mathcal{H}_d) \ , \quad d = 0, \ldots, D \ , \tag{1.25a}$$

by

$$\mathcal{B}_{d+1:d} = \mathcal{B}_{(d+1,0):(d,0)} = \mathcal{B}_{(d,1):(d,0)} \circ \ldots \circ \mathcal{B}_{(d,M):(d,M-1)} \circ \mathcal{B}_{(d+1,0):(d,M)} \ . \tag{1.25b}$$

Then, applying repeatedly Theorem 1 leads to the fact that the family $\{V_d\}_{d=0,\ldots,D+1}$ of slow scale value functions (1.24) satisfies

$$V_{D+1} = j \ , \tag{1.26a}$$

$$V_d = \mathcal{B}_{d+1:d} V_{d+1} \ , \quad \text{for } d = D, D-1, \ldots, 0 \ . \tag{1.26b}$$

Compatible State Reductions. We now rewrite Definition 4 in the context of the two time-scales problem.

Definition 6 (Compatible slow scale reduction)**.** *Let* $\{(\mathbb{X}_d, \mathcal{X}_d)\}_{d=0,\ldots,D+1}$ *be a family of measurable state spaces,* $\{\theta_d\}_{d=0,\ldots,D+1}$ *be family of measurable reduction mappings such that*

$$\theta_d : \mathbb{H}_d \to \mathbb{X}_d \ ,$$

and $\{f_{d:d+1}\}_{d=0,\ldots,D}$ *be a family of measurable dynamics such that*

$$f_{d:d+1} : \mathbb{X}_d \times \mathbb{H}_{d:d+1} \to \mathbb{X}_{d+1} \ .$$

The triplet $\big(\{\mathbb{X}_d\}_{d=0,\dots,D+1}, \{\theta_d\}_{d=0,\dots,D+1}, \{f_{d:d+1}\}_{d=0,\dots,D}\big)$ *is said to be a* slow scale state reduction *if for all* $d = 0, \dots, D$

$$\theta_{d+1}\big((h_d, h_{d:d+1})\big) = f_{d:d+1}\big(\theta_d(h_d), h_{d:d+1}\big) \ , \quad \forall (h_d, h_{d:d+1}) \in \mathbb{H}_{d+1} \ .$$

The slow scale state reduction $\big(\{\mathbb{X}_d\}_{d=0,\dots,D+1}, \{\theta_d\}_{d=0,\dots,D+1}, \{f_{d:d+1}\}_{d=0,\dots,D}\big)$ *is said to be* compatible with the two families $\big\{\rho_{(d,M):(d+1,0)}\big\}_{0\leqslant d\leqslant D}$ and $\big\{\rho_{(d,m-1):(d,m)}\big\}_{0\leqslant d\leqslant D, 1\leqslant m\leqslant M}$ of stochastic kernels *defined in (1.22a)–(1.22b) if for any* $d = 0, \dots, D$, *we have that*

- *there exists a* reduced stochastic kernel

$$\widetilde{\rho}_{(d,M):(d+1,0)} : \mathbb{X}_d \times \mathbb{H}_{(d,0):(d,M)} \to \Delta(\mathbb{W}_{d+1,0}) \ ,$$

  such that the stochastic kernel $\rho_{(d,M):(d+1,0)}$ *in (1.22a) can be factored as*

$$\rho_{(d,M):(d+1,0)}(h_{d,M}, \mathrm{d}w_{d+1,0}) = \widetilde{\rho}_{(d,M):(d+1,0)}\big(\theta_d(h_d), h_{(d,0):(d,M)}, \mathrm{d}w_{d+1,0}\big) \ , \quad \forall h_{d,M} \in \mathbb{H}_{(d,M)} \ ,$$

- *for each* $m = 1, \dots, M$ *, there exists a* reduced stochastic kernel

$$\widetilde{\rho}_{(d,m-1):(d,m)} : \mathbb{X}_d \times \mathbb{H}_{(d,0):(d,m-1)} \to \Delta(\mathbb{W}_{d,m}) \ ,$$

  such that the stochastic kernel $\rho_{(d,m-1):(d,m)}$ *in (1.22b) can be factored as*

$$\rho_{(d,m-1):(d,m)}(h_{d,m-1}, \mathrm{d}w_{d,m}) = \widetilde{\rho}_{(d,m-1):(d,m)}\big(\theta_d(h_d), h_{(d,0):(d,m-1)}, \mathrm{d}w_{d,m}\big) \ , \quad \forall h_{d,m-1} \in \mathbb{H}_{(d,m-1)} \ .$$

Dynamic Programming Equations. Using the reduced stochastic kernels of Definition 6, we apply Proposition 3 and obtain a *family of slow scale reduced Bellman operators across* $(d+1:d)$

$$\widetilde{\mathcal{B}}_{d+1:d} : \mathbb{L}^0_+(\mathbb{X}_{d+1}, \mathcal{X}_{d+1}) \to \mathbb{L}^0_+(\mathbb{X}_d, \mathcal{X}_d) \ , \quad d = 0, \dots, D \ . \tag{1.29}$$

We are now able to state the main result of this section.

Theorem 7. *Assume that there exists a compatible slow scale state reduction* $\big(\{\mathbb{X}_d\}_{d=0,\dots,D+1}, \{\theta_d\}_{d=0,\dots,D+1}, \{f_{d:d+1}\}_{d=0,\dots,D}\big)$ *and that there exists a* reduced criterion

$$\widetilde{j} : \mathbb{X}_{D+1} \to [0, +\infty] \ ,$$

such that the cost function j *in (1.23) can be factored as*

$$j = \widetilde{j} \circ \theta_{D+1} \ .$$

We define the family of reduced value functions $\{\widetilde{V}_d\}_{d=0,\dots,D+1}$ *by*

$$\widetilde{V}_{D+1} = \widetilde{j} \ , \tag{1.31a}$$

$$\widetilde{V}_d = \widetilde{\mathcal{B}}_{d+1:d}\widetilde{V}_{d+1} \ , \quad \textit{for } d = D, \dots, 0 \ . \tag{1.31b}$$

Then, the family $\{V_d\}_{d=0,\dots,D+1}$ *of slow scale value functions (1.24) satisfies*

$$V_d = \widetilde{V}_d \circ \theta_d \ , \quad d = 0, \dots, D \ . \tag{1.31c}$$

Proof. Since the triplet $(\{\mathbb{X}_d\}_{d=0,\dots,D+1}, \{\theta_d\}_{d=0,\dots,D+1}, \{f_{d:d+1}\}_{d=0,\dots,D})$ is a state reduction across the time blocks $[(d,0),(d{+}1,0)]$, which is compatible with the family $\{\rho_{(d,0):(d+1,0)}\}_{0\leqslant d\leqslant D}$ of stochastic kernels, the proof is an immediate consequence of Theorem 5. □

Thanks to Theorem 7, we are able to replace the optimization problem formulated on the whole time set $\mathbb{T}$ by a sequence of D optimization subproblems formulated each on a single time block $[(d,0),(d{+}1,0)]$. Moreover, the numerical burden of the method remains reasonable provided that the dimensions of the spaces $\mathbb{X}_d$ remain small, thus avoiding the curse of dimensionality. This is the benefit induced by *dynamic programming* which makes possible a time decomposition of the problem. However, to make the method operational, we need to compute the functions $\widetilde{V}_d$, whose expression is available thanks to Proposition 3:

$$\begin{aligned}\widetilde{V}_d(x_d) = \inf_{u_{d,0}\in\mathbb{U}_{d,0}} \int_{\mathbb{W}_{d,1}} \widetilde{\rho}_{(d,0):(d,1)}(x_d, \mathrm{d}w_{d,1}) \dots \\ \inf_{u_{d,M-1}\in\mathbb{U}_{d,M-1}} \int_{\mathbb{W}_{d,M}} \widetilde{\rho}_{(d,M-1):(d,M)}(x_d, u_{d,0}, w_{d,1}, \cdots, w_{d,M-1}, \mathrm{d}w_{d,M}) \\ \inf_{u_{d,M}\in\mathbb{U}_{d,M}} \int_{\mathbb{W}_{d+1,0}} \widetilde{V}_{d+1}\big(\widetilde{f}_{d:d+1}(x_d, u_{d,0}, w_{d,1}, \cdots, u_{d,M-1}, w_{d,M}, u_{d,M}, w_{d+1,0})\big) \\ \widetilde{\rho}_{(d,M):(d+1,0)}(x_d, u_{d,0}, w_{d,1}, \cdots, w_{d,M}, \mathrm{d}w_{d+1,0})\,.\end{aligned} \tag{1.32}$$

In many practical situations, this computation is tractable by using *stochastic programming*. For example, if the stochastic kernels $\widetilde{\rho}_{(d,m):(d,m+1)}$ do not depend on the past controls $(u_{d,0},\cdots,u_{d,m-1})$, then it is possible to approximate the optimization problem (1.32) by using scenario tree techniques. Note that these last techniques do not require stagewise independence of the noises. We are thus able to take advantage of both the dynamic programming world and the stochastic programming world:

- use dynamic programming at slow time scale across consecutive slow time steps, when the slow time scale noises are supposed to be stochastically independent; produce slow time scale Bellman functions;
- use stochastic programming at short time scale, within two consecutive slow time steps; the final short time scale cost is given by the slow time scale Bellman functions; no stagewise independence assumption is required for the short time scale noises.

1.4.2 Decision-Hazard-Decision Optimization Problems

We apply the reduction by time blocks to the so-called *decision-hazard-decision* dynamic programming.

Motivation for the Decision-Hazard-Decision Framework

We illustrate our motivation with a single dam management problem. We can model the dynamics of the water volume in a dam by

$$S_{t+1} = \min\{S^\sharp, S_t - q_t + a_{t+1}\} \ , \tag{1.33}$$

where $t = t_0, t_0 + 1, \ldots, T - 1$ and

- $S^\sharp$ is the maximal dam volume,
- S_t is the volume (stock) of water at the beginning of period $[t, t + 1[$,
- a_{t+1} is the inflow water volume (rain, etc.) during $[t, t + 1[$,
- q_t is the turbined outflow volume during $[t, t + 1[$ (control variable),
  - decided at the *beginning* of period $[t, t + 1[$,
  - chosen such that $0 \leqslant q_t \leqslant S_t$,
  - supposed to depend on the stock S_t but not on the inflow water a_{t+1}.

The min operation in Equation (1.33) ensures that the dam volume always remains below its maximal capacity, but induces a non linearity in the dynamics.

Alternatively, we can model the dynamics of the water volume in a dam by

$$S_{t+1} = S_t - q_t - a_{t+1} - r_{t+1} \ , \tag{1.34}$$

where $t = t_0, t_0 + 1, \ldots, T - 1$ and

- r_{t+1} is the spilled volume
  - decided at the *end* of period $[t, t + 1[$,
  - supposed to depend on the stock S_t and on the inflow water a_{t+1},
  - and chosen such that $0 \leqslant S_t - q_t + a_{t+1} - r_{t+1} \leqslant S^\sharp$.

Thus, with the formulation (1.34), we pay the price to add one control r_{t+1}, but we obtain a linear model instead of the nonlinear model (1.33). This is especially interesting when using the stochastic dual dynamic programming (SDDP), for which the linearity of the dynamics is used to obtain the convexity properties required by the algorithm.

Decision-Hazard-Decision Framework

We consider stochastic optimization problems where, during the time interval between two time steps, the decision-maker takes two decisions. At the end of the time interval $[s - 1, s[$, an uncertainty variable $w_s^\flat$ is produced, and then, at the beginning of the time interval $[s, s + 1[$, the decision-maker takes a *head decision* $u_s^\sharp$. What is new is that, at the end of the time interval $[s, s + 1[$, when an uncertainty variable $w_{s+1}^\flat$ is produced, the decision-maker has the possibility to make a *tail decision* $u_{s+1}^\flat$. This latter decision $u_{s+1}^\flat$ can be thought as a *recourse* variable for a two stage stochastic optimization problem that would take place inside the time interval $[s, s + 1[$. We call $w_0^\sharp$ the uncertainty happening

right before the first decision. The interplay between uncertainties and decisions is thus as follows (compare the chronology with the one in (1.2)):

$$w_0^\sharp \rightsquigarrow u_0^\sharp \rightsquigarrow w_1^\flat \rightsquigarrow u_1^\flat \rightsquigarrow u_1^\sharp \rightsquigarrow w_2^\flat \rightsquigarrow \quad \dots \quad \rightsquigarrow w_{S-1}^\flat \rightsquigarrow u_{S-1}^\flat \rightsquigarrow u_{S-1}^\sharp \rightsquigarrow w_S^\flat \rightsquigarrow u_S^\flat \ .$$

Let $S \in \mathbb{N}^*$. For each time $s = 0, 1, 2 \dots, S-1$, the *head decision* $u_s^\sharp$ takes values in a measurable set $\mathbb{U}_s^\sharp$, equipped with a σ-field $\mathcal{U}_s^\sharp$. For each time $s = 1, 2 \dots, S$, the *tail decision* $u_s^\flat$ takes values in measurable set $\mathbb{U}_s^\flat$, equipped with a σ-field $\mathcal{U}_s^\flat$. For each time $s = 1, 2 \dots, S$, the uncertainty $w_s^\flat$ takes its values in a measurable set $\mathbb{W}_s^\flat$, equipped with a σ-field $\mathcal{W}_s^\flat$. For time $s = 0$, the uncertainty $w_0^\sharp$ takes its values in a measurable set $\mathbb{W}_0^\sharp$, equipped with a σ-field $\mathcal{W}_0^\sharp$.

Again, in this application, the difficulty to apply Theorem 5 is mainly notational.

History Spaces. For $s = 0, 1, 2 \dots, S$, we define the *head history space*

$$\mathbb{H}_s^\sharp = \mathbb{W}_0^\sharp \times \prod_{s'=0}^{s-1} \left(\mathbb{U}_{s'}^\sharp \times \mathbb{W}_{s'+1}^\flat \times \mathbb{U}_{s'+1}^\flat \right) , \tag{1.35a}$$

and its associated *head history field* $\mathcal{H}_s^\sharp$. We also define, for $s = 1, 2 \dots, S$, the *tail history space*

$$\mathbb{H}_s^\flat = \mathbb{H}_{s-1}^\sharp \times \mathbb{U}_{s-1}^\sharp \times \mathbb{W}_s^\flat , \tag{1.35b}$$

and its associated *tail history field* $\mathcal{H}_s^\flat$.

Stochastic Kernels. We introduce a family of stochastic kernels $\{\rho_{s-1:s}\}_{1 \leqslant s \leqslant S}$, with

$$\rho_{s-1:s} : \mathbb{H}_{s-1}^\sharp \to \Delta(\mathbb{W}_s^\flat) \ . \tag{1.36}$$

History Feedbacks. For $s = 0, \dots, S-1$, a *head history feedback* at time s is a measurable mapping

$$\gamma_s^\sharp : \mathbb{H}_s^\sharp \to \mathbb{U}_s^\sharp \ .$$

We call $\Gamma_s^\sharp$ the *set of head history feedbacks at time* s, and we define $\Gamma_{s:S}^\sharp = \Gamma_s^\sharp \times \cdots \times \Gamma_S^\sharp$. We also define, for all $s = 1, 2 \dots, S$, a *tail history feedback* at time s as a measurable mapping

$$\gamma_s^\flat : \mathbb{H}_s^\flat \to \mathbb{U}_s^\flat \ .$$

We call $\Gamma_s^\flat$ the *set of tail history feedbacks at time* s, and we define $\Gamma_{s:S}^\flat = \Gamma_s^\flat \times \cdots \times \Gamma_S^\flat$.

Value Functions. We consider a nonnegative numerical function

$$j : \mathbb{H}_S^\sharp \to [0, +\infty] , \tag{1.38}$$

assumed to be measurable with respect to the head history field $\mathcal{H}_S^\sharp$.

For $s = 0, \dots, S$, we define *value functions* by

$$V_s(h_s^\sharp) = \inf_{\gamma^\sharp \in \Gamma_{s:S-1}^\sharp, \gamma^\flat \in \Gamma_{s+1:S}^\sharp} \int_{\mathbb{H}_S^\sharp} j(h'_S) \rho_{s:S}^{\gamma^\sharp, \gamma^\flat}(h_s^\sharp, \mathrm{d}h'_S) \ , \quad \forall h_s^\sharp \in \mathbb{H}_s^\sharp \ , \tag{1.39}$$

where $\rho_{s:S}^{\gamma^\sharp,\gamma^\flat}$ has to be understood as $\rho_{s:S}^{\gamma}$ (see Definition 11), with

$$\gamma_s(h_s^\sharp) = \gamma_s^\sharp(h_s^\sharp) \ , \ \ \forall h_s^\sharp \in \mathbb{H}_s^\sharp \ , \tag{1.40a}$$

$$\gamma_{s'}(h_{s'}^\flat) = \Big(\gamma_{s'}^\flat(h_{s'}^\flat), \gamma_{s'}^\sharp\big(h_{s'}^\flat, \gamma_{s'}^\flat(h_{s'}^\flat)\big)\Big) \ , \ \ \forall s' = s+1, \ldots, S-1 \ , \ \ \forall h_{s'}^\flat \in \mathbb{H}_{s'}^\flat \ , \tag{1.40b}$$

$$\gamma_S(h_S^\flat) = \gamma_S^\flat(h_S^\flat) \ , \ \ \forall h_S^\flat \in \mathbb{H}_S^\flat \ . \tag{1.40c}$$

The following proposition, whose proof has been relegated in 1.6.3, characterizes the dynamic programming equations in the decision-hazard-decision framework.

Proposition 8. *For $s = 0, \ldots, S-1$, we define the* Bellman operator

$$\mathcal{B}_{s+1:s} : \mathbb{L}_+^0(\mathbb{H}_{s+1}^\sharp, \mathcal{H}_{s+1}^\sharp) \to \mathbb{L}_+^0(\mathbb{H}_s^\sharp, \mathcal{H}_s^\sharp) \tag{1.41a}$$

such that, for all $\varphi \in \mathbb{L}_+^0(\mathbb{H}_{s+1}^\sharp, \mathcal{H}_{s+1}^\sharp)$ and for all $h_s^\sharp \in \mathbb{H}_s^\sharp$,

$$\big(\mathcal{B}_{s+1:s}\varphi\big)(h_s^\sharp) = \inf_{u_s^\sharp \in \mathbb{U}_s^\sharp} \int_{\mathbb{W}_{s+1}^\flat} \Big(\inf_{u_{s+1}^\flat \in \mathbb{U}_{s+1}^\flat} \varphi(h_s^\sharp, u_s^\sharp, w_{s+1}^\flat, u_{s+1}^\flat) \Big) \rho_{s:s+1}(h_s^\sharp, dw_{s+1}^\flat) \ . \tag{1.41b}$$

Then the value functions (1.39) satisfy

$$V_S = j \ , \tag{1.41c}$$

$$V_s = \mathcal{B}_{s+1:s} V_{s+1} \ , \ \ \forall s = 0, \ldots, S-1 \ . \tag{1.41d}$$

Compatible State Reductions. We now rewrite Definition 4 in the context of a decision-hazard-decision problem.

Definition 9 (Compatible state reduction)**.** *Let $\{\mathbb{X}_s\}_{s=0,\ldots,S}$ be a family of state spaces, $\{\theta_s\}_{s=0,\ldots,S}$ be a family of measurable reduction mappings such that*

$$\theta_s : \mathbb{H}_s^\sharp \to \mathbb{X}_s \ ,$$

and $\{f_{s:s+1}\}_{s=0,\ldots,S-1}$ be a family of measurable dynamics such that

$$f_{s:s+1} : \mathbb{X}_s \times \mathbb{U}_s^\sharp \times \mathbb{W}_{s+1} \times \mathbb{U}_{s+1}^\flat \to \mathbb{X}_{s+1} \ .$$

The triplet $\big(\{\mathbb{X}_s\}_{s=0,\ldots,S}, \{\theta_s\}_{s=0,\ldots,S}, \{f_{s:s+1}\}_{s=0,\ldots,S-1}\big)$ is said to be a decision-hazard-decision state reduction *if, for all $s = 0, \ldots, S-1$, we have that*

$$\theta_{s+1}\big((h_s, u_s^\sharp, w_{s+1}, u_{s+1}^\flat)\big) = f_{s:s+1}\big(\theta_s(h_s), u_s^\sharp, w_{s+1}, u_{s+1}^\flat\big) \ , \\ \forall (h_s, u_s^\sharp, w_{s+1}, u_{s+1}^\flat) \in \mathbb{H}_s^\sharp \times \mathbb{U}_s^\sharp \times \mathbb{W}_{s+1} \times \mathbb{U}_{s+1}^\flat \ .$$

The decision-hazard-decision state reduction is said to be compatible with the family $\{\rho_{s:s+1}\}_{0 \leqslant s \leqslant S-1}$ of stochastic kernels *in (1.36) if there exists a family $\{\widetilde{\rho}_{s:s+1}\}_{0 \leqslant s \leqslant S-1}$ of* reduced stochastic kernels

$$\widetilde{\rho}_{s:s+1} : \mathbb{X}_s \to \Delta(\mathbb{W}_{s+1}) \ ,$$

such that, for each $s = 0, \ldots, S-1$, the stochastic kernel $\rho_{s:s+1}$ in (1.36) can be factored as

$$\rho_{s:s+1}(h_s^\sharp, \mathrm{d}w_{s+1}) = \widetilde{\rho}_{s:s+1}\big(\theta_s(h_s^\sharp), \mathrm{d}w_{s+1}\big) \ , \ \ \forall h_s^\sharp \in \mathbb{H}_s^\sharp \ .$$

Dynamic Programming Equations. We state the main result of this section.

Theorem 10. *Assume that there exists a decision-hazard-decision state reduction* $\big(\{\mathbb{X}_s\}_{s=0,\dots,S}, \{\theta_s\}_{s=0,\dots,S}, \{f_{s:s+1}\}_{s=0,\dots,S-1}\big)$ *and that there exists a* reduced criterion

$$\widetilde{j} : \mathbb{X}_S \to [0, +\infty] ,$$

such that the cost function j in (1.38) can be factored as

$$j = \widetilde{j} \circ \theta_S .$$

We define a family of reduced Bellman operators across $(s+1:s)$

$$\widetilde{\mathcal{B}}_{s+1:s} : \mathbb{L}^0_+(\mathbb{X}_{s+1}, \mathcal{X}_{s+1}) \to \mathbb{L}^0_+(\mathbb{X}_s, \mathcal{X}_s) , \quad s = 1, \dots, S-1 , \tag{1.45a}$$

by, for any measurable function $\widetilde{\varphi} : \mathbb{X}_{s+1} \to [0, +\infty]$,

$$(\widetilde{\mathcal{B}}_{s+1:s}\widetilde{\varphi})(x_s) = \inf_{u^\sharp_s \in \mathbb{U}^\sharp_s} \int_{\mathbb{W}_{s+1}} \Big(\inf_{u^\flat_{s+1} \in \mathbb{U}^\flat_{s+1}} \widetilde{\varphi}\big(f_{s:s+1}(x_s, u^\sharp_s, w_{s+1}, u^\flat_{s+1})\big)\Big) \widetilde{\rho}_{s:s+1}(x_s, dw_{s+1}) . \tag{1.45b}$$

**We define the family of* reduced value functions $\{\widetilde{V}_s\}_{s=0,\dots,S}$ *by*

$$\widetilde{V}_S = \widetilde{j} \tag{1.46a}$$

$$\widetilde{V}_s = \widetilde{\mathcal{B}}_{s+1:s}\widetilde{V}_{s+1} \quad \textit{for } s = S-1, \dots, 0 . \tag{1.46b}$$

Then, the value functions V_s defined by (1.39) satisfy

$$V_s = \widetilde{V}_s \circ \theta_s , \quad s = 0, \dots, S . \tag{1.47}$$

Proof. It has been shown in the proof of Proposition 8 that the setting of a decision-hazard-decision problem was a particular kind of two time-scales problem. The proof of the theorem is then a direct application of Theorem 7. □

Theorem 10 allows to develop dynamic programming equations in the decision-hazard-decision framework. Such equations can be solved using the stochastic dual dynamic programming (SDDP) algorithm provided that convexity of the value functions is preserved. This requires linearity in the dynamics, a feature that may be recovered by modeling the problem in the decision-hazard-decision framework as illustrated in §1.4.2.

1.5 Conclusion and Perspectives

As said in the introduction, decomposition methods are appealing to tackle multistage stochastic optimization problems, as they are naturally large scale. The most common approaches are time decomposition (and state-based resolution methods, like stochastic dynamic programming, in stochastic optimal control), and scenario decomposition (like progressive hedging in stochastic programming). One also finds space decomposition methods [21].

This paper is part of a general research program that consists in *mixing* different decomposition bricks. Here, we tackled the issue of mixing time decomposition (stochastic dynamic programming) with scenario decomposition. For this purpose, we have revisited the notion of state, and have provided a way to perform time decomposition but only accross specified time blocks. Inside a time block, one can then use stochastic programming methods, like scenario decomposition. Our time blocks decomposition scheme is especially adapted to multi time-scales stochastic optimization problems. In this vein, we have shown its application to two time-scales and to the novel class of decision-hazard-decision problems.

We are currently working on how to mix time decomposition (stochastic dynamic programming) with space/units decomposition.

Acknowledgements. We thank Roger Wets for the fruitful discussions about the possibility of mixing stochastic dynamic programming with progressive hedging. We thank an anonymous reviewer for challenging our first version of the paper: the current version has been restructured according to his remarks.

1.6 Technical Details and Proofs

In this section, we provide technical details, constructions and proofs of results in the paper.

1.6.1 Histories, Feedbacks and Flows

We introduce the notations

$$\mathbb{W}_{r:t} = \prod_{s=r}^{t} \mathbb{W}_s \ , \ \ 0 \leqslant r \leqslant t \leqslant T \tag{1.48a}$$

$$\mathbb{U}_{r:t} = \prod_{s=r}^{t} \mathbb{U}_s \ , \ \ 0 \leqslant r \leqslant t \leqslant T-1 \tag{1.48b}$$

$$\mathbb{H}_{r:t} = \prod_{s=r-1}^{t-1} (\mathbb{U}_s \times \mathbb{W}_{s+1}) = \mathbb{U}_{r-1} \times \mathbb{W}_r \times \cdots \times \mathbb{U}_{t-1} \times \mathbb{W}_t \ , \ \ 1 \leqslant r \leqslant t \leqslant T \ . \tag{1.48c}$$

Let $0 \leqslant r \leqslant s \leqslant t \leqslant T$. From a history $h_t \in \mathbb{H}_t$, we can extract the $(r:s)$-*history uncertainty part*

$$[h_t]^{\mathbb{W}}_{r:s} = (w_r, \ldots, w_s) = w_{r:s} \in \mathbb{W}_{r:s} \ , \ \ 0 \leqslant r \leqslant s \leqslant t \ , \tag{1.49a}$$

the $(r:s)$-*history control part* (notice that the indices are special)

$$[h_t]^{\mathbb{U}}_{r:s} = (u_{r-1}, \ldots, u_{s-1}) = u_{r-1:s-1} \in \mathbb{U}_{r-1:s-1} \ , \ \ 1 \leqslant r \leqslant s \leqslant t \ , \tag{1.49b}$$

and the $(r:s)$-*history subpart*

$$[h_t]_{r:s} = (u_{r-1}, w_r, \ldots, u_{s-1}, w_s) = h_{r:s} \in \mathbb{H}_{r:s} \ , \ \ 1 \leqslant r \leqslant s \leqslant t \ , \tag{1.49c}$$

so that we obtain, for $0 \leqslant r+1 \leqslant s \leqslant t$,

$$h_t = (\underbrace{w_0, u_0, w_1, \ldots, u_{r-1}, w_r}_{h_r}, \underbrace{u_r, w_{r+1}, \ldots, u_{t-2}, w_{t-1}, u_{t-1}, w_t}_{h_{r+1:t}}) = (h_r, h_{r+1:t}) \ . \tag{1.49d}$$

Flows. Let r and t be given such that $0 \leqslant r < t \leqslant T$. For a $(r\!:\!t-1)$-history feedback $\gamma = \{\gamma_s\}_{s=r,\dots,t-1} \in \Gamma_{r:t-1}$, we define the *flow* $\Phi^\gamma_{r:t}$ by

$$\Phi^\gamma_{r:t} : \mathbb{H}_r \times \mathbb{W}_{r+1:t} \to \mathbb{H}_t \tag{1.50a}$$

$$(h_r, w_{r+1:t}) \mapsto (h_r, \gamma_r(h_r), w_{r+1}, \gamma_{r+1}(h_r, \gamma_r(h_r), w_{r+1}), w_{r+2}, \cdots, \gamma_{t-1}(h_{t-1}), w_t) \ , \tag{1.50b}$$

that is,

$$\Phi^\gamma_{r:t}(h_r, w_{r+1:t}) = (h_r, u_r, w_{r+1}, u_{r+1}, w_{r+2}, \dots, u_{t-1}, w_t) \ , \tag{1.50c}$$

$$\text{with } \ h_s = (h_r, u_r, w_{r+1}, \dots, u_{s-1}, w_s) \ , \ \ r < s \leqslant t \ , \tag{1.50d}$$

$$\text{and } \ u_s = \gamma_s(h_s) \ , \ \ r < s \leqslant t-1 \ . \tag{1.50e}$$

When $0 \leqslant r = t \leqslant T$, we put

$$\Phi^\gamma_{r:r} : \mathbb{H}_r \to \mathbb{H}_r \ , \ \ h_r \mapsto h_r \ . \tag{1.50f}$$

With this convention, the expression $\Phi^\gamma_{r:t}$ makes sense when $0 \leqslant r \leqslant t \leqslant T$: when $r = t$, no $(r\!:\!r-1)$-history feedback exists, but none is needed. The mapping $\Phi^\gamma_{r:t}$ gives the history at time t as a function of the initial history h_r at time r and of the history feedbacks $\{\gamma_s\}_{s=r,\dots,t-1} \in \Gamma_{r:t-1}$. An immediate consequence of this definition are the *flow properties*:

$$\Phi^\gamma_{r:t+1}(h_r, w_{r+1:t+1}) = \Big(\Phi^\gamma_{r:t}(h_r, w_{r+1:t}), \gamma_t\big(\Phi^\gamma_{r:t}(h_r, w_{r+1:t})\big), w_{t+1}\Big) \ , \ \ 0 \leqslant r \leqslant t \leqslant T-1 \ , \tag{1.51a}$$

$$\Phi^\gamma_{r:t}(h_r, w_{r+1:t}) = \Phi^\gamma_{r+1:t}\big((h_r, \gamma_r(h_r), w_{r+1}), w_{r+2:t}\big) \ , \ \ 0 \leqslant r < t \leqslant T \ . \tag{1.51b}$$

1.6.2 Building Stochastic Kernels from History Feedbacks

Definition 11. *Let r and t be given such that $0 \leqslant r \leqslant t \leqslant T$.*

- *When $0 \leqslant r < t \leqslant T$, for*

  1. *a $(r\!:\!t-1)$-history feedback $\gamma = \{\gamma_s\}_{s=r,\dots,t-1} \in \Gamma_{r:t-1}$,*
  2. *a family $\{\rho_{s-1:s}\}_{r+1 \leqslant s \leqslant t}$ of stochastic kernels*

  $$\rho_{s-1:s} : \mathbb{H}_{s-1} \to \Delta(\mathbb{W}_s) \ , \ \ s = r+1, \dots, t \ ,$$

  we define a stochastic kernel

  $$\rho^\gamma_{r:t} : \mathbb{H}_r \to \Delta(\mathbb{H}_t) \tag{1.52a}$$

by, for any $\varphi : \mathbb{H}_t \to [0, +\infty]$*, measurable nonnegative numerical function, that is,* $\varphi \in \mathbb{L}^0_+(\mathbb{H}_t, \mathcal{H}_t)$*,* [3]

$$\int_{\mathbb{H}_t} \varphi(h'_r, h'_{r+1:t}) \rho^\gamma_{r:t}(h_r, \mathrm{d}h'_t) = \int_{\mathbb{W}_{r+1:t}} \varphi\big(\Phi^\gamma_{r:t}(h_r, w_{r+1:t})\big) \prod_{s=r+1}^{t} \rho_{s-1:s}\big(\Phi^\gamma_{r:s-1}(h_r, w_{r+1:s-1}), \mathrm{d}w_s\big) \; . \tag{1.52b}$$

- *When* $0 \leqslant r = t \leqslant T$*, we define*

$$\rho^\gamma_{r:r} : \mathbb{H}_r \to \Delta(\mathbb{H}_r) \; , \;\; \rho^\gamma_{r:r}(h_r, \mathrm{d}h'_r) = \delta_{h_r}(\mathrm{d}h'_r) \; . \tag{1.52c}$$

The stochastic kernels $\rho^\gamma_{r:t}$ on $\mathbb{H}_t$, given by (1.52), are of the form

$$\rho^\gamma_{r:t}(h_r, \mathrm{d}h'_t) = \rho^\gamma_{r:t}(h_r, \mathrm{d}h'_r \mathrm{d}h'_{r+1:t}) = \delta_{h_r}(\mathrm{d}h'_r) \otimes \varrho^\gamma_{r:t}(h_r, \mathrm{d}h'_{r+1:t}) \; , \tag{1.53}$$

where, for each $h_r \in \mathbb{H}_r$, the probability distribution $\varrho^\gamma_{r:t}(h_r, \mathrm{d}h'_{r+1:t})$ only charges the histories visited by the flow from $r+1$ to t. The construction of the stochastic kernels $\rho^\gamma_{r:t}$ is developed in [23, p. 190] for relaxed history feedbacks and obtained by using [23, Proposition 7.45].

Proposition 12. *Following Definition 11, we can define a family* $\{\rho^\gamma_{s:t}\}_{r \leqslant s \leqslant t}$ *of stochastic kernels. This family has the flow property, that is, for* $s < t$*,*

$$\rho^\gamma_{s:t}(h_s, \mathrm{d}h'_t) = \int_{\mathbb{W}_{s+1}} \rho_{s:s+1}\big(h_s, \mathrm{d}w_{s+1}\big) \rho^\gamma_{s+1:t}\Big(\big(h_s, \gamma_s(h_s), w_{s+1}\big), \mathrm{d}h'_t\Big) \; . \tag{1.54}$$

Proof. Let $s < t$. For any $\varphi : \mathbb{H}_t \to [0, +\infty]$, we have that

$$\int_{\mathbb{H}_t} \varphi(h'_s, h'_{s+1:t}) \rho^\gamma_{s:t}(h_s, \mathrm{d}h'_t) \tag{1.55a}$$

$$= \int_{\mathbb{W}_{s+1:t}} \varphi\big(\Phi^\gamma_{s:t}(h_s, w_{s+1:t})\big) \prod_{s'=s+1}^{t} \rho_{s'-1:s'}\big(\Phi^\gamma_{s:s'-1}(h_s, w_{s+1:s'-1}), \mathrm{d}w_{s'}\big)$$

by the definition (1.52b) of the stochastic kernel $\rho^\gamma_{s:t}$,

$$= \int_{\mathbb{W}_{s+1:t}} \varphi\big(\Phi^\gamma_{s:t}(h_s, w_{s+1:t})\big) \rho_{s:s+1}\big(h_s, \mathrm{d}w_{s+1}\big) \prod_{s'=s+2}^{t} \rho_{s'-1:s'}\big(\Phi^\gamma_{s:s'-1}(h_s, w_{s+1:s'-1}), \mathrm{d}w_{s'}\big)$$

[3] See Footnote 1.

by the property (1.50f) of the flow $\Phi^{\gamma}_{s:s}$,

$$= \int_{\mathbb{W}_{s+1:t}} \varphi\big(\Phi^{\gamma}_{s+1:t}\big((h_s, \gamma_s(h_s)), w_{s+1}), w_{s+2:t}\big)\big) \rho_{s:s+1}\big(h_s, \mathrm{d}w_{s+1}\big) \prod_{s'=s+2}^{t} \rho_{s'-1:s'}\big(\Phi^{\gamma}_{s+1:s'-1}\big((h_s, \gamma_s(h_s)), w_{s+1}), w_{s+2:s'-1}\big), \mathrm{d}w_{s'}\big)$$

by the flow property (1.51b),

$$= \int_{\mathbb{W}_{s+1}} \rho_{s:s+1}\big(h_s, \mathrm{d}w_{s+1}\big) \int_{\mathbb{W}_{s+2:t}} \varphi\big(\Phi^{\gamma}_{s+1:t}\big((h_s, \gamma_s(h_s)), w_{s+1}), w_{s+2:t}\big)\big) \prod_{s'=s+2}^{t} \rho_{s'-1:s'}\big(\Phi^{\gamma}_{s+1:s'-1}\big((h_s, \gamma_s(h_s)), w_{s+1}), w_{s+2:s'-1}\big), \mathrm{d}w_{s'}\big)$$

by Fubini Theorem [27, p.137],

$$= \int_{\mathbb{W}_{s+1}} \rho_{s:s+1}\big(h_s, \mathrm{d}w_{s+1}\big) \int_{\mathbb{H}_t} \varphi\big((h'_s, \gamma_s(h'_s), w'_{s+1}), h'_{s+2:t}\big) \rho^{\gamma}_{s+1:t}\big((h_s, \gamma_s(h_s)), w_{s+1}), \mathrm{d}h'_t\big)$$

by definition (1.52b) of $\rho^{\gamma}_{s+1:t}$,

$$= \int_{\mathbb{H}_t} \varphi\big((h'_s, \gamma_s(h'_s), w'_{s+1}), h'_{s+2:t}\big) \int_{\mathbb{W}_{s+1}} \rho_{s:s+1}\big(h_s, \mathrm{d}w_{s+1}\big) \rho^{\gamma}_{s+1:t}\big((h_s, \gamma_s(h_s)), w_{s+1}), \mathrm{d}h'_t\big) \tag{1.55b}$$

by Fubini Theorem and by definition (1.52b) of $\rho^{\gamma}_{s:t}$. As the two expressions (1.55a) and (1.55b) are equal for any $\varphi : \mathbb{H}_t \to [0, +\infty]$, we deduce the flow property (1.54). This ends the proof. □

1.6.3 Proofs

Proof of Theorem 1

Proof. We only give a sketch of the proof, as it is a variation on different results of [23], the framework of which we follow.

We take the history space $\mathbb{H}_t$ for state space, and the state dynamics

$$f\big(h_t, u_t, w_{t+1}\big) = \big(h_t, u_t, w_{t+1}\big) = h_{t+1} \in \mathbb{H}_{t+1} = \mathbb{H}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1} \,. \tag{1.56}$$

Then, the family $\{\rho_{s-1:s}\}_{1 \leqslant s \leqslant T}$ of stochastic kernels (1.3) gives a family of disturbance kernels that do not depend on the current control. The criterion to be minimized (1.4) is a function of the history at time T, thus of the state at time T. Problem (1.5) is thus a finite horizon model with a final cost and we are minimizing over the so-called state-feedbacks. Then, the proof of Theorem 1 follows from the results developed in Chap. 7, 8 and 10 of [23] in a Borel setting. Since we are considering a finite horizon model with a final cost, we detail the steps needed to use the results of [23, Chap. 8].

The final cost at time T can be turned into an instantaneous cost at time $T-1$ by inserting the state dynamics (1.56) in the final cost. Getting rid of the disturbance in the expected cost by using the disturbance kernel is standard practice. Then, we can turn this non-homogeneous finite horizon model into a finite horizon model with homogeneous dynamics and costs by following the steps of [23, Chap. 10]. Using [23, Proposition 8.2], we obtain that the family of optimization problems (1.5), when minimizing over the relaxed state feedbacks, satisfies the Bellman equation (1.8); we conclude with [23, Proposition 8.4] which covers the minimization over state feedbacks.

To summarize, Theorem 1 is valid under the general Borel assumptions of [23, Chap. 8] and with the specific (F^-) assumption needed for [23, Proposition 8.4]; this last assumption is fulfilled here since we have assumed that the criterion (1.4) is nonnegative. □

Proof of Proposition 3

Proof. Let $\widetilde{\varphi}_t : \mathbb{X}_t \to [0, +\infty]$ be a given measurable nonnegative numerical function, and let $\varphi_t : \mathbb{H}_t \to [0, +\infty]$ be

$$\varphi_t = \widetilde{\varphi}_t \circ \theta_t \ . \tag{1.57}$$

Let $\varphi_r : \mathbb{H}_r \to [0, +\infty]$ be the measurable nonnegative numerical function obtained by applying the Bellman operator $\mathcal{B}_{t:r}$ across $(t:r)$ (see (1.12)) to the measurable nonnegative numerical function φ_t:

$$\varphi_r = \mathcal{B}_{t:r}\varphi_t = \mathcal{B}_{r+1:r} \circ \cdots \circ \mathcal{B}_{t:t-1}\varphi_t \ . \tag{1.58}$$

We will show that there exists a measurable nonnegative numerical function

$$\widetilde{\varphi}_r : \mathbb{X}_r \to [0, +\infty]$$

such that

$$\varphi_r = \widetilde{\varphi}_r \circ \theta_r \ . \tag{1.59}$$

First, we show by backward induction that, for all $s \in \{r, \dots, t\}$, there exists a measurable nonnegative numerical function $\overline{\varphi}_s$ such that $\varphi_s(h_s) = \overline{\varphi}_s(\theta_r(h_r), h_{r+1:s})$. Second, we prove that the function $\widetilde{\varphi}_r = \overline{\varphi}_r$ satisfies (1.59).

- For $s = t$, we have, by (1.57) and by (1.10c), that

  $$\varphi_t(h_t) = \widetilde{\varphi}_t\big(\theta_t(h_t)\big) = \widetilde{\varphi}_t\big(f_{r:t}(\theta_r(h_r), h_{r+1:t})\big) \ ,$$

  so that the measurable nonnegative numerical function $\overline{\varphi}_t$ is given by $\widetilde{\varphi}_t \circ f_{r:t}$.

- Assume that, at $s+1$, the result holds true, that is,

  $$\varphi_{s+1}(h_{s+1}) = \overline{\varphi}_{s+1}(\theta_r(h_r), h_{r+1:s+1}) \ . \tag{1.60}$$

  Then, by (1.58),

  $$\begin{aligned}\varphi_s(h_s) &= \big(\mathcal{B}_{s+1:s}\varphi_{s+1}\big)(h_s) \\ &= \inf_{u_s \in \mathbb{U}_s} \int_{\mathbb{W}_{s+1}} \varphi_{s+1}\big((h_s, u_s, w_{s+1})\big)\rho_{s:s+1}(h_s, \mathrm{d}w_{s+1})\end{aligned}$$

by definition (1.7) of the Bellman operator

$$= \inf_{u_s \in \mathbb{U}_s} \int_{\mathbb{W}_{s+1}} \overline{\varphi}_{s+1}\big((\theta_r(h_r), (h_{r+1:s}, u_s, w_{s+1}))\big) \rho_{s:s+1}(h_s, \mathrm{d}w_{s+1})$$

by induction assumption (1.60)

$$= \inf_{u_s \in \mathbb{U}_s} \int_{\mathbb{W}_{s+1}} \overline{\varphi}_{s+1}\big((\theta_r(h_r), (h_{r+1:s}, u_s, w_{s+1}))\big) \widetilde{\rho}_{s:s+1}\big((\theta_r(h_r), h_{r+1:s}), \mathrm{d}w_{s+1}\big)$$

by compatibility (1.11) of the stochastic kernel

$$= \overline{\varphi}_s\big(\theta_r(h_r), h_{r+1:s}\big) \ ,$$

where

$$\overline{\varphi}_s\big(x_r, h_{r+1:s}\big) = \inf_{u_s \in \mathbb{U}_s} \int_{\mathbb{W}_{s+1}} \overline{\varphi}_{s+1}\big((x_r, (h_{r+1:s}, u_s, w_{s+1}))\big) \widetilde{\rho}_{s:s+1}\big((x_r, h_{r+1:s}), \mathrm{d}w_{s+1}\big) \ .$$

The result thus holds true at time s.

The induction implies that, at time r, the expression of $\varphi_r(h_r)$ is

$$\varphi_r(h_r) = \overline{\varphi}_r\big(\theta_r(h_r)\big) \ ,$$

since the term $h_{r+1:r}$ vanishes. Choosing $\widetilde{\varphi}_r = \overline{\varphi}_r$ gives the expected result. □

Proof of Proposition 8

Proof. We now show that the setting in §1.4.2 is a particular kind of two time scales problem as seen in §1.4.1. For this purpose, we introduce a *spurious uncertainty variable* $w_s^\sharp$ taking values in a singleton set $\mathbb{W}_s^\sharp = \{\overline{w}_s^\sharp\}$, equipped with the trivial σ-field $\{\varnothing, \mathbb{W}_s^\sharp\}$, for each time $s = 1, 2 \ldots, S$. Now, we obtain the following sequence of events:

$$w_0^\sharp \rightsquigarrow u_0^\sharp \rightsquigarrow w_1^\flat \rightsquigarrow u_1^\flat \rightsquigarrow w_1^\sharp \rightsquigarrow u_1^\sharp \rightsquigarrow w_2^\flat \rightsquigarrow u_2^\flat \rightsquigarrow w_2^\sharp \rightsquigarrow u_2^\sharp \rightsquigarrow \quad \ldots$$
$$\rightsquigarrow w_{S-1}^\flat \rightsquigarrow u_{S-1}^\flat \rightsquigarrow w_{S-1}^\sharp \rightsquigarrow u_{S-1}^\sharp \rightsquigarrow w_S^\flat \rightsquigarrow u_S^\flat \rightsquigarrow w_S^\sharp \ ,$$

which coincides with a two time scales problem:

$$\underbrace{w_{0,0} = w_0^\sharp \rightsquigarrow u_{0,0} = u_0^\sharp \rightsquigarrow w_{0,1} = w_1^\flat \rightsquigarrow u_{0,1} = u_1^\flat}_{\text{slow time cycle}} \rightsquigarrow$$
$$\underbrace{w_{1,0} = w_1^\sharp \rightsquigarrow u_{1,0} = u_1^\sharp \rightsquigarrow w_{1,1} = w_2^\flat \rightsquigarrow u_{1,1} = u_2^\flat}_{\text{slow time cycle}} \rightsquigarrow$$
$$\cdots \rightsquigarrow \underbrace{w_{S-1,0} = w_{S-1}^\sharp \rightsquigarrow u_{S-1,0} = u_{S-1}^\sharp \rightsquigarrow w_{S-1,1} = w_S^\flat \rightsquigarrow u_{S-1,1} = u_S^\flat}_{\text{slow time cycle}} \rightsquigarrow w_{S,0} = w_S^\sharp \ .$$

We introduce the sets

$$\begin{aligned}
\mathbb{W}_{d,0} &= \mathbb{W}_d^\sharp, \text{ for } \ d \in \{0,\dots,S\},\\
\mathbb{W}_{d,1} &= \mathbb{W}_{d+1}^\flat, \text{ for } \ d \in \{0,\dots,S-1\},\\
\mathbb{U}_{d,0} &= \mathbb{U}_d^\sharp, \text{ for } \ d \in \{0,\dots,S-1\},\\
\mathbb{U}_{d,1} &= \mathbb{U}_{d+1}^\flat, \text{ for } \ d \in \{0,\dots,S-1\}.
\end{aligned}$$

As a consequence, we observe that the two time scales history spaces in §1.4.1 are in one to one correspondence with the decision-hazard-decision history spaces and fields in (1.35a)–(1.35b) as follows:

for $d = 0, 1, 2 \dots, S$,

$$\begin{aligned}
\mathbb{H}_{d,0} &= \mathbb{W}_0^\sharp \times \prod_{d'=0}^{d-1} \left(\mathbb{U}_{d',0} \times \mathbb{W}_{d',1} \times \mathbb{U}_{d',1} \times \mathbb{W}_{d'+1,0}\right)\\
&= \mathbb{W}_0^\sharp \times \prod_{d'=0}^{d-1} \left(\mathbb{U}_{d'}^\sharp \times \mathbb{W}_{d'+1}^\flat \times \mathbb{U}_{d'+1}^\flat \times \mathbb{W}_{d'+1}^\sharp\right)\\
&\equiv \mathbb{W}_0^\sharp \times \prod_{d'=0}^{d-1} \left(\mathbb{U}_{d'}^\sharp \times \mathbb{W}_{d'+1}^\flat \times \mathbb{U}_{d'+1}^\flat\right) = \mathbb{H}_d^\sharp \ ,
\end{aligned}$$

for $d = 0, 1, 2 \dots, S$,

$$\mathcal{H}_{d,0} = \mathcal{W}_0^\sharp \otimes \bigotimes_{d'=0}^{d-1} \left(\mathcal{U}_{d'}^\sharp \otimes \mathcal{W}_{d'+1}^\flat \otimes \mathcal{U}_{d'+1}^\flat \otimes \mathcal{W}_{d'+1}^\sharp\right) \ ,$$

for $d = 0, 1, 2 \dots, S-1$,

$$\begin{aligned}
\mathbb{H}_{d,1} &= \mathbb{W}_0^\sharp \times \prod_{d'=0}^{d-1} \left(\mathbb{U}_{d',0} \times \mathbb{W}_{d',1} \times \mathbb{U}_{d',1} \times \mathbb{W}_{d'+1,0}\right) \times \mathbb{U}_{d,0} \times \mathbb{W}_{d,1}\\
&= \mathbb{W}_0^\sharp \times \prod_{d'=0}^{d-1} \left(\mathbb{U}_{d'}^\sharp \times \mathbb{W}_{d'+1}^\flat \times \mathbb{U}_{d'+1}^\flat \times \mathbb{W}_{d'+1}^\sharp\right) \times \mathbb{U}_d^\sharp \times \mathbb{W}_{d+1}^\flat\\
&\equiv \mathbb{W}_0^\sharp \times \prod_{d'=0}^{d-1} \left(\mathbb{U}_{d'}^\sharp \times \mathbb{W}_{d'+1}^\flat \times \mathbb{U}_{d'+1}^\flat\right) \times \mathbb{U}_d^\sharp \times \mathbb{W}_{d+1}^\flat = \mathbb{H}_{d+1}^\flat \ ,
\end{aligned}$$

for $d = 0, 1, 2 \dots, S-1$,

$$\mathcal{H}_{d,1} = \mathcal{W}_0^\sharp \otimes \bigotimes_{d'=0}^{d-1} \left(\mathcal{U}_{d'}^\sharp \otimes \mathcal{W}_{d'+1}^\flat \otimes \mathcal{U}_{d'+1}^\flat \otimes \mathcal{W}_{d'+1}^\sharp\right) \otimes \mathcal{U}_d^\sharp \otimes \mathcal{W}_{d+1}^\flat \ .$$

For any element h of $\mathbb{H}_{d,0}$ or $\mathbb{H}_{d,1}$ we call $\left[h\right]^\sharp$ the element of $\mathbb{H}_d^\sharp$ or $\mathbb{H}_d^\flat$ corresponding to h with all the spurious uncertainties removed. By a slight abuse of notation, the

criterion j in (1.38) (decision-hazard-decision setting) corresponds to $j \circ \big[\,\cdot\,\big]^\sharp$ in the two time scales setting in §1.4.1. The feedbacks in the two time scales setting in §1.4.1 are in one to one correspondence with the same elements in the decision-hazard-decision setting, namely

$$\gamma_{d,0} = \gamma_d^\sharp \circ \big[\,\cdot\,\big]^\sharp \;,\;\; \gamma_{d,1} = \gamma_{d+1}^\flat \circ \big[\,\cdot\,\big]^\sharp \;.$$

Now we define two families of stochastic kernels

- a family $\big\{\rho_{(d,0):(d,1)}\big\}_{0 \leqslant d \leqslant D}$ of stochastic kernels within two consecutive slow scale indexes

$$\begin{aligned} \rho_{(d,0):(d,1)} : \mathbb{H}_{d,0} &\to \Delta(\mathbb{W}_{d,1}) \;, \\ h_{d,0} &\mapsto \rho_{d:d+1} \circ \big[\,\cdot\,\big]^\sharp \;. \end{aligned}$$

- a family $\big\{\rho_{(d,1):(d+1,0)}\big\}_{0 \leqslant d \leqslant D-1}$ of stochastic kernels across two consecutive slow scale indexes

$$\begin{aligned} \rho_{(d,1):(d+1,0)} : \mathbb{H}_{d,1} &\to \Delta(\mathbb{W}_{d+1,0}) \;, \\ h_{d,1} &\mapsto \delta_{\overline{w}_{d+1}^\sharp}(\cdot) \;, \end{aligned}$$

where we recall that $\mathbb{W}_{d+1,0} = \mathbb{W}_{d+1}^\sharp = \{\overline{w}_{d+1}^\sharp\}$.

With these notations, we obtain Equation (1.41b), where only one integral appears because of the Dirac in the stochastic kernels $\rho_{(d,1):(d+1,0)}$. Indeed, for any measurable function $\varphi : \mathbb{H}_{d+1,0} \to [0, +\infty]$, we have that

$$\begin{aligned} \big(\mathcal{B}_{d+1:d}\varphi\big)(h_{d,0}) = \inf_{u_{d,0} \in \mathbb{U}_{d,0}} \int_{\mathbb{W}_{d,1}} \rho_{(d,0):(d,1)}\Big(h_{d,0}, \mathrm{d}w_{d,1}\Big) \\ \inf_{u_{d,1} \in \mathbb{U}_{d,1}} \int_{\mathbb{W}_{d+1,0}} \varphi\big(h_{d,0}, u_{d,0}, w_{d,1}, u_{d,1}, w_{d+1,0}\big) \rho_{(d,1):(d+1,0)}\Big(h_{d,0}, h_{d:d+1}, \mathrm{d}w_{d+1,0}\Big) \;. \end{aligned}$$

Now, if there exists $\widetilde{\varphi} : \mathbb{H}_{d+1}^\sharp \to [0, +\infty]$ such that $\varphi = \widetilde{\varphi} \circ \big[\,\cdot\,\big]^\sharp$, we obtain that

$$\begin{aligned} \big(\mathcal{B}_{d+1:d}\varphi\big)(h_{d,0}) &= \inf_{u_{d,0} \in \mathbb{U}_{d,0}} \int_{\mathbb{W}_{d,1}} \rho_{(d,0):(d,1)}\Big(h_{d,0}, \mathrm{d}w_{d,1}\Big) \inf_{u_{d,1} \in \mathbb{U}_{d,1}} \widetilde{\varphi}\big(\big[h_{d,0}\big]^\sharp, u_{d,0}, w_{d,1}, u_{d,1}\big) \\ &\qquad \int_{\mathbb{W}_{d+1,0}} \rho_{(d,1):(d+1,0)}\Big(h_{d,0}, h_{d:d+1}, \mathrm{d}w_{d+1,0}\Big) \\ &= \inf_{u_{d,0} \in \mathbb{U}_{d,0}} \int_{\mathbb{W}_{d,1}} \rho_{(d,0):(d,1)}\Big(h_{d,0}, \mathrm{d}w_{d,1}\Big) \inf_{u_{d,1} \in \mathbb{U}_{d,1}} \widetilde{\varphi}\big(\big[h_{d,0}\big]^\sharp, u_{d,0}, w_{d,1}, u_{d,1}\big) \end{aligned}$$

by the Dirac probability of the stochastic kernels $\rho_{(d,1):(d+1,0)}$,

$$= \inf_{u_d^\sharp \in \mathbb{U}_d^\sharp} \int_{\mathbb{W}_{d+1}^\flat} \rho_{(d,0):(d,1)}\Big(h_d^\sharp, \mathrm{d}w_{d+1}^\flat\Big) \inf_{u_{d+1}^\flat \in \mathbb{U}_{d+1}^\flat} \widetilde{\varphi}(h_d^\sharp, u_d^\sharp, w_{d+1}^\flat, u_{d+1}^\flat)$$

This ends the proof. □

1.7 Dynamic Programming with Unit Time Blocks

Here, we recover the classical dynamic programming equations when a state reduction exists at each time $t = 0, \dots, T-1$, with associated dynamics. Following the setting in §1.2.2, we consider a family $\{\rho_{t-1:t}\}_{1 \leqslant t \leqslant T}$ of stochastic kernels as in (1.3) and a measurable nonnegative numerical cost function j as in (1.4).

1.7.1 The General Case of Unit Time Blocks

First, we treat the general criterion case. We assume the existence of a family of measurable state spaces $\{\mathbb{X}_t\}_{t=0,\dots,T}$ and the existence of a family of measurable mappings $\{\theta_t\}_{t=0,\dots,T}$ with $\theta_t : \mathbb{H}_t \to \mathbb{X}_t$. We suppose that there exists a family of measurable dynamics $\{f_{t:t+1}\}_{t=0,\dots,T-1}$ with $f_{t:t+1} : \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1} \to \mathbb{X}_{t+1}$, such that

$$\theta_{t+1}\big((h_t, u_t, w_{t+1})\big) = f_{t:t+1}\big(\theta_t(h_t), u_t, w_{t+1}\big) \ , \ \ \forall (h_t, u_t, w_{t+1}) \in \mathbb{H}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1} \ . \quad (1.65)$$

The following proposition is a immediate application of Theorem 5 and Proposition 3.

Proposition 13. *Suppose that the triplet* $(\{\mathbb{X}_t\}_{t=0,\dots,T}, \{\theta_t\}_{t=0,\dots,T}, \{f_{t:t+1}\}_{t=0,\dots,T-1})$*, which is a state reduction across the consecutive time blocks* $[t, t+1]_{t=0,\dots,T-1}$ *of the time span, is compatible with the family* $\{\rho_{t-1:t}\}_{t=1,\dots,T}$ *of stochastic kernels in (1.3) (see Definition 4).*

Suppose that there exists a measurable nonnegative numerical function

$$\widetilde{j} : \mathbb{X}_T \to [0, +\infty] \ ,$$

such that the cost function j in (1.4) can be factored as

$$j = \widetilde{j} \circ \theta_T \ .$$

Define the family $\left\{\widetilde{V}_t\right\}_{t=0,\dots,T}$ *of functions by the backward induction*

$$\widetilde{V}_T(x_T) = \widetilde{j}(x_T) \ , \ \ \forall x_T \in \mathbb{X}_T \ , \quad (1.67a)$$

$$\widetilde{V}_t(x_t) = \inf_{u_t \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \widetilde{V}_{t+1}\big(f_{t:t+1}(x_t, u_t, w_{t+1})\big) \widetilde{\rho}_{t:t+1}(x_t, \mathrm{d}w_{t+1}) \ , \ \ \forall x_t \in \mathbb{X}_t \ , \quad (1.67b)$$

for $t = T-1, \dots, 0$.

Then, the family $\{V_t\}_{t=0,\dots,T}$ *of value functions defined by the family of optimization problems (1.6) satisfies*

$$V_t = \widetilde{V}_t \circ \theta_t \ , \ \ t = 0, \dots, T \ . \quad (1.68)$$

1.7.2 The Case of Time Additive Cost Functions

A time additive stochastic optimal control problem is a particular form of the stochastic optimization problem presented previously. As in §1.7.1, we assume the existence of a family of measurable state spaces $\{\mathbb{X}_t\}_{t=0,\dots,T}$, the existence of a family of measurable

mappings $\{\theta_t\}_{t=0,\dots,T}$, and the existence of a family of measurable dynamics such that Equation (1.65) is fulfilled.

We then assume that, for $t = 0, \dots, T-1$, there exist measurable nonnegative numerical functions (*instantaneous cost*)

$$L_t : \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1} \to [0, +\infty] \ ,$$

and that there exists a measurable nonnegative numerical function (*final cost*)

$$K : \mathbb{X}_T \to [0, +\infty] \ ,$$

such that the cost function j in (1.4) writes

$$j(h_T) = \sum_{t=0}^{T-1} L_t\big(\theta_t(h_t), u_t, w_{t+1}\big) + K\big(\theta_T(h_T)\big) \ .$$

The following proposition is an immediate consequence of the specific form of the cost function j when applying Proposition 13.

Proposition 14. *Suppose that the triplet* $(\{\mathbb{X}_t\}_{t=0,\dots,T}, \{\theta_t\}_{t=0,\dots,T}, \{f_{t:t+1}\}_{t=0,\dots,T-1})$*, which is a state reduction across the consecutive time blocks* $[t, t+1]_{t=0,\dots,T-1}$ *of the time span, is compatible with the family* $\{\rho_{t-1:t}\}_{t=1,\dots,T}$ *of stochastic kernels in (1.3) (see Definition 4).*

We inductively define the family of functions $\{\widehat{V}_t\}_{t=0,\dots,T}$*, with* $\widehat{V}_t : \mathbb{X}_t \to [0, +\infty]$*, by the relations*

$$\widehat{V}_T(x_T) = K(x_T) \ , \quad \forall x_T \in \mathbb{X}_T \tag{1.70a}$$

and, for $t = T-1, \dots, 0$ *and for all* $x_t \in \mathbb{X}_t$*,*

$$\widehat{V}_t(x_t) = \inf_{u_t \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \Big(L_t(x_t, u_t, w_{t+1}) + \widehat{V}_{t+1}\big(f_{t:t+1}(x_t, u_t, w_{t+1})\big)\Big) \widetilde{\rho}_{t:t+1}(x_t, \mathrm{d}w_{t+1}) \ . \tag{1.70b}$$

Then, the family $\{V_t\}_{t=0,\dots,T}$ *of value functions defined by the family of optimization problems (1.6) satisfies*

$$V_t(h_t) = \sum_{s=0}^{t-1} L_s\big(\theta_s(h_s), u_s, w_{s+1}\big) + \widehat{V}_t\big(\theta_t(h_t)\big) \ , \quad t = 1, \dots, T \ , \tag{1.71a}$$

$$V_0(h_0) = \widehat{V}_0\big(\theta_0(h_0)\big) \ . \tag{1.71b}$$

1.8 The Case of Optimization with Noise Process

In this section, the *noise* at time t is modeled as a random variable $\boldsymbol{W}_t$. We suppose given a stochastic process $\{\boldsymbol{W}_t\}_{t=0,\dots,T}$ called *noise process*. Then, optimization with noise process becomes a special case of the setting in §1.2.2. Therefore, we can apply the results obtained in Sect. 1.3.

We moreover assume that, for any $s = 0, \dots, T-1$, the set $\mathbb{U}_s$ in §1.2.2 is a separable complete metric space.

1.8.1 Optimization with Noise Process

Noise Process and Stochastic Kernels. Let $(\Omega, \mathcal{A})$ be a measurable space. For $t = 0, \dots, T$, the *noise* at time t is modeled as a random variable $\boldsymbol{W}_t$ defined on Ω and taking values in $\mathbb{W}_t$. Therefore, we suppose given a stochastic process $\{\boldsymbol{W}_t\}_{t=0,\dots,T}$ called *noise process*. The following assumption is made in the sequel.

Assumption 1. *For any $1 \leqslant s \leqslant T$, there exists a regular conditional distribution of the random variable $\boldsymbol{W}_s$ knowing the random process $\boldsymbol{W}_{0:s-1}$, denoted by $\mathbb{P}_{\boldsymbol{W}_s}^{\boldsymbol{W}_{0:s-1}}(w_{0:s-1}, \mathrm{d}w_s)$.*

Under Assumption 1, we can introduce the family $\{\rho_{s-1:s}\}_{1 \leqslant s \leqslant T}$ of stochastic kernels

$$\rho_{s-1:s} : \mathbb{H}_{s-1} \to \Delta(\mathbb{W}_s) \ , \quad s = 1, \dots, T \ , \tag{1.72a}$$

defined by

$$\rho_{s-1:s}(h_{s-1}, \mathrm{d}w_s) = \mathbb{P}_{\boldsymbol{W}_s}^{\boldsymbol{W}_{0:s-1}}\big([h_{s-1}]_{0:s-1}^{\mathbb{W}}, \mathrm{d}w_s\big) \ , \quad s = 1, \dots, T \ , \tag{1.72b}$$

where $[h_{s-1}]_{0:s-1}^{\mathbb{W}} = (w_0, w_1, \dots, w_{s-1})$ is the uncertainty part of the history h_{s-1} (see Equation (1.49a)).

Then, using Definition 11, the stochastic kernels $\rho_{r:t}^{\gamma} : \mathbb{H}_r \to \Delta(\mathbb{H}_t)$ are defined, for any measurable nonnegative numerical function $\varphi : \mathbb{H}_t \to [0, +\infty]$, by

$$\begin{aligned} \int_{\mathbb{H}_t} \varphi(h_t') \rho_{r:t}^{\gamma}(h_r, \mathrm{d}h_t') &= \int_{\mathbb{W}_{r+1:t}} \varphi\Big(\Phi_{r:t}^{\gamma}(h_r, w_{r+1:t})\Big) \mathbb{P}_{\boldsymbol{W}_{r+1:t}}^{\boldsymbol{W}_{0:r}}\big([h_r]_{0:r}^{\mathbb{W}}, \mathrm{d}w_{r+1:t}\big) \ . \\ &= \mathbb{E}\Big[\varphi\big(\Phi_{r:t}^{\gamma}(h_r, \boldsymbol{W}_{r+1:t})\big) \ \Big| \ \boldsymbol{W}_{0:r} = [h_r]_{0:r}^{\mathbb{W}}\Big] \ , \end{aligned} \tag{1.73}$$

where $\Phi_{r:t}^{\gamma}(h_r, w_{r+1:t}) = (h_r, \gamma_r(h_r), w_{r+1}, \gamma_{r+1}(h_r, \gamma_r(h_r), w_{r+1}), w_{r+2}, \cdots, \gamma_{t-1}(h_{t-1}), w_t)$ is the flow induced by the feedback γ (see §1.6.1).

Adapted Control Processes. Let t be given such that $0 \leqslant t \leqslant T-1$. We introduce

$$\mathcal{A}_{t:t} = \{\varnothing, \Omega\} \ , \ \mathcal{A}_{t:t+1} = \sigma(\boldsymbol{W}_{t+1}) \ , \ \dots, \ , \ \mathcal{A}_{t:T-1} = \sigma(\boldsymbol{W}_{t+1}, \dots, \boldsymbol{W}_{T-1}) \ .$$

Let $\mathbb{L}^0(\Omega, \mathcal{A}_{t:T-1}, \mathbb{U}_{t:T-1})$ be the space of $\mathcal{A}$-*adapted control processes* $(\boldsymbol{U}_t, \dots, \boldsymbol{U}_{T-1})$ with values in $\mathbb{U}_{t:T-1}$, that is, such that

$$\sigma(\boldsymbol{U}_s) \subset \mathcal{A}_{t:s} \ , \quad s = t, \dots, T-1 \ .$$

Family of Optimization Problems over Adapted Control Processes. We suppose here that the measurable space $(\Omega, \mathcal{A})$ is equipped with a probability $\mathbb{P}$, so that $(\Omega, \mathcal{A}, \mathbb{P})$ is a probability space. Following the setting given in §1.2.2, we consider a measurable nonnegative numerical cost function j as in Equation (1.4).

We consider the following family of optimization problems, indexed by $t = 0, \dots, T-1$ and by $h_t \in \mathbb{H}_t$,

$$\check{V}_t(h_t) = \inf_{(\boldsymbol{U}_{t:T-1}) \in \mathbb{L}^0(\Omega, \mathcal{A}_{t:T-1}, \mathbb{U}_{t:T-1})} \mathbb{E}\Big[j(h_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}, \dots, \boldsymbol{U}_{T-1}, \boldsymbol{W}_T) \ \Big| \ \boldsymbol{W}_{0:t} = [h_t]_{0:t}^{\mathbb{W}}\Big] \ . \tag{1.74}$$

Proposition 15. *Let $t \in \{0, \ldots, T-1\}$ and $h_t \in \mathbb{H}_t$ be given. Problem (1.5) and Problem (1.74) coincide, that is,*

$$\check{V}_t(h_t) = V_t(h_t) \ , \tag{1.75}$$

where the family of value functions $\{V_t\}_{t=0,\ldots,T}$ is defined by (1.6).

Proof. Let $t \in \{0, \ldots, T-1\}$ and $h_t \in \mathbb{H}_t$ be given. We show that Problem (1.74) and Problem (1.5) are in one-to-one correspondence.

- First, for any history feedback $\gamma_{t:T-1} = \{\gamma_s\}_{s=t,\ldots,T-1} \in \Gamma_{t:T-1}$, we define $(\boldsymbol{U}_{t:T-1}) \in \mathbb{L}^0(\Omega, \mathcal{A}_{t:T-1}, \mathbb{U}_{t:T-1})$ by

$$(\boldsymbol{U}_t, \ldots, \boldsymbol{U}_{T-1}) = \left[\Phi^{\gamma}_{t:T}(h_t, \boldsymbol{W}_{t+1}, \ldots, \boldsymbol{W}_T)\right]^{\mathbb{U}}_{t+1:T} \ , \tag{1.76}$$

  where the flow $\Phi^{\gamma}_{t:T}$ has been defined in (1.50) and the history control part $[\cdot]^{\mathbb{U}}_{t+1:T}$ in (1.49b). By the expression (1.72b) of $\rho_{s:s+1}(h'_s, \mathrm{d}w_{s+1})$ and by Definition 11 of the stochastic kernel $\rho^{\gamma}_{t:T}$, we obtain that

$$\begin{aligned}\mathbb{E}\Big[j(h_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}, \ldots, \boldsymbol{U}_{T-1}, \boldsymbol{W}_T) \ \Big| \ \boldsymbol{W}_{0:t} = [h_t]^{\mathbb{W}}_{0:t}\Big] &= \mathbb{E}\Big[j(\Phi^{\gamma}_{t:T}(h_t, \boldsymbol{W}_{t+1}, \ldots, \boldsymbol{W}_T)) \ \Big| \ \boldsymbol{W}_{0:t} = [h_t]^{\mathbb{W}}_{0:t}\Big] \\ &= \int_{\mathbb{H}_T} j(h'_T)\rho^{\gamma}_{t:T}(h_t, \mathrm{d}h'_T) \ .\end{aligned} \tag{1.77}$$

  As a consequence

$$\begin{aligned}\inf_{(\boldsymbol{U}_{t:T-1}) \in \mathbb{L}^0(\Omega, \mathcal{A}_{t:T-1}, \mathbb{U}_{t:T-1})} &\mathbb{E}\Big[j(h_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}, \ldots, \boldsymbol{U}_{T-1}, \boldsymbol{W}_T) \ \Big| \ \boldsymbol{W}_{0:t} = [h_t]^{\mathbb{W}}_{0:t}\Big] \\ &\leqslant \inf_{\gamma_{t:T-1} \in \Gamma_{t:T-1}} \int_{\mathbb{H}_T} j(h'_T)\rho^{\gamma}_{t:T}(h_t, \mathrm{d}h'_T) \ .\end{aligned} \tag{1.78}$$

- Second, we define a $(t:T-1)$-*noise feedback* as a sequence $\lambda = \{\lambda_s\}_{s=t,\ldots,T-1}$ of measurable mappings (the mapping λ_t is constant)

$$\lambda_t \in \mathbb{U}_t \ , \ \ \lambda_s : \mathbb{W}_{t+1:s} \to \mathbb{U}_s \ , \ \ t+1 \leqslant s \leqslant T-1 \ .$$

  We denote by $\Lambda_{t:T-1}$ the set of $(t:T-1)$-noise feedbacks. Let $(\boldsymbol{U}_t, \ldots, \boldsymbol{U}_{T-1}) \in \mathbb{L}^0(\Omega, \mathcal{A}_{t:T-1}, \mathbb{U}_{t:T-1})$. As each set $\mathbb{U}_s$ is a separable complete metric space, for $s = t, \ldots, T-1$, we can invoke Doob Theorem (see [25, Chap. 1, p. 18]). Therefore, there exists a $(t:T-1)$-noise feedback $\lambda = \{\lambda_s\}_{s=t,\ldots,T-1} \in \Lambda_{t:T-1}$ such that

$$\boldsymbol{U}_t = \lambda_t \ , \ \ \boldsymbol{U}_s = \lambda_s(\boldsymbol{W}_{t+1:s}) \ , \ \ t+1 \leqslant s \leqslant T-1 \ .$$

  Then, we define the history feedback $\gamma_{t:T-1} = \{\gamma_s\}_{s=t,\ldots,T-1} \in \Gamma_{t:T-1}$ by, for any history $h'_r \in \mathbb{H}_r$, $r = t, \ldots, T-1$:

$$\begin{aligned}\gamma_t(h'_t) &= \lambda_t \ , \\ \gamma_{t+1}(h'_{t+1}) &= \lambda_{t+1}\Big(\big[h'_{t+1}\big]^{\mathbb{W}}_{t+1:t+1}\Big) = \lambda_{t+1}(w'_{t+1}) \ , \\ &\vdots \\ \gamma_{T-1}(h'_{T-1}) &= \lambda_{T-1}\Big(\big[h'_{T-1}\big]^{\mathbb{W}}_{t+1:T-1}\Big) = \lambda_{T-1}(w'_{t+1}, \cdots, w'_{T-1}) \ .\end{aligned}$$

By the expression (1.72b) of $\rho_{s:s+1}(h'_s, \mathrm{d}w_{s+1})$ and by Definition 11 of the stochastic kernel $\rho^{\gamma}_{t:T}$, we obtain that

$$\int_{\mathbb{H}_T} j(h'_T)\rho^{\gamma}_{t:T}(h_t, \mathrm{d}h'_T) = \mathbb{E}\Big[j(h_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}, \dots, \boldsymbol{U}_{T-1}, \boldsymbol{W}_T) \;\Big|\; \boldsymbol{W}_{0:t} = [h_t]^{\mathbb{W}}_{0:t}\Big] .$$

As a consequence

$$\begin{aligned} &\inf_{\gamma_{t:T-1}\in\Gamma_{t:T-1}} \int_{\mathbb{H}_T} j(h'_T)\rho^{\gamma}_{t:T}(h_t, \mathrm{d}h'_T) \\ &\leqslant \inf_{(\boldsymbol{U}_t,\dots,\boldsymbol{U}_{T-1})\in\mathbb{L}^0(\Omega,\mathcal{A}_{t:T-1},\mathbb{U}_{t:T-1})} \mathbb{E}\Big[j(h_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}, \dots, \boldsymbol{U}_{T-1}, \boldsymbol{W}_T) \;\Big|\; \boldsymbol{W}_{0:t} = [h_t]^{\mathbb{W}}_{0:t}\Big] . \end{aligned} \tag{1.79}$$

Gathering inequalities (1.78) and (1.79) leads to (1.75). This ends the proof. □

The following proposition is an immediate consequence of Theorem 1 and Proposition 15.

Proposition 16. *The family* $\left\{\check{V}_t\right\}_{t=0,\dots,T}$ *of functions in (1.74) satisfies the backward induction*

$$\check{V}_T(h_T) = j(h_T) , \quad \forall h_T \in \mathbb{H}_T , \tag{1.80a}$$

and, for $t = T-1, \dots, 0$,

$$\check{V}_t(h_t) = \inf_{u_t} \int_{\mathbb{W}_{t+1}} \check{V}_{t+1}\big(h_t, u_t, w_{t+1}\big)\mathbb{P}^{\boldsymbol{W}_{0:t}}_{\boldsymbol{W}_{t+1}}\big([h_t]^{\mathbb{W}}_{0:t}, \mathrm{d}w_{t+1}\big) \tag{1.80b}$$

$$= \inf_{u_t} \mathbb{E}\big[\check{V}_{t+1}\big(h_t, u_t, \boldsymbol{W}_{t+1}\big) \;\big|\; \boldsymbol{W}_{0:t} = [h_t]^{\mathbb{W}}_{0:t}\big] , \quad \forall h_t \in \mathbb{H}_t . \tag{1.80c}$$

1.8.2 Two Time-Scales Dynamic Programming

We adopt the notation of §1.4.1. We suppose given a two time-scales noise process

$$\boldsymbol{W}_{(0,0):(D+1,0)} = \big(\boldsymbol{W}_{0,0}, \boldsymbol{W}_{0,1}, \dots, \boldsymbol{W}_{0,M}, \boldsymbol{W}_{1,0}, \dots, \boldsymbol{W}_{D,M}, \boldsymbol{W}_{D+1,0}\big) .$$

For any $d \in \{0, 1, \dots, D\}$, we introduce the σ-fields

$$\mathcal{A}_{d,0} = \{\varnothing, \Omega\} , \quad \mathcal{A}_{d,m} = \sigma(\boldsymbol{W}_{(d,1):(d,m)}) , \quad m = 1, \dots, M .$$

The proof of the following proposition is left to the reader.

Proposition 17. *Suppose that there exists a family* $\{\mathbb{X}_d\}_{d=0,\dots,D+1}$ *of measurable state spaces, with* $\mathbb{X}_0 = \mathbb{W}_{0,0}$*, and a family* $\{f_{d:d+1}\}_{d=0,\dots,D}$ *of measurable dynamics*

$$f_{d:d+1} : \mathbb{X}_d \times \mathbb{H}_{d:d+1} \to \mathbb{X}_{d+1} .$$

Suppose that the slow scale subprocesses $\boldsymbol{W}_{(d,1):(d+1,0)} = \big(\boldsymbol{W}_{d,1}, \cdots, \boldsymbol{W}_{d+1,0}\big)$, $d = 0, \dots, D$, *are independent (under the probability law* $\mathbb{P}$*).*

For a measurable nonnegative numerical cost function

$$\widetilde{j} : \mathbb{X}_{D+1} \to [0, +\infty] \,,$$

we define the family $\Big\{\widetilde{V}_d\Big\}_{d=0,\dots,D+1}$ *of functions by the backward induction*

$$\widetilde{V}_{D+1}(x_{D+1}) = \widetilde{j}(x_{D+1}) \,, \tag{1.81a}$$

$$\widetilde{V}_d(x_d) = \inf_{\boldsymbol{U}_{(d,0):(d,M)} \in \mathbb{L}^0(\Omega, \mathcal{A}_{(d,0):(d,M)}, \mathbb{U}_{(d,0):(d,M)})} \mathbb{E}\Big[\widetilde{V}_{d+1}\big(f_{d:d+1}(x_d, \boldsymbol{U}_{d,0}, \boldsymbol{W}_{d,1}, \cdots, \boldsymbol{U}_{d,M}, \boldsymbol{W}_{d+1,0})\big)\Big] \,. \tag{1.81b}$$

Then, the value functions $\widetilde{V}_d$ *are the solution of the following family of optimization problems, indexed by* $d = 0, \dots, D$ *and by* $x_d \in \mathbb{X}_d$*,*

$$\widetilde{V}_d(x_d) = \inf_{U_{(d,0):(D,M)} \in \mathbb{L}^0(\Omega, \mathcal{A}_{(d,0):(D,M)}, \mathbb{U}_{(d,0):(D,M)})} \mathbb{E}\big[\widetilde{j}(\boldsymbol{X}_{D+1})\big] \,, \tag{1.82a}$$

where, for all $d' = d, \dots, D$*,*

$$\boldsymbol{X}_d = x_d \,, \quad \boldsymbol{X}_{d'+1} = f_{d':d'+1}\big(\boldsymbol{X}_{d'}, \boldsymbol{U}_{d',0}, \boldsymbol{W}_{d',1}, \cdots, \boldsymbol{U}_{d',M}, \boldsymbol{W}_{d'+1,0}\big) \,. \tag{1.82b}$$

1.8.3 Decision-Hazard-Decision Dynamic Programming

We adopt the notation of §1.4.2. We suppose given a noise process

$$\boldsymbol{W}_{0:S} = \big(\boldsymbol{W}_0^\sharp, \boldsymbol{W}_1^\flat, \dots, \boldsymbol{W}_S^\flat\big) \,. \tag{1.83}$$

For any $s \in \{0, 1, \dots, S-1\}$, we introduce the σ-fields

$$\mathcal{A}_s = \{\varnothing, \Omega\} \,, \quad \mathcal{A}_{s'} = \sigma(\boldsymbol{W}_{s+1:s'}^\flat) \,, \quad s' = s+1, \dots, S \,. \tag{1.84}$$

The proof of the following proposition is left to the reader.

Proposition 18. *Suppose that there exists a family* $\{\mathbb{X}_s\}_{s=0,\dots,S}$ *of measurable state spaces, with* $\mathbb{X}_0 = \mathbb{W}_0^\sharp$*, and a family* $\{f_{s:s+1}\}_{s=0,\dots,S-1}$ *of measurable dynamics*

$$f_{s:s+1} : \mathbb{X}_s \times \mathbb{U}_s^\sharp \times \mathbb{W}_{s+1}^\flat \times \mathbb{U}_{s+1}^\flat \to \mathbb{X}_{s+1} \,.$$

Suppose that the noise process $\big\{\boldsymbol{W}_s^\flat\big\}_{s=0,\dots,S}$ *is made of independent random variables (under the probability law* $\mathbb{P}$*).*

For a measurable nonnegative numerical cost function

$$\widetilde{j} : \mathbb{X}_S \to [0, +\infty] \,, \tag{1.85}$$

we define the family of functions $\left\{\widetilde{V}_s\right\}_{s=0,\dots,S}$ *by the backward induction*

$$\widetilde{V}_S(x_S) = \widetilde{j}(x_S) \;, \tag{1.86a}$$

$$\widetilde{V}_s(x_s) = \inf_{u_s^\sharp \in \mathbb{U}_s^\sharp} \mathbb{E}\Big[\inf_{u_{s+1}^\flat \in \mathbb{U}_{s+1}^\flat} \widetilde{V}_{s+1}\Big(f_{s':s'+1}\big(x_s, u_s^\sharp, \boldsymbol{W}_{s+1}^\flat, u_{s+1}^\flat\big)\Big)\Big] \;. \tag{1.86b}$$

Then, the value functions $\widetilde{V}_s$ *in (1.86) are the solution of the following family of optimization problems, indexed by* $s = 0, \dots, S-1$ *and by* $x_s \in \mathbb{X}_s$*,*

$$\widetilde{V}_s(x_s) = \inf_{\boldsymbol{U}_{s:S-1}^\sharp \in \mathbb{L}^0(\Omega, \mathcal{A}_{s:S-1}, \mathbb{U}_{s:S-1}^\sharp)} \; \inf_{\boldsymbol{U}_{s+1:S}^\flat \in \mathbb{L}^0(\Omega, \mathcal{A}_{s+1:S}, \mathbb{U}_{s+1:S}^\flat)} \mathbb{E}\big[\widetilde{j}(\boldsymbol{X}_S)\big] \;, \tag{1.87a}$$

where

$$\boldsymbol{X}_{s'} = x_s \;, \;\; \boldsymbol{X}_{s'+1} = f_{s':s'+1}\big(\boldsymbol{X}_{s'}, \boldsymbol{U}_{s'}^\sharp, \boldsymbol{W}_{s'+1}^\flat, \boldsymbol{U}_{s'+1}^\flat\big) \;, \;\; \forall s' = s, \dots, S-1 \;. \tag{1.87b}$$

Chapter 1. Bibliography

[21] K. Barty, P. Carpentier, and P. Girardeau. Decomposition of large-scale stochastic optimal control problems. *RAIRO Operations Research*, 44(3):167–183, 2010.

[22] R. Bellman. *Dynamic Programming*. Princeton University Press, New Jersey, 1957.

[23] D. P. Bertsekas and S. E. Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Athena Scientific, Belmont, Massachusetts, 1996.

[24] P. Carpentier, J.-P. Chancelier, G. Cohen, and M. De Lara. *Stochastic Multi-Stage Optimization. At the Crossroads between Discrete Time Stochastic Control and Stochastic Programming*. Springer-Verlag, Berlin, 2015.

[25] C. Dellacherie and P. Meyer. *Probabilités et potentiel*. Hermann, Paris, 1975.

[26] I. V. Evstigneev. Measurable selection and dynamic programming. *Mathematics of Operations Research*, 1(3):267–272, 1976.

[27] M. Loève. *Probability Theory I*. Springer-Verlag, New York, fourth edition, 1977.

[28] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

[29] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147, 1991.

[30] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[31] H. S. Witsenhausen. A standard form for sequential stochastic control. *Mathematical Systems Theory*, 7(1):5–11, 1973.

[32] H. S. Witsenhausen. On policy independence of conditional expectations. *Information and Control*, 28(1):65–75, 1975.

Chapter 2

A template to design online policies for multistage stochastic optimization problems

This is a joint work with Pierre Carpentier, Jean-Philippe Chancelier, Michel De Lara and François Pacaud.

Chapter Abstract

The solutions of multistage stochastic optimization problems are policies, that is, mappings from past information into the current decision set, at each stage. In practice, one does not compute the whole policy but produces the value of the policy for the current argument and stage. We propose a general template to design online control policies for multistage stochastic optimization problems. Our approach stresses the role of information structures in the design of online policies. We use Chapter 1 formalism to build the general template of online policies. Then, we frame well known methods (Stochastic Dynamic Programming, Stochastic Programming, Stochastic Model Predictive Control) to produce online control policies within this template.

Contents

2.1 Introduction

In multistage stochastic optimization problems, the decision maker chooses, at every stage, a decision in a way that depends at most upon past uncertainties. Once a decision made, she/he has to wait till the next realization of uncertainty to make a new decision.

The resolution of such problems proves to be difficult. On the one hand, stochastic programming methods (see [48]) handle the resolution by writing a scenario tree corresponding to realizations of all uncertainties, hence potentially (very) large. On the other hand, stochastic control methods rely on the dynamic programming principle and look for solutions as feedback on previous history, but are naturally confronted to the *curse of dimensionality*. We refer to [33]—[35]—[45] for an overview of stochastic dynamic programming methods.

Because exact solutions are out of reach, there is ongoing interest in the design of *approximate* resolution algorithms designed to tackle the shortfalls of *exact* resolution methods. In theory, one looks for a policy, that is, a mapping from past information into the current decision set, at each stage. However, in practice, one only needs the single value of the policy at each stage when evaluated for the current past information. We will call *online policy* any mechanism that makes it possible to compute a single decision (output) on the basis of the current stage and current past information (input).

This is the approach taken by Bertsekas and Powell in their panoramic papers on the design of online policies. In [34], the emphasis is put on the well-known model predictive control method. In [35, Chap. 6], a whole chapter is dedicated to *approximate dynamic programming* methods. More recently, [44] proposed a framework to categorize different resolution algorithms in four different classes, with emphasis on approximate dynamic programming and reinforcement learning methods.

In this paper, we offer a complementary view as our approach stresses the role of information structures in the design of online policies. We show that well-known classical methods can be put in a common framework by using variations on the measurability properties (information patterns) of the ingredients that constitute them. We moreover explore the links between measurability and discretization for these methods. In Sect. 2.2, we formulate multistage stochastic programming over a *history* space and we introduce two classes of history feedback policies: the so-called cost-to-go policies and lookahead policies. We sketch theses two classes in the history framework (of increasing size as

time goes on) and not in the traditional state space framework as it is usual, at least for cost-to-go policies. This allows us to use the same notations in both frameworks. In Sect. 2.3, we focus on lookahead policies and frame well-known existing algorithms in this class. We show that stochastic programming, open-loop feedback control and model predictive controls share the same framework. In Sect. 2.4, we describe the cost-to-go methods and detail the offline computation of cost-to-go. Again, we show that stochastic dynamic programming, stochastic dual dynamic programming and approximate dynamic programming share the same framework. Finally, in Sect. 2.5, we present a method to compare different online policies together, in a fair manner. Technical material is relegated in the Appendix.

2.2 Multistage stochastic optimization problems and online policies

We formulate multistage stochastic optimization problems over increasing history spaces in §2.2.1. Here, a history is made of past noises and controls, differing from histories in scenario trees and in state space formulations. Thus, solutions are policies which are mappings from history spaces to control sets. This is discussed in §2.2.2, where we introduce Bellman recursive equations over increasing history spaces, then discretization issues for the design of online policies.

2.2.1 Multistage stochastic optimization problems over history

Consider the time span $\{0, 1, 2 \ldots, T-1, T\}$, with *horizon* $T \in \mathbb{N}^*$. At the end of the time interval $[t-1, t)$, an uncertainty variable w_t is produced. Then, at the beginning of the time interval $[t, t+1)$, a decision-maker takes a decision u_t. The interplay between uncertainty and decision is $w_0 \rightsquigarrow u_0 \rightsquigarrow w_1 \rightsquigarrow u_1 \rightsquigarrow \quad \ldots \quad \rightsquigarrow w_{T-1} \rightsquigarrow u_{T-1} \rightsquigarrow w_T$. We use the notation $[\![r, s]\!] = \{r, r+1, \ldots, s-1, s\}$ to denote the integer interval between $r \in \mathbb{N}$ and $s \in \mathbb{N}$ with $s \geqslant r$.

Histories and history feedback policies

We first define the spaces needed to formulate multistage stochastic optimization problems. Then, we introduce a class of solutions called history feedback policies.

Histories and history spaces. For each time $t \in [\![0, T-1]\!]$, the decision u_t takes its values in a measurable set $\mathbb{U}_t$ equipped with a σ-field $\mathcal{U}_t$. For each time $t \in [\![0, T]\!]$, the uncertainty w_t takes its values in a measurable set $\mathbb{W}_t$ equipped with a σ-field $\mathcal{W}_t$. We suppose that all spaces are separable complete metric, a proper assumption to apply a Doob theorem [39, Chap. 1, p. 18].

For $t \in [\![1, T]\!]$, we define the *history space* $\mathbb{H}_t$ equipped with the *history field* $\mathcal{H}_t$

$$\mathbb{H}_t = \mathbb{W}_0 \times \prod_{s=0}^{t-1} (\mathbb{U}_s \times \mathbb{W}_{s+1}) \text{ and } \mathcal{H}_t = \mathcal{W}_0 \otimes \bigotimes_{s=0}^{t-1} (\mathcal{U}_s \otimes \mathcal{W}_{s+1}) \,, \tag{2.1}$$

with the particular case $\mathbb{H}_0 = \mathbb{W}_0$, $\mathcal{H}_0 = \mathcal{W}_0$. A generic element $h_t \in \mathbb{H}_t$ is called a *history*:

$$h_t = \big(w_0, (u_s, w_{s+1})_{s\in[\![0,t-1]\!]}\big) = (w_0, u_0, w_1, u_1, w_2, \ldots, u_{t-2}, w_{t-1}, u_{t-1}, w_t) \; . \tag{2.2}$$

For $0 \leqslant r \leqslant t \leqslant T$, we introduce the notations

$$\mathbb{W}_{r:t} = \prod_{s=r}^{t} \mathbb{W}_s \; , \;\; \mathbb{U}_{r:t} = \prod_{s=r}^{t} \mathbb{U}_s \; , \;\; \text{and} \;\; \mathbb{H}_{r:t} = \prod_{s=r-1}^{t-1} (\mathbb{U}_s \times \mathbb{W}_{s+1}) \; , \tag{2.3}$$

with the specific case $\mathbb{H}_{0:t} = \mathbb{H}_t$. An element $h_{r:t} = (u_{r-1}, w_r, \ldots, u_{t-1}, w_t) \in \mathbb{H}_{r:t}$ is called a *partial history*.

History feedback policies. When $0 \leqslant r \leqslant t \leqslant T-1$, we define a $(r:t)$-*history feedback policy* as a sequence $\{\gamma_s\}_{s\in[\![r,t]\!]}$ of measurable mappings

$$\gamma_s : (\mathbb{H}_s, \mathcal{H}_s) \to (\mathbb{U}_s, \mathcal{U}_s) \; . \tag{2.4}$$

We call $\Gamma_{r:t}$ the set of $(r:t)$-history feedback policies. A generic element of $\Gamma_{r:t}$ will be denoted $\gamma_{r:t}$.

Optimization problems formulated with stochastic kernels

To cover both stochastic programming and dynamic programming in the same setting, we propose to formulate optimization problems by means of history feedbacks, criterion and stochastic kernels.

We first recall the notion of stochastic kernel. Let $(\mathbb{X}, \mathcal{X})$ and $(\mathbb{Y}, \mathcal{Y})$ be two measurable spaces. A *stochastic kernel* from $(\mathbb{X}, \mathcal{X})$ to $(\mathbb{Y}, \mathcal{Y})$ is a mapping $\rho : (\mathbb{X}, \mathcal{X}) \times \mathcal{Y} \to [0,1]$ such that i) for any $Y \in \mathcal{Y}$, $\rho(\cdot, Y)$ is $\mathcal{X}$-measurable ii) for any $x \in \mathbb{X}$, $\rho(x, \cdot)$ is a probability measure on $\mathcal{Y}$. A stochastic kernel is sometimes denoted as a mapping $\rho : (\mathbb{X}, \mathcal{X}) \to \Delta(\mathbb{Y}, \mathcal{Y})$ from the measurable space $(\mathbb{X}, \mathcal{X})$ towards the space $\Delta(\mathbb{Y}, \mathcal{Y})$ of probability measures over $\mathcal{Y}$, with the property that the function $x \in \mathbb{X} \mapsto \int_Y \rho(x, \mathrm{d}y)$ is $\mathcal{X}$-measurable for any $Y \in \mathcal{Y}$.

In what follows, a function taking its values in $[-\infty, +\infty]$ is called an *extended real-valued* function.

Family of optimization problems with stochastic kernels. To build a family of optimization problems over the time span $\{0, \ldots, T-1\}$, we introduce two ingredients:

- a family $\{\rho_{s-1:s}\}_{1\leqslant s\leqslant T}$ of stochastic kernels

$$\rho_{s-1:s} : (\mathbb{H}_{s-1}, \mathcal{H}_{s-1}) \times \mathcal{W}_s \to [0,1] \; , \;\; s = 1, \ldots, T \; , \tag{2.5}$$

- an extended real-valued function, a criterion (playing the role of a cost) to be minimized,

$$j : (\mathbb{H}_T, \mathcal{H}_T) \to [0, +\infty] \; , \tag{2.6}$$

assumed to be nonnegative[1] and measurable[2] with respect to the field $\mathcal{H}_T$.

We define, for any $(t:T-1)$-*history feedback policy* $\{\gamma_s\}_{s\in[\![t,T-1]\!]} \in \Gamma_{t:T-1}$, a new family of stochastic kernels

$$\rho^{\gamma}_{t:T} : (\mathbb{H}_t, \mathcal{H}_t) \times \mathcal{H}_T \to [0,1] \ ,$$

which captures the transitions between histories when the dynamics of the history, that is, $h_{s+1} = \big(h_s, u_s, w_{s+1}\big)$, is driven by $u_s = \gamma_s(h_s)$ for $s \in [\![t, T-1]\!]$ (see Definition 22 in Appendix 2.7 for the detailed construction of $\rho^{\gamma}_{r:t}$). Thus, $\rho^{\gamma}_{t:T}$ induces a family of probability distributions on the largest space $\mathbb{H}_T$ of histories (over the whole span $\{0,\dots,T\}$).

We consider the family of optimization problems, indexed by $t \in [\![0, T-1]\!]$ and parameterized by the history $h_t \in \mathbb{H}_t$:

$$\inf_{\gamma_{t:T-1}\in\Gamma_{t:T-1}} \int_{\mathbb{H}_T} j(h'_T)\rho^{\gamma}_{t:T}(h_t, \mathrm{d}h'_T) \ , \quad \forall h_t \in \mathbb{H}_t \ , \tag{2.7}$$

where the integral in the right-hand side of the above equation corresponds to the cost induced by the feedback $\gamma_{t:T-1}$ when starting at time t with a given history h_t. For all $t \in [\![0, T-1]\!]$, we define the minimum value of Problem (2.7) by

$$V_t(h_t) = \inf_{\gamma_{t:T-1}\in\Gamma_{t:T-1}} \int_{\mathbb{H}_T} j(h'_T)\rho^{\gamma}_{t:T}(h_t, \mathrm{d}h'_T) \ , \quad \forall h_t \in \mathbb{H}_t \ , \tag{2.8a}$$

and we also define

$$V_T(h_T) = j(h_T) \ , \quad \forall h_T \in \mathbb{H}_T \ . \tag{2.8b}$$

Each function V_t is nonnegative since so is the criterion j in (2.6). The extended real-valued function $V_t : (\mathbb{H}_t, \mathcal{H}_t) \to [0, +\infty]$ is called the *value function* at time t.

2.2.2 Online policies

In §2.2.2, we write explicitly the solution of multistage stochastic problems via Bellman recursive equations on the increasing history spaces. Then, in §2.2.2, we use the tools of partitions and finite subfields to tackle the issue of discretization, indispensable for the design of algorithms. Finally, we introduce in §2.2.2 two schemes to describe online policies — the cost-to-go policies and the lookahead policies — that will be developed in Sect. 2.3.

Stochastic dynamic programming with history feedback policies

We now recall the well-known result that the value functions defined in (2.8) are *Bellman functions*, that is, they are solution of the Bellman or dynamic programming equation. We make the following two assumptions.

[1] We could also consider any $j : \mathbb{H}_t \to \mathbb{R}$, measurable bounded function, or measurable and uniformly bounded below function. When $j(h_T) = +\infty$, this materializes joint constraints between uncertainties and controls.

[2] See [36] for the measurability of an extended real function.

Assumption 2 (Measurable function)**.** *For all $t \in [\![0, T-1]\!]$ and for all nonnegative measurable extended real-valued function $\varphi : \mathbb{H}_{t+1} \to [0, +\infty]$, the extended real-valued function*

$$h_t \mapsto \inf_{u_t \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \varphi(h_t, u_t, w_{t+1}) \rho_{t:t+1}(h_t, dw_{t+1}) \tag{2.9}$$

is measurable[3] *from $(\mathbb{H}_t, \mathcal{H}_t)$ to $[0, +\infty]$.*

Assumption 3 (Measurable selection)**.** *For all $t \in [\![0, T-1]\!]$, there exists a* measurable selection,[4] *that is, a measurable mapping*

$$\gamma_t^\star : (\mathbb{H}_t, \mathcal{H}_t) \to (\mathbb{U}_t, \mathcal{U}_t) \tag{2.10a}$$

such that

$$\gamma_t^\star(h_t) \in \arg\min_{u_t \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} V_{t+1}(h_t, u_t, w_{t+1}) \rho_{t:t+1}(h_t, \mathrm{d}w_{t+1}) \ , \tag{2.10b}$$

where the extended real-valued function V_{t+1} is given by (2.8).

For $t \in [\![0, T]\!]$, let $\mathbb{L}^0_+(\mathbb{H}_t, \mathcal{H}_t)$ be the space of nonnegative measurable extended real-valued functions over $\mathbb{H}_t$.

Definition 19. *For $t \in [\![0, T-1]\!]$, we define the* Bellman operator

$$\mathcal{B}_{t+1:t} : \mathbb{L}^0_+(\mathbb{H}_{t+1}, \mathcal{H}_{t+1}) \to \mathbb{L}^0_+(\mathbb{H}_t, \mathcal{H}_t) \tag{2.11a}$$

such that, for all $\varphi \in \mathbb{L}^0_+(\mathbb{H}_{t+1}, \mathcal{H}_{t+1})$ and for all $h_t \in \mathbb{H}_t$,

$$\big(\mathcal{B}_{t+1:t}\varphi\big)(h_t) = \inf_{u_t \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \varphi(h_t, u_t, w_{t+1}) \rho_{t:t+1}(h_t, dw_{t+1}) \ . \tag{2.11b}$$

Since $\varphi \in \mathbb{L}^0_+(\mathbb{H}_{t+1}, \mathcal{H}_{t+1})$, we have that $\mathcal{B}_{t+1:t}\varphi$ is a well defined nonnegative extended real-valued function and, by Assumption 2, we know that $\mathcal{B}_{t+1:t}\varphi$ is a measurable extended real-valued function, hence belongs to $\mathbb{L}^0_+(\mathbb{H}_t, \mathcal{H}_t)$.

We now state a dynamic programming equation, without requiring independence assumption between the uncertainties.

Proposition 20. *The value functions in (2.8) satisfy the* Bellman equation*, also called* (stochastic) dynamic programming equation*:*

$$V_T = j \ , \tag{2.12a}$$

$$V_t = \mathcal{B}_{t+1:t} V_{t+1} \ , \quad \text{for } t = T-1, \dots, 0 \ . \tag{2.12b}$$

Moreover, a solution to any Problem (2.7) — that is, whatever the index $t \in [\![0, T-1]\!]$ and the parameter $h_t \in \mathbb{H}_t$ — is any history feedback $\gamma^\star = \{\gamma_s^\star\}_{s \in [\![t, T-1]\!]}$ defined by the collection of mappings $\gamma_s^\star$ in (2.10).

[3] This is a delicate issue, treated in [36].

[4] See [36] and [46] for background on measurable selections.

Discretization of measurable spaces with partitions

The numerical implementation of Proposition 20 (stochastic dynamic programming algorithm) often requires the discretization of infinite sets such as the decision sets $\mathbb{U}_t$ and the uncertainty sets $\mathbb{W}_t$ (and hence the history sets $\mathbb{H}_t$ defined in §2.2.1), as well as their associated σ-fields. We present here a tool based on quantization in order to obtain such a discretization (see [38, Chap. 6] for further details).

Let $(\mathbb{W}, \mathcal{W})$ be a measurable space. To discretize $(\mathbb{W}, \mathcal{W})$, we use the following steps.

- We introduce a measurable mapping $Q : (\mathbb{W}, \mathcal{W}) \to (\mathbb{W}, \mathcal{W})$ such that its range $Q(\mathbb{W})$ has a finite cardinality N. The mapping Q is called a *quantifier* and we use the notation $Q(\mathbb{W}) = \{\overline{w}^1, \dots, \overline{w}^N\}$.

- Then, we consider the finite partition $\big(W^1, \dots, W^N\big)$ of $\mathbb{W}$ induced by the quantifier Q, that is, $W^i = Q^{-1}(\overline{w}^i)$ for $i = 1, \dots, N$, and the associated σ-field $\widetilde{\mathcal{W}} = \sigma\big(W^1, \dots, W^N\big)$ of $\mathbb{W}$, made of (finite) unions of the elements of any subfamily of $\big(W^1, \dots, W^N\big)$.

The discretization of the measurable space $(\mathbb{W}, \mathcal{W})$ consists in replacing it by $(\mathbb{W}, \widetilde{\mathcal{W}})$. It is a discretization in the sense that the σ-field $\widetilde{\mathcal{W}}$ is finitely generated. Moreover, for any measurable function $j : (\mathbb{W}, \mathcal{W}) \to (\mathbb{R}, \mathcal{B}^{\mathrm{o}}_{\mathbb{R}})$, we are able to construct a measurable function $\widetilde{j} : (\mathbb{W}, \widetilde{\mathcal{W}}) \to (\mathbb{R}, \mathcal{B}^{\mathrm{o}}_{\mathbb{R}})$, namely $\widetilde{j} = j \circ Q$, which only involves the values of j on the finite set $\{\overline{w}^1, \dots, \overline{w}^N\}$.

A template to design online policies

The exact resolution of Bellman equations (2.12) is generally out of reach, as is the exact computation of the optimal Bellman policies $\{\gamma_t^\star\}_{t \in [\![0, T-1]\!]}$ given by Proposition 20. Practioners use Equations (2.8), (2.10) and (2.12) as *templates* to design *online* policies $\{\gamma_t\}_{t \in [\![0, T-1]\!]}$. By *online policy*, we mean a tool allowing to compute a decision at time t knowing the history h_t. Here, we sketch two design schemes that lead to two classes of online policies: the cost-to-go policies and the lookahead policies.

Cost-to-go policies. *Cost-to-go policies* use Equation (2.10) as a template to compute a decision u_t at time t. For this purpose, three ingredients have to be chosen as follows.

a) A subfield $\widetilde{\mathcal{W}}_{t+1}$ of $\mathcal{W}_{t+1}$, corresponding to noise discretization as introduced in 2.2.2, from which we build the subfield $\widetilde{\mathcal{H}}_{t+1}$ of $\mathcal{H}_{t+1}$ by

$$\widetilde{\mathcal{H}}_{t+1} = \mathcal{H}_t \otimes \mathcal{U}_t \otimes \widetilde{\mathcal{W}}_{t+1} \,. \tag{2.13}$$

b) A measurable cost-to-go function $\widetilde{V}_{t+1} : (\mathbb{H}_{t+1}, \widetilde{\mathcal{H}}_{t+1}) \to [0, +\infty]$.

c) A stochastic kernel $\widetilde{\rho}_{t:t+1} : (\mathbb{H}_t, \widetilde{\mathcal{H}}_t) \times \to [0, 1]$.

At time $t \in [\![0, T-1]\!]$, for a given history h_t, the corresponding value of the cost-to-go policy is obtained as a solution of the following *one-stage optimization problem*[5]

$$\gamma_t(h_t) \in \operatorname*{arg\,min}_{u_t \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \widetilde{V}_{t+1}(h_t, u_t, w_{t+1}) \widetilde{\rho}_{t:t+1}(h_t, dw_{t+1}) \ . \tag{2.14}$$

Problem (2.14) must be solved each time a decision has to be taken (see Sect. 2.5 for examples of using a policy). We note that the history h_t, belonging to the measurable space $(\mathbb{H}_t, \mathcal{H}_t)$, only acts as a parameter in Problem (2.14).

Lookahead policies. Whereas cost-to-go policies take inspiration from Equation (2.10), *lookahead policies* use directly Equation (2.8) as a template.

For this purpose, five ingredients have to be chosen as follows.

a) A lookahead horizon $\widetilde{T} \in [\![t+1, T]\!]$.

b) A sequence $\{\widetilde{\mathcal{W}}_s\}_{s \in [\![t+1, \widetilde{T}]\!]}$ of σ-fields, corresponding to noise discretization as introduced in 2.2.2, such that $\widetilde{\mathcal{W}}_s \subset \mathcal{W}_s$, from which we iteratively build a new sequence of σ-fields $\{\widetilde{\mathcal{H}}_s\}_{s \in [\![t+1, \widetilde{T}]\!]}$ by

$$\widetilde{\mathcal{H}}_t = \mathcal{H}_t \ , \tag{2.15a}$$

$$\widetilde{\mathcal{H}}_{s+1} = \widetilde{\mathcal{H}}_s \otimes \mathcal{U}_s \otimes \widetilde{\mathcal{W}}_{s+1} \ , \quad s = t, \dots, \widetilde{T}-1 \ . \tag{2.15b}$$

By construction, $\widetilde{\mathcal{H}}_s$ is a subfield of the history field $\mathcal{H}_s$ defined in (2.1) for each $s \in [\![t+1, \widetilde{T}]\!]$.

c) A sequence $\{\widetilde{\mathcal{H}}^{\Gamma}_s\}_{s \in [\![t+1, \widetilde{T}-1]\!]}$ of σ-fields, called *policy subfields*, such that

$$\widetilde{\mathcal{H}}^{\Gamma}_s \subset \widetilde{\mathcal{H}}_s \ , \ \ \forall s \in [\![t+1, \widetilde{T}-1]\!] \ , \tag{2.16}$$

that makes it possible to define the set of admissible history feedback policies

$$\widetilde{\Gamma}_{t+1:\widetilde{T}-1} = \Big\{ (\widetilde{\gamma}_{t+1}, \dots, \widetilde{\gamma}_{\widetilde{T}-1}) \mid \widetilde{\gamma}_s : (\mathbb{H}_s, \widetilde{\mathcal{H}}^{\Gamma}_s) \to (\mathbb{U}_s, \mathcal{U}_s) \ , \\ \widetilde{\gamma}_s^{-1}(\mathcal{U}_s) \subset \widetilde{\mathcal{H}}^{\Gamma}_s \ , \ \ \forall s \in [\![t+1, \widetilde{T}-1]\!] \Big\} \ . \tag{2.17}$$

A generic element of $\widetilde{\Gamma}_{t+1:\widetilde{T}-1}$ will be denoted $\widetilde{\gamma}_{t+1:\widetilde{T}-1}$.

d) A sequence of stochastic kernels $\{\widetilde{\rho}_{s:s+1}\}_{s \in [\![t, \widetilde{T}-1]\!]}$, with

$$\widetilde{\rho}_{s:s+1} : (\mathbb{H}_s, \widetilde{\mathcal{H}}^{\rho}_s) \times \widetilde{\mathcal{W}}_{s+1} \to [0, 1] \ ,$$

where $\{\widetilde{\mathcal{H}}^{\rho}_s\}_{s \in [\![t, \widetilde{T}-1]\!]}$ is a sequence of σ-fields such that

$$\widetilde{\mathcal{H}}^{\rho}_s \subset \widetilde{\mathcal{H}}_s \ , \ \ \forall s \in [\![t, \widetilde{T}-1]\!] \ . \tag{2.18}$$

[5]Practitioners often replace the original control set $\mathbb{U}_t$ by another set $\widetilde{\mathbb{U}}_t$ in order to make easier the minimization in (2.14). This minor modification is not considered in this paper.

e) A measurable lookahead cost-to-go function $\widetilde{V}_{\widetilde{T}} : (\mathbb{H}_{\widetilde{T}}, \widetilde{\mathcal{H}}_{\widetilde{T}}) \to [0, +\infty]$.

Using the first four ingredients, we are able to build the lookahead stochastic kernel

$$\widetilde{\rho}^{\widetilde{\gamma}}_{t:\widetilde{T}} : (\mathbb{H}_t, \widetilde{\mathcal{H}}_t) \times \widetilde{\mathcal{H}}_{\widetilde{T}} \to [0, 1] \, ,$$

for any admissible history feedback policy $\widetilde{\gamma}_{t+1:\widetilde{T}-1} \in \widetilde{\Gamma}_{t+1:\widetilde{T}-1}$ as in (2.17) (see Definition 22 for this contruction: restricting both the measurability of the admissible policies by (2.16) and the measurability of the stochastic kernels by (2.18) does not change the construction).

At time $t \in [\![0, T-1]\!]$, for a given history h_t belonging to the measurable space $(\mathbb{H}_t, \mathcal{H}_t)$, the corresponding value of the lookahead policy is obtained as the solution of the following *multistage optimization problem*:

$$\gamma_t(h_t) \in \operatorname*{arg\,min}_{u_t \in \mathbb{U}_t} \min_{\widetilde{\gamma}_{t+1:\widetilde{T}-1} \in \widetilde{\Gamma}_{t+1:\widetilde{T}-1}} \int_{\mathbb{H}_{\widetilde{T}}} \widetilde{V}_{\widetilde{T}}(h_t, u_t, w_{t+1}, h_{t+1:\widetilde{T}}) \widetilde{\rho}^{\widetilde{\gamma}}_{t:\widetilde{T}}(h_t, \mathrm{d}h_{\widetilde{T}}) \, . \tag{2.19}$$

As in §2.2.2, Problem (2.19) must be solved to take a decision at time t. Again, the history h_t only acts as a parameter in this problem.

2.3 A template for lookahead policies

In Sect. 2.2, we have presented an overview of the different ingredients required to formulate and solve stochastic optimization problems, and we have introduced two templates to design *online* policies: the class of *cost-to-go* policies and the class of *lookahead* policies.

In this section, we focus on lookahead policies. We first depict the general structure of lookahead policies in §2.3.1, and then frame three classical algorithms (model predictive control, open-loop feedback control and stochastic programming) in §2.3.2 using the lookahead template.

2.3.1 Design of lookahead policies

We presented in §2.2.2 the ingredients used to devise lookahead policies. We now discuss these ingredients more in detail.

a) **Choosing the lookahead horizon.** The lookahead horizon $\widetilde{T}$ belongs to $[\![t+1, T]\!]$. Here are two classical choices.

   Rolling horizon. We choose a lookahead time step Δt and we set $\widetilde{T} = \min\{t + \Delta t, T\}$.

   Shrinking horizon. In that case, the chosen lookahead horizon $\widetilde{T}$ does not depend on t. Most often, practitioners choose $\widetilde{T} = T$.

b) **Choosing uncertainty subfields and stochastic kernels.** The choices for the sequences of subfields $\{\widetilde{\mathcal{W}}_s\}_{s \in [\![t+1, \widetilde{T}]\!]}$ and of stochastic kernels $\{\widetilde{\rho}_{s:s+1}\}_{s \in [\![t, \widetilde{T}-1]\!]}$ model the *views* of the decision-maker regarding the uncertainties after time $t+1$. We present hereunder two classical situations.

Finite scenario tree. For each s in the lookahead horizon $[\![t+1, \widetilde{T}]\!]$, we discretize the measurable space $(\mathbb{W}_s, \mathcal{W}_s)$ as follows (see §2.2.2).

- We first choose a quantifier $Q_s : (\mathbb{W}_s, \mathcal{W}_s) \to (\mathbb{W}_s, \mathcal{W}_s)$ such that its range has a finite cardinality N_s. This range is denoted $Q_s(\mathbb{W}_s) = \{\overline{w}_s^1, \dots, \overline{w}_s^{N_s}\}$.
- We then build the σ-field $\widetilde{\mathcal{W}}_s = \sigma(Q_s)$ generated by the finite partition $\big(W_s^1, \dots, W_s^{N_s}\big) = \big(Q_s^{-1}(\overline{w}_s^1), \dots, Q_s^{-1}(\overline{w}_s^{N_s})\big)$ of $\mathbb{W}_s$ induced by Q_s:

$$\widetilde{\mathcal{W}}_s = \sigma(Q_s) = \sigma\big(Q_s^{-1}(\overline{w}_s^1), \dots, Q_s^{-1}(\overline{w}_s^{N_s})\big) = \sigma\big(W_s^1, \dots, W_s^{N_s}\big) \; . \tag{2.20}$$

Following Equation (2.15), we build the lookahead history fields

$$\widetilde{\mathcal{H}}_s = \mathcal{H}_t \otimes \bigotimes_{r=t}^{s-1} \big(\mathcal{U}_r \otimes \widetilde{\mathcal{W}}_{r+1}\big) \; , \quad s \in [\![t+1, \widetilde{T}]\!] \; . \tag{2.21a}$$

We also build the following (kernel) subfields $\widetilde{\mathcal{H}}_s^\rho$ of $\widetilde{\mathcal{H}}_s$:

$$\widetilde{\mathcal{H}}_s^\rho = \mathcal{H}_t \otimes \bigotimes_{r=t}^{s-1} \big(\{\varnothing, \mathbb{U}_r\} \otimes \widetilde{\mathcal{W}}_{r+1}\big) \; , \quad s \in [\![t+1, \widetilde{T}]\!] \; . \tag{2.21b}$$

We then choose a sequence of stochastic kernels $\{\widetilde{\rho}_{s:s+1}\}_{s \in [\![t, \widetilde{T}-1]\!]}$, with

$$\widetilde{\rho}_{s:s+1} : (\mathbb{H}_s, \widetilde{\mathcal{H}}_s^\rho) \times \widetilde{\mathcal{W}}_{s+1} \to [0,1] \; . \tag{2.22}$$

Note, on the one hand, that the stochastic kernel $\widetilde{\rho}_{s:s+1}$ is only able to measure sets belonging to the subfield (2.20) generated by the partition $\big(W_{s+1}^1, \dots, W_{s+1}^{N_{s+1}}\big)$, and, on the other hand, that the stochastic kernel $\widetilde{\rho}_{s:s+1}$ does not depend on the previous controls $(u_t, \dots, u_{s-1})$ as the result of the choice of the subfield $\widetilde{\mathcal{H}}_s^\rho$ in (2.21b). This last property is of paramount importance to be able to build a scenario tree.

Single scenario. It is the case of a finite scenario tree, the range of each quantifier Q_s reduces to a singleton $\{\overline{w}_s\}$. Then, the sequence $(\overline{w}_{t+1}, \dots, \overline{w}_{\widetilde{T}})$ is the unique scenario of the tree. The partition induced by Q_s consists of the set $\mathbb{W}_s$ itself, and the associated σ-field $\widetilde{\mathcal{W}}_s$ is the trivial σ-field $\{\varnothing, \mathbb{W}_s\}$. The lookahead history field $\widetilde{\mathcal{H}}_s$ (as in (2.21a)) is, accordingly,

$$\widetilde{\mathcal{H}}_s = \mathcal{H}_t \otimes \bigotimes_{r=t}^{s-1} \big(\mathcal{U}_r \otimes \{\varnothing, \mathbb{W}_{r+1}\}\big) \; , \tag{2.23a}$$

and the kernel subfield $\widetilde{\mathcal{H}}_s^\rho$ in (2.21b) is now

$$\widetilde{\mathcal{H}}_s^\rho = \mathcal{H}_t \otimes \bigotimes_{r=t}^{s-1} \big(\{\varnothing, \mathbb{U}_r\} \otimes \{\varnothing, \mathbb{W}_{r+1}\}\big) \; . \tag{2.23b}$$

The stochastic kernel

$$\widetilde{\rho}_{s:s+1} : (\mathbb{H}_s, \widetilde{\mathcal{H}}_s^\rho) \times \{\varnothing, \mathbb{W}_{s+1}\} \to [0,1] \; , \tag{2.24}$$

is only able to measure the set $\mathbb{W}_{s+1}$ (the associated probability weight being equal to 1), hence representing a deterministic view of the future. It does not depend on the previous controls $(u_t, \dots, u_{s-1})$ and on the previous uncertainties $(w_{t+1}, \dots, w_s)$.

c) **Choosing the set of policies.** Here are two examples of sequences of information fields $\{\widetilde{\mathcal{H}}_s^\Gamma\}_{s\in[\![t+1,\widetilde{T}-1]\!]}$ used to define the set of admissible history feedback policies as in (2.17). We recall that, in (2.16), we imposed the compatibility condition $\widetilde{\mathcal{H}}_s^\Gamma \subset \widetilde{\mathcal{H}}_s$, for any $s \in [\![t+1, \widetilde{T}-1]\!]$.

Open-loop policy. An open-loop policy is made of feedbacks $\widetilde{\gamma}_s$, $s \in [\![t+1, \widetilde{T}-1]\!]$, that are measurable w.r.t. the σ-field

$$\widetilde{\mathcal{H}}_s^\Gamma = \mathcal{H}_t \otimes \bigotimes_{r=t}^{s-1} \left(\{\varnothing, \mathbb{U}_r\} \otimes \{\varnothing, \mathbb{W}_{r+1}\}\right) , \quad s \in [\![t+1, \widetilde{T}-1]\!] . \tag{2.25}$$

Otherwise stated, the feedback $\widetilde{\gamma}_s : (\mathbb{H}_s, \widetilde{\mathcal{H}}_s^\Gamma) \to (\mathbb{U}_s, \mathcal{U}_s)$ only depends on h_t, that is, it assigns the same value to all partial histories $h_{t+1:s} = (u_t, w_{t+1}, \dots, u_{s-1}, w_s)$. This is a way to represent an *open-loop control*.

Closed-loop policy. A closed-loop policy takes advantage of all information available at each time step of the lookahead horizon. In that case,

$$\widetilde{\mathcal{H}}_s^\Gamma = \widetilde{\mathcal{H}}_s , \quad s \in [\![t+1, \widetilde{T}-1]\!] , \tag{2.26}$$

as in (2.21a). This is a way to represent a *closed loop control* making use of maximal past information.

d) **Choosing a final cost-to-go.** The cost-to-go is a function $\widetilde{V}_{\widetilde{T}} : (\mathbb{H}_{\widetilde{T}}, \widetilde{\mathcal{H}}_{\widetilde{T}}) \to [0, +\infty]$. Note that the measurability of $\widetilde{V}_{\widetilde{T}}$ is given by the σ-field $\widetilde{\mathcal{H}}_{\widetilde{T}}$ in (2.21a), so that $\widetilde{V}_{\widetilde{T}}$ a priori depend on the past controls $(u_t, \dots, u_{s-1})$.

Practitioners often design $\widetilde{V}_{\widetilde{T}}$ to mimic a final cost at time $\widetilde{T}$. The function $\widetilde{V}_{\widetilde{T}}$ may also be chosen as a penalty to reach a defined target at the end of the lookahead horizon, for instance $\widetilde{V}_{\widetilde{T}}(h) = \alpha(\theta(h) - \widehat{x})^2$ where θ is a function mapping the whole history into a "state" space $\mathbb{X}_{\widetilde{T}}$ and $\widehat{x} \in \mathbb{X}_{\widetilde{T}}$ is the target.

2.3.2 Classical lookahead methods

As detailed in §2.3.1, the design of lookahead policies depends, on the one hand, on the choice of the two sequences of σ-fields $\{\widetilde{\mathcal{W}}_s\}_{s\in[\![t+1,\widetilde{T}]\!]}$ and of stochastic kernels $\{\widetilde{\rho}_{s:s+1}\}_{s\in[\![t,\widetilde{T}-1]\!]}$, and, on the other hand, on the choice of the σ-fields $\{\widetilde{\mathcal{H}}_s^\Gamma\}_{s\in[\![t+1,\widetilde{T}-1]\!]}$ used to set the measurability of the feedbacks $\widetilde{\gamma}_s$ in the set of admissible history feedback policies $\widetilde{\Gamma}_{t+1:\widetilde{T}-1}$ defined by (2.17).

In the sequel, we specialize the design of lookahead policies to the specific following cases: stochastic programming (SP), open-loop feedback control (OLFC) and model predictive control (MPC). The different combinations of σ-fields associated to the stochastic kernels and to the feedbacks are summarized in Table 2.1.

Stochastic programming

In our framework (§2.3.1), we interpret stochastic programming (SP) (see [48]) as a lookahead method with finite partitions to model the noise fields $\{\widetilde{\mathcal{W}}_s\}_{s\in[\![t+1,\widetilde{T}]\!]}$ (see §2.2.2). We introduce stochastic kernels compatible with the finite partitions, thus encoding a scenario tree, with a decision attached to every node of the tree.

		Open-loop	Closed-loop
	Kernel $\widetilde{\mathcal{H}}_s^{\rho}$ \ Policy $\widetilde{\mathcal{H}}_s^{\Gamma}$	$\bigotimes_{r=t}^{s-1}\{\varnothing,\mathbb{U}_r\}\otimes\{\varnothing,\mathbb{W}_{r+1}\}$	$\bigotimes_{r=t}^{s-1}\{\varnothing,\mathbb{U}_r\}\otimes\widetilde{\mathcal{W}}_{r+1}$
Single scenario	$\bigotimes_{r=t}^{s-1}\{\varnothing,\mathbb{U}_r\}\otimes\{\varnothing,\mathbb{W}_{r+1}\}$	MPC	$\varnothing$
Scenario tree	$\bigotimes_{r=t}^{s-1}\{\varnothing,\mathbb{U}_r\}\otimes\widetilde{\mathcal{W}}_{r+1}$	OLFC	SP

Table 2.1: Classification of classic lookahead policies according to kernel and policy σ-fields

Ingredients

a) *Horizon.* $\widetilde{T}\in[\![t+1,T]\!]$ is given. Rolling or shrinking horizons are allowed.

b) *Uncertainty subfields and stochastic kernels.* The measurable spaces $(\mathbb{W}_s,\mathcal{W}_s)$ are discretized using quantifiers Q_s as described in §2.3.1 for the finite scenario tree case. This leads to the sequence $\{\widetilde{\mathcal{H}}_s\}_{s\in[\![t+1,\widetilde{T}]\!]}$ of lookahead σ-fields given by (2.21a). We also introduce the sequence of subfields $\{\widetilde{\mathcal{H}}_s^{\rho}\}_{s\in[\![t+1,\widetilde{T}]\!]}$ given by (2.21b), namely

$$\widetilde{\mathcal{H}}_s^{\rho}=\mathcal{H}_t\otimes\bigotimes_{r=t}^{s-1}\big(\{\varnothing,\mathbb{U}_r\}\otimes\widetilde{\mathcal{W}}_{r+1}\big)\ ,\quad s\in[\![t+1,\widetilde{T}]\!]\ . \tag{2.27a}$$

Then, we choose a sequence $\{\widetilde{\rho}_{s:s+1}\}_{s\in[\![t,\widetilde{T}-1]\!]}$ of stochastic kernels as in (2.22). As already noticed, the stochastic kernel $\widetilde{\rho}_{s:s+1}$ does not depend on the past controls. Moreover, its dependence on the uncertainty w_r is measurable w.r.t. the σ-field $\widetilde{\mathcal{W}}_r$. As the underlying spaces are separable complete metric, by a Doob theorem [39, Chap. 1, p. 18], the stochastic kernel $\widetilde{\rho}_{s:s+1}$ depends on $Q_r(w_r)$ rather than w_r; by a slight abuse of notation, we write

$$\widetilde{\rho}_{s:s+1}\big(h_t,Q_{t+1}(w_{t+1}),\dots,Q_s(w_s),\mathrm{d}w_{s+1}\big)\ . \tag{2.27b}$$

c) *Policies.* We consider closed-loop policies as in (2.26), that is, with information σ-fields $\widetilde{\mathcal{H}}_s^{\Gamma}$ equal to $\widetilde{\mathcal{H}}_s$ as in (2.21a). A standard inductive reasoning shows that the information given by the σ-field $\widetilde{\mathcal{H}}_s^{\Gamma}$ is the same as the one given by the σ-field $\widetilde{\mathcal{H}}_s^{\rho}$ of (2.27a), so that we can limit ourselves to

$$\widetilde{\mathcal{H}}_s^{\Gamma}=\widetilde{\mathcal{H}}_s^{\rho}\ , \tag{2.27c}$$

given by (2.27a). Again, using the measurability property of the feedback $\widetilde{\gamma}_s:(\mathbb{H}_s,\widetilde{\mathcal{H}}_s^{\Gamma})\to(\mathbb{U}_s,\mathcal{U}_s)$ (see (2.17)), we are able to represent it as a function only depending on the past quantified uncertainties:

$$\widetilde{\gamma}_s\big(h_t,Q_{t+1}(w_{t+1}),\dots,Q_s(w_s)\big)\ . \tag{2.27d}$$

d) *Final cost-to-go.* The final cost $\widetilde{V}_{\widetilde{T}}:(\mathbb{H}_{\widetilde{T}},\widetilde{\mathcal{H}}_{\widetilde{T}})\to[0,+\infty]$ is chosen as a measurable function w.r.t. $\widetilde{\mathcal{H}}_{\widetilde{T}}$. With the same abuse of notation as for the stochastic kernels, the cost-to-go is written

$$\widetilde{V}_{\widetilde{T}}\big(h_t,u_t,Q_{t+1}(w_{t+1}),\dots,u_{\widetilde{T}-1},Q_s(w_{\widetilde{T}})\big)\ . \tag{2.27e}$$

Discretized problem. We now write a discretized version of Problem (2.19). Since $Q_s(w_s) \in \{\overline{w}_s^1, \dots, \overline{w}_s^{N_s}\}$, all the expressions (2.27) constituting the stochastic programming model only depend on the values $\overline{w}_s^i$. We denote by

$$\pi_{s+1}(h_t, Q_{t+1}(w_{t+1}), \dots, Q_s(w_s), \overline{w}_{s+1}^i) \ ,$$

the probability weight $\widetilde{\rho}_{s:s+1}\big(h_t, Q_{t+1}(w_{t+1}), \dots, Q_s(w_s), W_{s+1}^i\big)$, where W_{s+1}^i is an atom of the finite partition of $\mathbb{W}_{s+1}$ generating the σ-field $\widetilde{\mathcal{W}}_{s+1}$ as in (2.20). Then Problem (2.19) is rewritten as

$$\begin{aligned}\arg\min_{u_t \in \mathbb{U}_t} \min_{\widetilde{\gamma}_{t+1:\widetilde{T}-1} \in \widetilde{\Gamma}_{t+1:\widetilde{T}-1}} \sum_{i_{t+1}=1}^{N_{t+1}} \pi_{t+1}(h_t, \overline{w}_{t+1}^{i_{t+1}}) \ \dots \sum_{i_{\widetilde{T}}=1}^{N_{\widetilde{T}}} \pi_{\widetilde{T}}(h_t, \overline{w}_{t+1}^{i_{t+1}}, \dots, \overline{w}_{\widetilde{T}}^{i_{\widetilde{T}}}) \\ \widetilde{V}_{\widetilde{T}}\big(h_t, u_t, \overline{w}_{t+1}^{i_{t+1}}, \widetilde{\gamma}_{t+1}(h_t, \overline{w}_{t+1}^{i_{t+1}}), \overline{w}_{t+2}^{i_{t+2}}, \cdots, \widetilde{\gamma}_{\widetilde{T}-1}(h_t, \overline{w}_{t+1}^{i_{t+1}}, \dots, \overline{w}_{\widetilde{T}-1}^{i_{\widetilde{T}-1}}), \overline{w}_{\widetilde{T}}^{i_{\widetilde{T}}}\big) \ . \quad (2.28)\end{aligned}$$

The minimization w.r.t. the feedback $\widetilde{\gamma}_s$ in (2.28) only involves an expression of the form $\widetilde{\gamma}_s(h_t, \overline{w}_{t+1}^{i_{t+1}}, \dots, \overline{w}_s^{i_s})$, namely a control value for each sequence $(h_t, \overline{w}_{t+1}^{i_{t+1}}, \dots, \overline{w}_s^{i_s})$. Therefore, we are in the situation where the uncertainty is discretized using a scenario tree, with a decision attached to each node of the tree, that is, the standard stochastic programming framework. The stochastic programming tree and information structure are depicted in Figure 2.1.

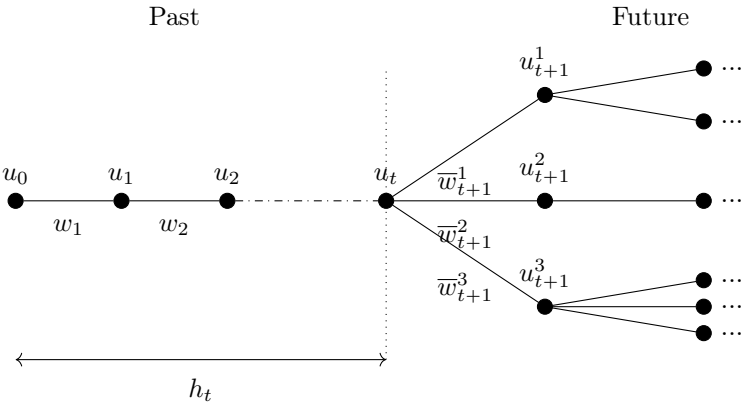

Figure 2.1: SP represents the future as a tree with a decision at each node

Open-loop feedback control

The open-loop feedback control (OLFC) method is described for example in [35, Chap. 6]. In our framework, OLFC is a lookahead method where the uncertainty fields $\{\widetilde{\mathcal{W}}_s\}_{s \in [\![t+1, \widetilde{T}]\!]}$, and the stochastic kernels $\{\widetilde{\rho}_{s:s+1}\}_{s \in [\![t, \widetilde{T}-1]\!]}$, as presented in §2.3.1, are chosen as in the stochastic programming approach in §2.3.2. The major difference between OLFC and SP is that the decisions are open-loop in OLFC, that is, a decision is attached to each stage of the tree rather than to each node as in SP.

Ingredients

a) *Horizon.* $\widetilde{T} \in [\![t+1, T]\!]$ is given. Rolling or shrinking horizons are allowed.

b) *Uncertainty subfields and stochastic kernels.* The measurable spaces $(\mathbb{W}_s, \mathcal{W}_s)$ are discretized as in the stochastic programming model, leading to sequences $\{\widetilde{\mathcal{H}}_s\}_{s\in[\![t+1,\widetilde{T}]\!]}$ and $\{\widetilde{\mathcal{H}}^{\rho}_s\}_{s\in[\![t+1,\widetilde{T}]\!]}$ of lookahead σ-fields given by (2.21a) and (2.21b) respectively. The sequence $\{\widetilde{\rho}_{s:s+1}\}_{s\in[\![t,\widetilde{T}-1]\!]}$ of stochastic kernels is also chosen as in the stochastic programming model, that is, as in (2.22), so that the stochastic kernels do not depend on the past controls, and each can be written in the form

$$\widetilde{\rho}_{s:s+1}\big(h_t, Q_{t+1}(w_{t+1}), \dots, Q_s(w_s), \mathrm{d}w_{s+1}\big) \ . \tag{2.29a}$$

c) *Policies.* OLFC uses open-loop policies as in Equation (2.25), with information σ-fields

$$\widetilde{\mathcal{H}}^{\Gamma}_s = \mathcal{H}_t \times \bigotimes_{r=t}^{s-1} \big(\{\varnothing, \mathbb{U}_r\} \otimes \{\varnothing, \mathbb{W}_{r+1}\}\big) \ .$$

From the measurability properties of any admissible policy $\widetilde{\gamma}_s : (\mathbb{H}_s, \widetilde{\mathcal{H}}^{\Gamma}_s) \to (\mathbb{U}_s, \mathcal{U}_s)$ we deduce that $\widetilde{\gamma}_s$ is a function only depending on h_t, giving the control $u_s = \widetilde{\gamma}_s(h_t)$.

d) *Final cost-to-go.* The final cost $\widetilde{V}_{\widetilde{T}} : (\mathbb{H}_{\widetilde{T}}, \widetilde{\mathcal{H}}_{\widetilde{T}}) \to [0, +\infty]$ is chosen as a measurable function w.r.t. $\widetilde{\mathcal{H}}_{\widetilde{T}}$, and is written

$$\widetilde{V}_{\widetilde{T}}\big(h_t, u_t, Q_{t+1}(w_{t+1}), \dots, u_{\widetilde{T}-1}, Q_s(w_{\widetilde{T}})\big) \ . \tag{2.29b}$$

Discretized problem. To write a discretized version of Problem (2.19), we follow the same path as the one used in the stochastic programming model in §2.3.2. The only difference is that the minimization w.r.t. the feedbacks $\widetilde{\gamma}_s$ in (2.28) only involves expression of the form $\widetilde{\gamma}_s(h_t)$, that is, a control value u_s for each time s. Problem (2.28) becomes

$$\begin{aligned}\underset{u_t\in\mathbb{U}_t}{\arg\min} \min_{u_{t+1:\widetilde{T}-1}\in\mathbb{U}_{t+1:\widetilde{T}-1}} \sum_{i_{t+1}=1}^{N_{t+1}} \pi_{t+1}(h_t, \overline{w}^{i_{t+1}}_{t+1}) \ \dots \sum_{i_{\widetilde{T}}=1}^{N_{\widetilde{T}}} \pi_{\widetilde{T}}(h_t, \overline{w}^{i_{t+1}}_{t+1}, \dots, \overline{w}^{i_{\widetilde{T}}}_{\widetilde{T}}) \\ \widetilde{V}_{\widetilde{T}}\big(h_t, u_t, \overline{w}^{i_{t+1}}_{t+1}, u_{t+1}, \overline{w}^{i_{t+2}}_{t+2}, \dots, u_{\widetilde{T}-1}, \overline{w}^{i_{\widetilde{T}}}_{\widetilde{T}}\big) \ .\end{aligned} \tag{2.30}$$

Therefore, we are in the situation where the uncertainty is discretized using a scenario tree, with a decision attached at each stage of the tree, that is, the open-loop feedback control framework. The open-loop feedback control tree and information structure are depicted in Figure 2.2.

Model predictive control

Model predictive control (MPC) is a well-known method that tackles uncertainties by using deterministic forecasts (see [40] and [34]).

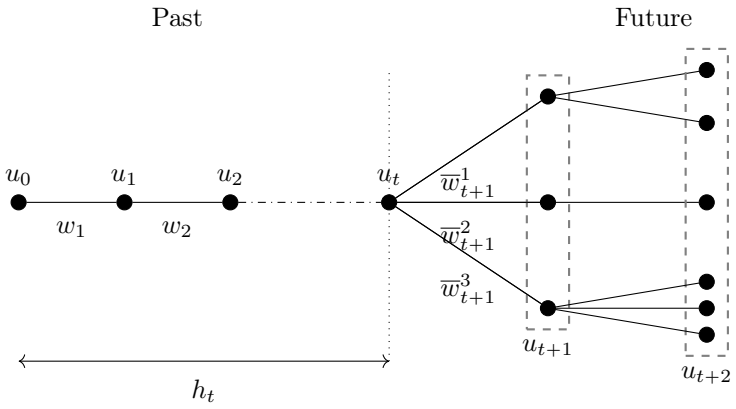

Figure 2.2: OLFC represents the future as a tree with a decision at each time instant (node)

Ingredients

a) *Horizon.* $\widetilde{T} \in [\![t+1, T]\!]$ is given. Rolling or shrinking horizons are allowed.

b) *Uncertainty subfields and stochastic kernels.* The measurable spaces $(\mathbb{W}_s, \mathcal{W}_s)$ are discretized using constant quantifiers Q_s, as described for the single scenario case in §2.3.1. The subfield $\widetilde{\mathcal{W}}_s$ associated to Q_s in (2.20) is thus $\{\varnothing, \mathbb{W}_s\}$. This leads to the sequence $\{\widetilde{\mathcal{H}}_s\}_{s\in[\![t+1,\widetilde{T}]\!]}$ of lookahead σ-fields given by (2.23a), namely

$$\widetilde{\mathcal{H}}_s = \mathcal{H}_t \otimes \bigotimes_{r=t}^{s-1} \big(\mathcal{U}_r \otimes \{\varnothing, \mathbb{W}_{r+1}\}\big) ,$$

and we also introduce the sequence of subfields $\{\widetilde{\mathcal{H}}^\rho_s\}_{s\in[\![t+1,\widetilde{T}]\!]}$ given by (2.23b), namely

$$\widetilde{\mathcal{H}}^\rho_s = \mathcal{H}_t \otimes \bigotimes_{r=t}^{s-1} \big(\{\varnothing, \mathbb{U}_r\} \otimes \{\varnothing, \mathbb{W}_{r+1}\}\big) .$$

We then choose a sequence $\{\widetilde{\rho}_{s:s+1}\}_{s\in[\![t,\widetilde{T}-1]\!]}$ of stochastic kernels as in Equation (2.24). Each stochastic kernel $\widetilde{\rho}_{s:s+1}$ only depends on h_t and is trivial, as it only measures the whole set $\mathbb{W}_{s+1}$ and the emptyset $\varnothing$.

c) *Policies.* MPC uses open-loop policies as in Equation (2.25), with information σ-fields

$$\widetilde{\mathcal{H}}^\Gamma_s = \mathcal{H}_t \times \bigotimes_{r=t}^{s-1} \big(\{\varnothing, \mathbb{U}_r\} \otimes \{\varnothing, \mathbb{W}_{r+1}\}\big) .$$

From the measurability properties of the policy $\widetilde{\gamma}_s : (\mathbb{H}_s, \widetilde{\mathcal{H}}^\Gamma_s) \to (\mathbb{U}_s, \mathcal{U}_s)$ we deduce that $\widetilde{\gamma}_s$ is a function only depending on h_t, so that the history feedback $\widetilde{\gamma}_s(h_t)$ can be identified with a value u_s.

d) *Final cost-to-go.* The final cost $\widetilde{V}_{\widetilde{T}} : (\mathbb{H}_{\widetilde{T}}, \widetilde{\mathcal{H}}_{\widetilde{T}}) \to [0, +\infty]$ is chosen as a measurable function w.r.t. $\widetilde{\mathcal{H}}_{\widetilde{T}}$ and thus is written

$$\widetilde{V}_{\widetilde{T}}\big(h_t, u_t, \overline{w}_{t+1}, \ldots, u_{\widetilde{T}-1}, \overline{w}_{\widetilde{T}}\big) .$$

Discretized problem. According to the ingredients described above, a discretized version of Problem (2.19) is obtained from the OLFC discretized Problem (2.30) when considering a single scenario, that is,

$$\underset{u_t \in \mathbb{U}_t}{\arg\min} \quad \min_{u_{t+1:\widetilde{T}-1} \in \mathbb{U}_{t+1:\widetilde{T}-1}} \widetilde{V}_{\widetilde{T}}\big(h_t, u_t, \overline{w}_{t+1}, u_{t+1}, \overline{w}_{t+2}, \cdots, u_{\widetilde{T}-1}, \overline{w}_{\widetilde{T}}\big) \ . \tag{2.32}$$

Therefore, we are in the situation where the uncertainty is discretized using a single scenario, with a decision attached at each time. Problem (2.32) is a deterministic optimization problem, possibly solvable by proper mathematical programming methods. The MPC scenario and information scheme are depicted in Figure 2.3.

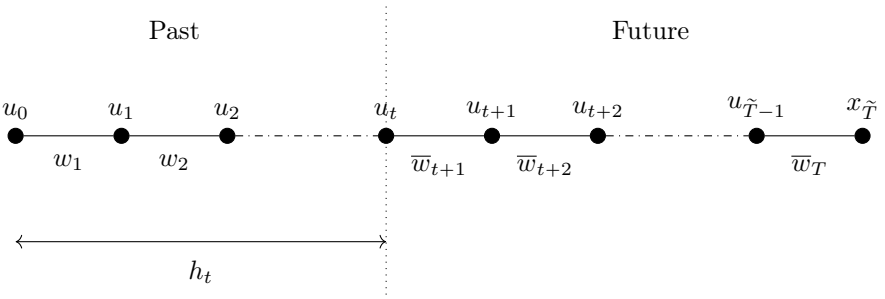

Figure 2.3: MPC models the future with a deterministic forecast

Discussion

In this section, we have detailed a general structure for lookahead optimization problems, and have presented the stochastic programming model, the open-loop feedback control model as a restriction of the SP model, and the model predictive control model itself as a restriction of the OLFC model.

Lookahead algorithms correspond to the *implicit cost-to-go approximation* presented in [34], where the future cost $\widetilde{V}_{t+1}$ is estimated online by restricting the set of admissible policies $\widetilde{\gamma}_s$. Most algorithms that rely on *forecasts* are lookahead algorithms. For instance, we can frame the stochastic model predictive control algorithm either as a open-loop feedback control or as a stochastic programming method, depending on the information structure that defines the set of future policies.

In the following section, we deal with cost-to-go policies, corresponding to the *explicit cost-to-go approximation* of [34].

2.4 A template for cost-to-go policies

We focus in this section on *cost-to-go* policies as obtained by solving Problem (2.14). We first describe the generic template of cost-to-go policies in §2.4.1 and §2.4.2, and then frame three classical algorithms (Stochastic Dynamic Programming, Stochastic Dual Dynamic Programming and Approximate Dynamic Programming) with the template of cost-to-go policies in §2.4.3.

Cost-to-go policies are evaluated on the fly, online, but based on offline computations. This is why, we will put a superscript $^{\mathfrak{on}}$ or$^{\mathfrak{of}}$ on some of the mathematical material introduced in the sequel.

2.4.1 Design of cost-to-go policies

The three ingredients of cost-to-go policies are (see §2.2.2):

a) a subfield $\widetilde{\mathcal{W}}_{t+1}^{\mathfrak{on}}$ of $\mathcal{W}_{t+1}$, from which we build the subfield $\widetilde{\mathcal{H}}_{t+1}^{\mathfrak{on}} = \mathcal{H}_t \otimes \mathcal{U}_t \otimes \widetilde{\mathcal{W}}_{t+1}^{\mathfrak{on}}$ of $\mathcal{H}_{t+1}$,

b) a stochastic kernel $\widetilde{\rho}_{t:t+1}^{\mathfrak{on}} : (\mathbb{H}_t, \widetilde{\mathcal{H}}_t^{\mathfrak{on}}) \times \widetilde{\mathcal{W}}_{t+1}^{\mathfrak{on}} \to [0,1]$,

c) a measurable cost-to-go function $\widetilde{V}_{t+1} : (\mathbb{H}_{t+1}, \widetilde{\mathcal{H}}_{t+1}) \to [0, +\infty]$.

These ingredients are used to compute an online control policy by solving Problem (2.14), that we recall here for convenience:

$$\gamma_t(h_t) \in \operatorname*{arg\,min}_{u_t \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \widetilde{V}_{t+1}(h_t, u_t, w_{t+1}) \widetilde{\rho}_{t:t+1}^{\mathfrak{on}}(h_t, dw_{t+1}) \; . \tag{2.33}$$

The first and second ingredients (subfield and stochastic kernel) have been discussed in the description of lookahead policies (see §2.3), and the procedure to design them is similar for cost-to-go policies. As far as the third ingredient (cost-to-go function) is concerned, explicit cost-to-go functions $\widetilde{V}_{t+1} : (\mathbb{H}_{t+1}, \widetilde{\mathcal{H}}_{t+1}) \to [0, +\infty]$ are required as input. We now detail how to obtain these functions.

2.4.2 Computation of offline cost-to-go functions

To compute the cost-to-go functions $\widetilde{V}_{t+1}$ used in §2.4.1, we solve *offline* backward recursive equations mimicking the Bellman equations (2.12). For this purpose, starting with a given $\widetilde{V}_T$, we design *offline Bellman* operators $\widetilde{\mathcal{B}}_{t+1:t}^{\mathfrak{of}}$ such that

$$\widetilde{V}_t = \widetilde{\mathcal{B}}_{t+1:t}^{\mathfrak{of}} \widetilde{V}_{t+1} \; , \quad \forall \; t = T-1, \ldots, 0 \; . \tag{2.34}$$

The following four ingredients enter the design of the offline Bellman operators $\widetilde{\mathcal{B}}_{t+1:t}^{\mathfrak{of}}$.

1) A sequence of σ-fields $\{\widetilde{\mathcal{W}}_t^{\mathfrak{of}}\}_{t \in [\![0,T]\!]}$, with $\widetilde{\mathcal{W}}_t^{\mathfrak{of}} \subset \mathcal{W}_t$.

2) A sequence of stochastic kernels $\{\widetilde{\rho}_{t:t+1}^{\mathfrak{of}}\}_{t \in [\![0,T-1]\!]}$, $\widetilde{\rho}_{t:t+1}^{\mathfrak{of}} : (\mathbb{H}_t, \mathcal{H}_t) \times \widetilde{\mathcal{W}}_{t+1}^{\mathfrak{of}} \to [0,1]$.

3) A sequence of sets $\{\widetilde{\mathcal{L}}_t\}_{t \in [\![0,T]\!]}$ of extended real-valued functions, with $\widetilde{\mathcal{L}}_t \subset \mathbb{L}_+^0(\mathbb{H}_t, \widetilde{\mathcal{H}}_t)$, the space of nonnegative $\widetilde{\mathcal{H}}_t$-measurable extended real-valued functions over $\mathbb{H}_t$.

4) A sequence of operators $\{\mathfrak{P}_t\}_{t \in [\![0,T]\!]}$, with $\mathfrak{P}_t : \mathbb{L}_+^0(\mathbb{H}_t, \mathcal{H}_t) \to \widetilde{\mathcal{L}}_t$

For compatibility reasons (justified in the following footnote 6) with the design of the cost-to-go policies presented in §2.4.1, we assume that

$$\widetilde{\mathcal{W}}_t^{\mathrm{on}} \subset \widetilde{\mathcal{W}}_t^{\mathrm{of}} \subset \mathcal{W}_t \ . \tag{2.35}$$

Starting with a given cost-to-go function $\widetilde{V}_T \in \widetilde{\mathcal{L}}_T$ at horizon T, the construction in backward time of the operators

$$\widetilde{\mathcal{B}}_{t+1:t} : \widetilde{\mathcal{L}}_{t+1} \to \widetilde{\mathcal{L}}_t \ , \tag{2.36}$$

involves the following two steps at each time t.

- *Designing the offline minimization.* In this step, we use the function $\widetilde{V}_{t+1}$, and the stochastic kernel $\widetilde{\rho}_{t:t+1}^{\mathrm{of}}$ to set up the optimization problem:[6]

$$\widetilde{V}_t^{\mathrm{of}}(h_t) = \inf_{u_t \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \widetilde{V}_{t+1}(h_t, u_t, w_{t+1}) \widetilde{\rho}_{t:t+1}^{\mathrm{of}}(h_t, dw_{t+1}) \ . \tag{2.37}$$

  This defines an operator $\widetilde{\mathcal{B}}_{t+1:t}^{\mathrm{of}} : \widetilde{\mathcal{L}}_{t+1} \to \mathbb{L}_+^0(\mathbb{H}_t, \mathcal{H}_t)$ such that

$$\widetilde{V}_t^{\mathrm{of}} = \widetilde{\mathcal{B}}_{t+1:t}^{\mathrm{of}} \widetilde{V}_{t+1} \ . \tag{2.38a}$$

- *Designing the operator domain.* The purpose of this step is to obtain a cost-to-go function $\widetilde{V}_t \in \widetilde{\mathcal{L}}_t$. This is achieved by using the operator $\mathfrak{P}_t : \mathbb{L}_+^0(\mathbb{H}_t, \mathcal{H}_t) \to \widetilde{\mathcal{L}}_t$, namely

$$\widetilde{V}_t = \mathfrak{P}_t \widetilde{V}_t^{\mathrm{of}} \ . \tag{2.38b}$$

Thanks to these two steps, we define the operator $\widetilde{\mathcal{B}}_{t+1:t} : \widetilde{\mathcal{L}}_{t+1} \to \widetilde{\mathcal{L}}_t$ by

$$\widetilde{\mathcal{B}}_{t+1:t} = \mathfrak{P}_t \circ \widetilde{\mathcal{B}}_{t+1:t}^{\mathrm{of}} \ . \tag{2.39}$$

Then, we are able to compute the cost-to-go functions $\{\widetilde{V}_t\}_{t \in [\![0,T]\!]}$ in a backward manner using the recursive equation (2.34).

Remark 21. *In the above construction, we have introduced the operators $\mathfrak{P}_t$ and $\mathcal{B}_{t+1:t}^{\mathrm{of}}$ for the sake of clarity. Most of the time, the computation of the whole function $\widetilde{V}_t^{\mathrm{of}}$ remains out of reach (but Problem (2.37) is such that it is easy to compute the value of $\widetilde{V}_t^{\mathrm{of}}$ for a given value of the history h_t). In fact, practitioners compute directly the composed operator $\widetilde{\mathcal{B}}_{t+1:t}$ instead of each operator $\mathfrak{P}_t$ and $\widetilde{\mathcal{B}}_{t+1:t}^{\mathrm{of}}$ separately. This is made clearer in the examples of the next paragraph.*

2.4.3 Classical cost-to-go methods

We show that we are able to frame well-known algorithms (stochastic dynamic programming, stochastic dual dynamic programming and approximate dynamic programming) only by describing their corresponding operators $\widetilde{\mathcal{B}}_{t+1:t}$.

We sketch the algorithms in the history framework of §2.2.2, and not in the traditional state space framework. In other words, we replace the state spaces (generally of fixed dimension) of the traditional approach by history spaces (of increasing dimension). Of course there is no difficulty to present all the algorithms in the traditional state space framework.

[6] Note that the compatibility condition (2.35) ensures that the integral in (2.37) is well defined.

Stochastic dynamic programming

The stochastic dynamic programming (SDP) algorithm (see [35]-[45]) computes a set of cost-to-go functions using a discretization of the history spaces. More precisely, at time t, SDP chooses a finite grid of size N_t in $\mathbb{H}_t$, that is, a set $\{h_t^i\}_{i\in[\![1,N_t]\!]}$ of N_t elements of $\mathbb{H}_t$. Then, we introduce the trace operator $\mathfrak{T}_t^{N_t} : \mathbb{L}_+^0(\mathbb{H}_t, \mathcal{H}_t) \to \big(\mathbb{H}_t \times [0,+\infty]\big)^{N_t}$ (that associates the representation $\{(h_t^i, \phi_t(h_t^i))\}_{i\in[\![1,N_t]\!]}$ with any function $\phi_t \in \mathbb{L}_+^0(\mathbb{H}_t, \mathcal{H}_t)$), and a regression-interpolation operator $\mathfrak{R}_t : \big(\mathbb{H}_t \times [0,+\infty]\big)^{N_t} \to \widetilde{\mathcal{L}}_t$ (as an operator that interpolates the values on the grid $\{h_t^i\}_{i\in[\![1,N_t]\!]}$ to obtain a function of $\widetilde{\mathcal{L}}_t$). Then, we define the operator $\mathfrak{P}_t$ (introduced as the fourth ingredient in §2.4.2) by $\mathfrak{P}_t = \mathfrak{R}_t \circ \mathfrak{T}_t^{N_t}$. In practice, SDP directly computes $\mathfrak{T}_t^{N_t}\widetilde{V}_t^{\mathbf{cf}}$, that is, it solves Problem (2.37) at each point of the grid $\{h_t^i\}_{i\in[\![1,N_t]\!]}$. Then, the regression-interpolation operator $\mathfrak{R}_t$ usually consists of building a piecewise constant or piecewise linear function from the N_t available points.

Stochastic dual dynamic programming

Stochastic dual dynamic programming (SDDP) takes advantage of a well-known result in convex analysis that allows to represent any proper convex l.s.c. function as a supremum of affine functions. The idea behind SDDP was introduced in [49] and extended further in [42]. We refer to [47] for a recent overview of the algorithm.

We suppose that the future cost-to-go $\widetilde{V}_{t+1}$ in (2.34) is convex. Then, SDDP approximates *iteratively* the offline Bellman operators (2.37) by a supremum of *affine functions*. We accordingly define the sets $\widetilde{\mathcal{L}}_t$ by

$$\widetilde{\mathcal{L}}_t = \big\{\phi : (\mathbb{H}_t, \widetilde{\mathcal{H}}_t) \to [0,+\infty] \mid \exists N \in \mathbb{N}\ ,\ \exists(\lambda^i, \beta^i)_{i\in[\![1,N]\!]}\ ,\ \phi(h) = \max_{i=1\cdots N} \big\langle \lambda^i\, , h\big\rangle + \beta^i\big\}\ . \tag{2.40}$$

SDDP computes the approximations of the Bellman value functions $\widetilde{V}_t$ defined in (2.34) by running an alternation of *forward* and *backward* passes. At iteration k, we suppose given a family of cost-to-go functions $\{\widetilde{V}_t^k\}_{t\in[\![0,T]\!]}$. Then, SDDP refines the value functions as follows.

- *Forward pass.* Let $w^k = (w_0^k, \dots, w_T^k) \in \mathbb{W}_0 \times \cdots \times \mathbb{W}_T$ be a noise scenario. SDDP computes a history trajectory $h^k = (h_0^k, \dots, h_T^k)$ along the scenario w^k by using the cost-to-go policy induced by (2.33) as follows: for $t = 0, \dots, T-1$, compute

$$\gamma_t^k(h_t^k) \in \operatorname*{arg\,min}_{u_t\in\mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \widetilde{V}_{t+1}^k(h_t^k, u_t, w_{t+1})\widetilde{\rho}_{t:t+1}(h_t^k, dw_{t+1})\ , \tag{2.41}$$

  and set $h_{t+1}^k = (h_t^k, \gamma_t^k(h_t^k), w_{t+1}^k)$.

- *Backward pass.* SDDP refines the approximation along the history trajectory h^k, in a backward manner. Let $\widetilde{V}_T^k = \widetilde{V}_T$ be given. For $t = T-1, \dots, 0$, the algorithm computes a new affine minorant of $\widetilde{V}_t^{\mathbf{cf}} = \mathcal{B}_{t+1:t}^{\mathbf{cf}}\widetilde{V}_{t+1}^k$ at point h_t^k by solving Problem (2.37). This operator which generates a cut is denoted $\mathfrak{S}_t(h_t^k)$. The update of the cost-to-go functions consists in adding the new cut to the existing ones:

$$\widetilde{V}_t^{k+1} = \mathfrak{P}_t\widetilde{V}_t^{\mathbf{cf}} = \max\big\{\widetilde{V}_t^k,\ \mathfrak{S}_t(h_t^k) \circ \widetilde{V}_t^{\mathbf{cf}}\big\}\ . \tag{2.42}$$

We observe that at iteration k, the update of the value function (2.42) is written as a maximum of the previous value function and an affine function. Thus, the value function $\widetilde{V}_t^{k+1}$ remains in $\widetilde{\mathcal{L}}_t$ provided that the value function $\widetilde{V}_t^k$ at the previous iteration lies in $\widetilde{\mathcal{L}}_t$ defined in (2.40).

In practice, SDDP never computes $\widetilde{V}_t^{\mathrm{cf}} = \mathcal{B}_{t+1:t}^{\mathrm{cf}} \widetilde{V}_{t+1}^k$, but only the new cut.

Approximate dynamic programming

Approximate dynamic programming (ADP) approximates the cost-to-go functions in finite dimension functional spaces. We refer to [37] and [43] for an extensive overview of ADP. We describe hereafter ADP in the history framework of §2.2.2.

Approximate Dynamic Programming approximates the function $\widetilde{V}_t^{\mathrm{cf}}$ in Equation (2.37) by an element of the set of functions $\widetilde{\mathcal{L}}_t = \mathrm{span}\{\phi_t^1, \ldots, \phi_t^N\}$ spanned by a finite number of functions $\phi_t^1, \ldots, \phi_t^N$. The functions $\phi_t^i : (\mathbb{H}_t, \widetilde{\mathcal{H}}_t) \to [0, +\infty]$ form a finite basis (e.g. spline or quadratic functions). The cost-to-go function is $\widetilde{V}_t = \mathfrak{P}_t \widetilde{V}_t^{\mathrm{cf}}$, with $\mathfrak{P}_t = \mathrm{proj}_{\widetilde{\mathcal{L}}_t}$, namely the projection on the subspace $\widetilde{\mathcal{L}}_t$. The projection consists in computing a finite number of parameters $\{\alpha_t^i\}_{i \in [\![1,N]\!]}$ such that $\widetilde{V}_t(h_t) = \sum_{i=1}^N \alpha_t^i \phi_t^i(h_t)$, $\forall h_t \in \mathbb{H}_t$.

Difference between offline algorithms

The only difference between the algorithms that compute a set of cost-to-go offline are the backward operators $\widetilde{\mathcal{B}}_{t+1:t}$ used. The different choices are summarized in Table 2.2.

Algo	$\widetilde{\mathcal{B}}_{t+1:t}$
SDP	Discretize then interpolate
SDDP	Accumulate cuts using convex duality
ADP	Project on a basis of functions

Table 2.2: The approximate Bellman operators of SDP, SDDP and ADP

2.5 Assessment of online policies

Policies $\gamma_t : (\mathbb{H}_t, \mathcal{H}_t) \to (\mathbb{U}_t, \mathcal{U}_t)$ are mappings that take as argument a history h_t to return a decision u_t. Depending on the ingredients chosen as explained in §2.2.2, the performances of policies may differ. We detail hereafter a procedure to compare the performances of different policies.

Usually, the decision maker has used a sample $\mathfrak{W}^{opt} = \{w_{opt}^1, \ldots, w_{opt}^M\}$ of uncertainty scenarios to design the online policies (for instance to infer possible probability laws for the future uncertainties). We will call the scenarios in $\mathfrak{W}^{opt}$ *optimization scenarios*. A different set of scenarios $\mathfrak{W}^{sim} = \{w^1, \ldots, w^N\}$ is used during the assessment procedure, and is called the set of *assessment scenarios*.

To ensure that the comparison of different policies remains fair, we must ensure that the set of optimization scenarios does not intersect the set of assessment scenarios. The assessment procedure is *out-of-sample* as soon as $\mathfrak{W}^{opt} \cap \mathfrak{W}^{sim} = \varnothing$.

2.5.1 Simulating the flow induced by a policy along a scenario

We consider a family of history feedbacks $\gamma = (\gamma_0, \dots, \gamma_{T-1}) \in \Gamma_{0:T-1}$, as those given by (2.14) (cost-to-go policy) or (2.19) (lookahead policy), and an uncertainty scenario $w^s = (w_0^s, \dots, w_T^s) \in \mathbb{W}_0 \times \cdots \times \mathbb{W}_T$. The simulator computes stage by stage the value of the *flow* $\Phi_{0:T}^\gamma(w_0^s, \dots, w_T^s) \in \mathbb{H}_T$ along scenario w^s (see Appendix 2.7.1 for the definition of a flow) by

$$\Phi_{0:T}^\gamma(w_0^s, \dots, w_T^s) = \big(w_0^s, \gamma_0(w_0^s), w_1^s, \gamma_1(w_0^s, \gamma_0(w_0^s), w_1^s), \dots, \gamma_{T-1}(h_{T-1}^s), w_T^s\big) = h_T^s \ . \tag{2.43}$$

Then, the cost of policy γ along scenario w^s is

$$j(h_T^s) = j \circ \Phi_{0:T}^\gamma(w^s) \ , \tag{2.44}$$

where $j : \mathbb{H}_T \to \mathbb{R}$ is the cost function in Equation (2.6).

Algorithm 1: Simulation along a scenario

Data: Policy $\gamma = (\gamma_0, \dots, \gamma_{T-1})$, assessment scenario $w^s = (w_0^s, \dots, w_T^s)$
Result: Flow $\Phi_{0:T}^\gamma(w^s) = (w_0^s, u_0^s, w_1^s, u_1^s, \dots, u_{T-1}^s, w_T^s)$
$h_0^s = w_0^s$;
for $t \in [\![0, T-1]\!]$ **do**
  $u_t^s = \gamma_t(h_t^s)$;
  $h_{t+1}^s = (h_t^s, u_t^s, w_{t+1}^s)$;
end

2.5.2 Comparing policies

Once the flow of a given policy γ computed, we can attach a value (expected cost, etc.) to the policy γ, so that we can compare it with other policies.

Assessment of a single policy

Assessment procedure. By computing the flow $\Phi_{0:T}^\gamma$ for a given policy γ along a *single* scenario w^s, we are able to associate a cost value $j \circ \Phi_{0:T}^\gamma(w^s)$. By iterating the procedure along *multiple* scenarios $w^1, \dots, w^N$ we are able to obtain different values $j(h_T^1), \dots, j(h_T^N)$ that can be used to estimate the expected cost of the policy. We depict the assessment procedure in Algorithm 2.

Algorithm 2: Assessment of a policy γ with N scenarios

Data: Policy γ, assessment scenarios $w^1, \dots, w^N$
Result: Cost vector $\big(j(h_T^1), \dots, j(h_T^N)\big)$
for $w^s \in (w^1, \dots, w^N)$ **do**
  $j(h_T^s) = j \circ \Phi_{0:T}^\gamma(w^s)$;
end

It yields a cost vector $(j(h_T^1), \dots, j(h_T^N)) \in \mathbb{R}^N$. We can use some metrics $\mu^N : \mathbb{R}^N \to \mathbb{R}$ (such as the average, the median, the variance, etc) to obtain a real number $\mu^N\big(j(h_T^1), \dots, j(h_T^N)\big)$ characterizing the performance of the policy.

As an illustration, we consider the (quite common) case where μ^N is the arithmetic mean, that is, $\mu^N(x^1,\dots,x^N) = \dfrac{1}{N}\sum_{s=1}^N x^s$. This specific metrics μ^N opens the way to assess the policy's performance by using the law of large numbers and the central limit theorem. The procedure is as follows.

1. Draw a (large) number N of independent scenarios w^s and run Algorithm 2.
2. Compute the sample cost average $m = \frac{1}{N}\sum_{s=1}^N j(h_T^s)$ and the sample cost standard deviation $\sigma^2 = \frac{1}{N-1}\sum_{s=1}^N \big(j(h_T^s) - m\big)^2$.
3. Use the values m and σ to obtain a confidence interval for $\mathbb{E}\big[j\circ\Phi_{0:T}^{\gamma}(\boldsymbol{W}_{0:T})\big]$ thanks to the central limit theorem.

Comparing different policies

We are able to assess M different policies $\gamma^1,\dots,\gamma^M$ along the same set of scenarios $w^1,\dots,w^N$ and compare them altogether with the same metrics μ^N. The best policy is the policy that yields the minimum cost in the vector

$$\left\{\mu^N\Big(\big\{j\circ\Phi_{0:T-1}^{\gamma^m}(w^s)\big\}_{s\in[\![1,N]\!]}\Big)\right\}_{m\in[\![1,M]\!]}. \tag{2.45}$$

2.6 Discussion

In practice, to solve multistage stochastic optimization problems, one does not need the whole theoretical solution, but only needs to compute a proper decision on the fly, as information unfolds. This is the essence of online policies. In this paper, we have offered a complementary view to existing classifications by emphasizing the role of information structures in the design of online policies. We have shown that well-known classical methods can be put in a common framework by using variations on the measurability properties (information patterns) of the ingredients that constitute them.

We sketch hereafter two other classifications — the classifications introduced in [34] and in [44] — and compare them with our own taxonomy. We note that our classification is written with a generic information structure depending on a history process, whereas the two other classifications rely on a state process.

A comparison with the classification of Bertsekas. In [34], the author introduces a unifying suboptimal control framework, in which rollout algorithms and model predictive control stand as a special case of approximate dynamic programming methods. The author frames existing algorithms in two main classes.

1. *Explicit cost-to-go* methods compute offline a sequence of cost-to-go $\{\widetilde{V}_t\}_{t\in[\![0,T]\!]}$ and solve online problems similar to (2.14). The author outlines two main classes of algorithms among these methods.
   - Get cost-to-go $\widetilde{V}_t$ by applying dynamic programming on a simpler problem. This is equivalent to define an operator $\widetilde{\mathcal{B}}_{t+1:t}$ mimicking the Bellman operators of the original problem.

- Use a parametric approximation to approximate the cost-to-go $\widetilde{V}_t$, then tune the approximation by some systematic methods.

2. *Implicit cost-to-go* methods compute online a cost-to-go by solving a problem similar to (2.19). The author distinguishes three classes of algorithms.

   - Rollout algorithm solves Problem (2.19) by using suboptimal/heuristic policies $\gamma_{t+1}, \ldots, \gamma_{\widetilde{T}-1}$ up to a given horizon $\widetilde{T}$.
   - Open-loop feedback control (OLFC) solves Problem (2.19) with open-loop policies.
   - Model predictive control (MPC) is a variant of OLFC where the future scenario is deterministic.

A comparison with the classification of Powell. We exhibit hereafter the link existing between the cost-to-go and lookahead policies we introduced earlier and the classification introduced in [44].

Powell states that there exists four classes of algorithms to design online policies. He frames these four classes as cost-to-go methods or lookahead methods.

1. The *policy functions approximation (PFA)* directly parameterizes the online policy γ_t with a fixed rule or a parametric model depending on history (for instance, γ_t may encode a proportional-integral (PI) discrete controller or a (s, S) policy).

2. The *cost function approximation (CFA)* minimizes a parameterized cost expression of the form
   $\gamma_t(h_t) \in \arg\min_{u_t \in \widetilde{\mathbb{U}}_t} \int_{\mathbb{W}_{t+1}} \big(\sum_{f \in F} \gamma^f \Phi_t^f(h_t, u_t, w_{t+1}) \big) \rho_{t+1}(h_t, dw_{t+1})$, with $\{\Phi_t^f\}_{f \in F}$ a set of basis functions. That corresponds to the approximate dynamic programming (ADP) cost-to-go policies, which use a parametric approximation of the cost-to-go $\widetilde{V}_{t+1} : \mathbb{H}_{t+1} \to \mathbb{R}$, as described in §2.4.3.

3. The *value function approximation (VFA)* approximates the cost-to-go $\widetilde{V}_{t+1}$ and defines the corresponding online policies as in (2.14). This is exactly the template of cost-to-go policies introduced in Sect. 2.4.

4. The *lookahead policies* optimize a multistage problem over a given horizon $\widetilde{T}$, yielding exactly the template of the lookahead policies described in Sect. 2.3.

2.7 Flows and stochastic kernels

2.7.1 Flows

We consider two time instants r and t such that

$$0 \leqslant r < t \leqslant T \, .$$

For a given $(r:t-1)$-history feedback $\gamma = \{\gamma_s\}_{s\in[\![r,t-1]\!]} \in \Gamma_{r:t-1}$, we define the *flow* $\Phi^\gamma_{r:t}$ by

$$\Phi^\gamma_{r:t} :(\mathbb{H}_r, \mathcal{H}_r) \times (\mathbb{W}_{r+1:t}, \mathcal{W}_{r+1:t}) \to (\mathbb{H}_t, \mathcal{H}_t) \tag{2.46a}$$

$$(h_r, w_{r+1:t}) \mapsto (h_r, \gamma_r(h_r), w_{r+1}, \gamma_{r+1}(h_r, \gamma_r(h_r), w_{r+1}), w_{r+2}, \dots, w_t) \ , \tag{2.46b}$$

that is,

$$\Phi^\gamma_{r:t}(h_r, w_{r+1:t}) = (h_r, u_r, w_{r+1}, u_{r+1}, w_{r+2}, \dots, u_{t-1}, w_t) \ , \tag{2.46c}$$

where $u_s = \gamma_s(h_s)$ and $h_s = (h_r, u_r, w_{r+1}, \dots, u_{s-1}, w_s)$ for $r < s \leqslant t-1$. We complete this definition in the case $0 \leqslant r = t \leqslant T$ with the relation

$$\Phi^\gamma_{r:r} : (\mathbb{H}_r, \mathcal{H}_r) \to (\mathbb{H}_r, \mathcal{H}_r) \tag{2.46d}$$

$$h_r \mapsto h_r \ . \tag{2.46e}$$

The mapping $\Phi^\gamma_{r:t}$ gives the history at time t as a function of the initial history h_r at time r and of the history feedback policies $\{\gamma_s\}_{s\in[\![r,t-1]\!]} \in \Gamma_{r:t-1}$. An immediate consequence of this definition are the two following *flow properties*:

$$\Phi^\gamma_{r:t+1}(h_r, w_{r+1:t+1}) = \Big(\Phi^\gamma_{r:t}(h_r, w_{r+1:t}), \gamma_t\big(\Phi^\gamma_{r:t}(h_r, w_{r+1:t})\big), w_{t+1}\Big) \ , \ \ 0 \leqslant r \leqslant t \leqslant T-1 \ , \tag{2.47a}$$

$$\Phi^\gamma_{r:t}(h_r, w_{r+1:t}) = \Phi^\gamma_{r+1:t}\big((h_r, \gamma_r(h_r), w_{r+1}), w_{r+2:t}\big) \ , \ \ 0 \leqslant r < t \leqslant T \ . \tag{2.47b}$$

2.7.2 Building stochastic kernels from history feedback

Definition 22. *Let r and t be given such that $0 \leqslant r \leqslant t \leqslant T$.*

- *For $0 \leqslant r < t \leqslant T$, let be*

  1. *a $(r:t-1)$-history feedback $\gamma = \{\gamma_s\}_{s=r,\dots,t-1} \in \Gamma_{r:t-1}$,*
  2. *a family $\{\rho_{s-1:s}\}_{r+1\leqslant s\leqslant t}$ of stochastic kernels*

  $$\rho_{s-1:s} : (\mathbb{H}_{s-1}, \mathcal{H}_{s-1}) \times \mathcal{W}_s \to [0,1] \ , \ \ s = r+1, \dots, t \ . \tag{2.48}$$

  We define a stochastic kernel

  $$\rho^\gamma_{r:t} : (\mathbb{H}_r, \mathcal{H}_r) \times \mathcal{H}_t \to [0,1] \tag{2.49a}$$

  by, for any $\varphi : (\mathbb{H}_t, \mathcal{H}_t) \to [0, +\infty]$, measurable nonnegative extended real-valued function,[7]

  $$\int_{\mathbb{H}_t} \varphi(h'_r, h'_{r+1:t}) \rho^\gamma_{r:t}(h_r, \mathrm{d}h'_t) = \int_{\mathbb{W}_{r+1:t}} \varphi\big(\Phi^\gamma_{r:t}(h_r, w_{r+1:t})\big) \prod_{s=r+1}^{t} \rho_{s-1:s}\big(\Phi^\gamma_{r:s-1}(h_r, w_{r+1:s-1}), \mathrm{d}w_s\big) \ . \tag{2.49b}$$

[7] We could also consider any $\varphi : \mathbb{H}_t \to \mathbb{R}$, measurable bounded function, or measurable and uniformly bounded below function. However, for the sake of simplicity, we will deal in the sequel with measurable nonnegative extended real-valued functions.

- *When* $0 \leqslant r = t \leqslant T$, *we define*

$$\rho^{\gamma}_{r:r} : (\mathbb{H}_r, \mathcal{H}_r) \times \mathcal{H}_r \to [0,1] \,, \tag{2.49c}$$

by: $\rho^{\gamma}_{r:r}(h_r, \mathrm{d}h'_r) = \delta_{h_r}(\mathrm{d}h'_r)$.

The stochastic kernels $\rho^{\gamma}_{r:t}$ given by (2.49) are of the form

$$\rho^{\gamma}_{r:t}(h_r, \mathrm{d}h'_t) = \rho^{\gamma}_{r:t}(h_r, \mathrm{d}h'_r \mathrm{d}h'_{r+1:t}) = \delta_{h_r}(\mathrm{d}h'_r) \otimes \varrho^{\gamma}_{r:t}(h_r, \mathrm{d}h'_{r+1:t}) \,, \tag{2.50}$$

where, for each $h_r \in \mathbb{H}_r$, the probability distribution $\varrho^{\gamma}_{r:t}(h_r, \mathrm{d}h'_{r+1:t})$ only charges the histories visited by the flow from $r+1$ to t.

Proposition 23 (Flow property)**.** *The family* $\{\rho^{\gamma}_{s:t}\}_{r \leqslant s \leqslant t}$ *of stochastic kernels. given in Definition 22, has the flow property, that is, for* $s < t$,

$$\rho^{\gamma}_{s:t}(h_s, \mathrm{d}h'_t) = \int_{\mathbb{W}_{s+1}} \rho_{s:s+1}\big(h_s, \mathrm{d}w_{s+1}\big) \rho^{\gamma}_{s+1:t}\Big(\big(h_s, \gamma_s(h_s), w_{s+1}\big), \mathrm{d}h'_t\Big) \,. \tag{2.51}$$

2.7.3 Proof of Proposition 20

Proof. We define, for any $\{\gamma_s\}_{s \in [\![t, T-1]\!]} \in \Gamma_{t:T-1}$,

$$V^{\gamma}_t(h_t) = \int_{\mathbb{H}_T} j(h'_T) \rho^{\gamma}_{t:T}(h_t, \mathrm{d}h'_T) \,, \quad \forall h_t \in \mathbb{H}_t \,. \tag{2.52}$$

From the definition (2.52), we have for any $\{\gamma_s\}_{s \in [\![t, T-1]\!]} \in \Gamma_{t:T-1}$,

$$V^{\gamma}_t(h_t) = \int_{\mathbb{H}_T} j(h'_T) \rho^{\gamma}_{t:T}(h_t, \mathrm{d}h'_T)$$

that only depends on $\{\gamma_s\}_{s \in [\![t, T-1]\!]}$

$$= \int_{\mathbb{H}_T} j(h'_T) \int_{\mathbb{W}_{t+1}} \rho_{t:t+1}\big(h_t, \mathrm{d}w_{t+1}\big) \rho^{\gamma}_{t+1:T}\Big(\big(h_t, \gamma_t(h_t), w_{t+1}\big), \mathrm{d}h'_T\Big)$$

by the flow property (2.51) for stochastic kernels

$$= \int_{\mathbb{W}_{t+1}} \rho_{t:t+1}\big(h_t, \mathrm{d}w_{t+1}\big) \int_{\mathbb{H}_T} j(h'_T) \rho^{\gamma}_{t+1:T}\Big(\big(h_t, \gamma_t(h_t), w_{t+1}\big), \mathrm{d}h'_T\Big)$$

by Fubini Theorem [41, p.137]

$$= \int_{\mathbb{W}_{t+1}} \rho_{t:t+1}\big(h_t, \mathrm{d}w_{t+1}\big) V^{\gamma}_{t+1}\big(h_t, \gamma_t(h_t), w_{t+1}\big)$$

by definition (2.52) of V_{t+1}^{γ}

$$\geqslant \int_{\mathbb{W}_{t+1}} \rho_{t:t+1}\big(h_t, \mathrm{d}w_{t+1}\big) V_{t+1}\big(h_t, \gamma_t(h_t), w_{t+1}\big)$$

by definition (2.8) of the value function V_{t+1}, and as V_{t+1}^{γ} only depends on $\{\gamma_s\}_{s\in[\![t+1,T-1]\!]}$. We deduce that

$$V_t(h_t) \geqslant \inf_{u_t} \int_{\mathbb{W}_{t+1}} \rho_{t:t+1}\big(h_t, \mathrm{d}w_{t+1}\big) V_{t+1}\big(h_t, u_t, w_{t+1}\big) \ .$$

This last inequality is in fact an equality, as seen by using any measurable history feedback policy $\gamma^\star = \{\gamma_s^\star\}_{s\in[\![t,T-1]\!]}$ given by Assumption 3. □

Chapter 2. Bibliography

[33] R. Bellman. *Dynamic Programming*. Princeton University Press, New Jersey, 1957.

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

[35] D. P. Bertsekas. *Dynamic Programming and Optimal Control: Approximate Dynamic Programming*. Athena Scientific, fourth edition, 2012.

[36] D. P. Bertsekas and S. E. Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Athena Scientific, Belmont, Massachusets, 1996.

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

[38] P. Carpentier, J.-P. Chancelier, G. Cohen, and M. De Lara. *Stochastic Multi-Stage Optimization. At the Crossroads between Discrete Time Stochastic Control and Stochastic Programming*. Springer-Verlag, Berlin, 2015.

[39] C. Dellacherie and P. A. Meyer. *Probabilités et potentiel*. Hermann, 1975.

[40] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[41] M. Loève. *Probability Theory I*. Springer Science & Business Media, New York, fourth edition, 1977.

[42] M. V. Pereira and L. M. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1-3):359–375, 1991.

[43] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[44] W. B. Powell. Clearing the jungle of stochastic optimization. *Informs*, pages 109–137, 2014.

[45] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

[46] R. T. Rockafellar and R. J.-B. Wets. *Variational Analysis*. Springer Science & Business Media, Berlin, 1998.

[47] A. Shapiro. Analysis of Stochastic Dual Dynamic Programming Method. *European Journal of Operational Research*, 209:63–72, 2011.

[48] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[49] R. M. Van Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663, 1969.

Part II

Stochastic optimization of storage energy management in microgrids

Chapter 3

Energy and air quality management in a subway station using stochastic dynamic optimization

This is a joint work with Pierre Carpentier, Jean-Philippe Chancelier, Michel De Lara and Julien Waeytens. It was published in *Transactions on Power Systems* (IEEE).

Chapter Abstract

In the Paris subway system, stations represent about one third of the overall energy consumption. Within stations, ventilation is among the top consuming devices; it is operated at maximum airflow all day long, for air quality reasons. In this paper, we present a concept of energy system that displays comparable air quality while consuming much less energy. The system comprises a battery that makes it possible to recover the trains braking energy, arriving under the form of erratic and strong peaks. We propose an energy management system (EMS) that, at short time scale, controls energy flows and ventilation airflow. By using proper optimization algorithms, we manage to match supply with demand, while minimizing energy daily costs. For this purpose, we have designed algorithms that take into account the braking variability. They are based on the so-called Stochastic Dynamic Programming (SDP) mathematical framework. We fairly compare SDP based algorithms with the widespread Model Predictive Control (MPC) ones. First, both SDP and MPC yield energy/money operating savings of the order of one third, compared to the current management without battery (our figure does not include the cost of the battery). Second, depending on the specific design, we observe that SDP outperforms MPC by a few percent, with an easier online numerical implementation.

Contents

3.1 Introduction

3.1.1 Context

Apart from train traction, subway stations themselves represent a significant part (one third) of the energy consumption of a subway system in cities like Paris. Fortunately, there is room to reduce their consumption by harvesting some of their unexploited energy potential. We study here the potential energy recovery of a subway station equipped with a battery to recover regenerative braking energy of subways. The application to braking energy storage displays features that go beyond the specific instance: the strong and abrupt intermittency of energy production bursts requires decisions made at short time scale, hence fast computing online algorithms.

Ventilations are among the most significant energy consuming devices in subway stations. One of the reason is because train braking produces a lot of particles that need to be removed by ventilation. By producing regenerative braking energy, trains can dissipate their kinetic energy with a lower bake pads wear. Hence recovering braking energy improves air quality in stations. It might then be useful to control simultaneously a ventilation and a battery to maximize the benefits provided by this interaction.

We present and compare hereby two classes of methods to solve optimal control problems in the presence of uncertainty. The first one is Model Predictive Control which requires only deterministic optimization tools. The second one is Stochastic Dynamic Programming based on Bellman equation. We apply two different algorithms in each class, differing on how online information is used to compute a decision. For SDP, we compare a state augmentation method, to obtain the best possible performance we can achieve with SDP, and a more classic one based on a Markovian simplification for the offline phase of the algorithm. For MPC, we compare a version that models stochasticity as scenario fans and another one that relies on forecasts. We make a fair comparison of these methods by Monte Carlo simulation and present the results.

3.1.2 Literature

We survey literature on regenerative braking energy, air quality modelling and energy storage management.

Regenerative braking energy

in most recent subway systems, trains already produce regenerative energy when they brake and transmit it to accelerating trains on the same line. However when there is no accelerating train nearby it is not possible to ensure the electrical supply demand equilibrium and regenerative braking is impossible. A comprehensive study of all possible ways to recover that energy is presented in [57]. The authors of [58] conclude that wayside energy storage is relevant to reduce the energy consumption of subway stations. The Southeastern Pennsylvania Transportation Authority (SEPTA) successfully installed wayside batteries to recover braking energy as reported in [56]. SEPTA is about to generalize the project to multiple stations.

Air quality in buildings and subway stations

an ANSES report [50] about air quality of underground subway stations states that the concentration of particulate matter whose size is inferior to $10\mu m$ (PM10) can be unhealthy for the workers and maybe users. This is mainly due to ferrous PM10 that are generated during braking of the trains as stated in [74] and [59]. Subway stations operators in Paris took measures to monitor the concentration of PM10 [59] that are openly available online. Many studies used Computational Fluid Dynamics technics to model the dispersion of pollutants in subway stations to produce predictive models as did the authors of [53]. These methods are computationally very expensive and could hardly be integrated in an optimization problem without using reduced basis methods [69] that are challenging to implement in dynamic environments such as subway stations. The methods presented in [74] and [70] use zonal models to compute an estimation of

the global indoor air quality. These models are much more computationally efficient but require many approximations. The authors of [73] used MPC to control the energy consumption of ventilations and the related climate in a subway station. They estimate that their strategy could save up to 30% of energy while maintaining the same comfort levels but don't manage an electrical storage simultaneously.

Energy storage management

most of the litterature apply MPC or Two Stage Stochastic Programming techniques to short term operation optimization of energy storage with uncertain supply as observed in [65]. Authors of [66] and [67] present MPC strategies to manage energy in battery and building climate. In [61], [62] and [60] the authors present SDP strategies to control batteries in microgrids. In [76] SDP is applied to smart home management with electricle vehicle battery management. Few papers [72], [68] seem to compare the performance of different stochastic optimal control strategies.

3.2 Energy system model

We consider the energy system sketched in Figure 3.1. We present the equations describing its physical evolution in continuous time (denoted by t): energy storage, Kirchhoff laws and air quality. This energy system model will be the basis to simulate different management strategies corresponding to different EMS.

3.2.1 Energy storage model

We use a classical simple model of the dynamics of the energy storage system, with the following variables:

- $s(t)$ (%), the state of charge of the battery at time t;
- $u^b(t)$ (kW), the charge ($u^b(t) \geqslant 0$) or discharge ($u^b(t) \leqslant 0$) power of the battery at time t; indeed, we observe on Figure 3.1 that the battery can draw power on the national grid or provide power to the station.

The dynamics of the state of charge is[1]:

$$\frac{ds}{dt} = \rho_c (u^b(t))^+ + \frac{1}{\rho_d}(u^b(t))^- \ , \tag{3.1}$$

with charge/discharge efficiencies ρ_c and ρ_d. This simple linear dynamical model is relevant as long as we can ensure, by proper management, that the state $s(t)$ of charge is kept between proper bounds $\underline{s} \leqslant s(t) \leqslant \overline{s}$ (like 30% and 90% of the capacity), which also ensures a good ageing of the battery.

[1] We recall that $(x)^+ = \max(x, 0)$ and $(x)^- = \min(0, x)$.

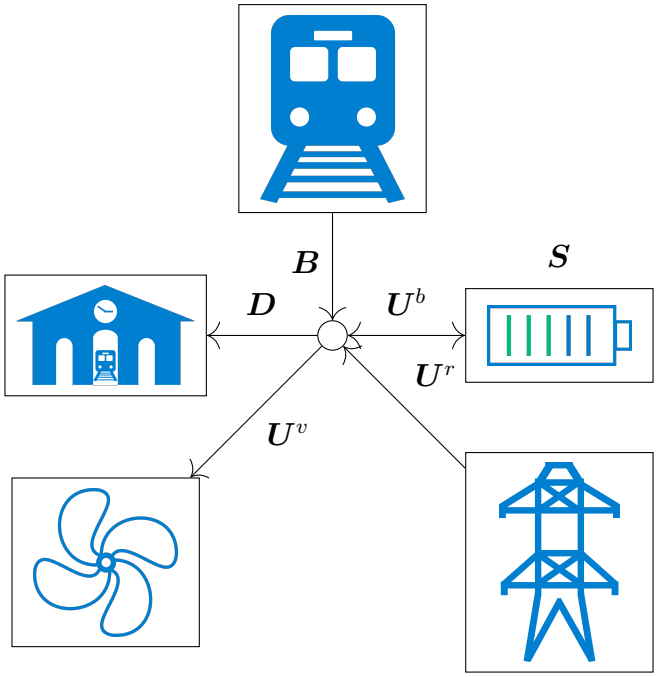

Figure 3.1: Electrical network representation

3.2.2 Kirchhoff laws

On Figure 3.1, we observe that all flows must be balanced at the central node, by Kirchhoff laws. The balance equation writes

$$d(t) + u^v(t) + u^b(t) = b(t) + u^r(t) \ . \tag{3.2}$$

We comment the different terms:

- the station consumes a purely exogenous power $d(t)$ (kW) on the grid at time t;
- the ventilations of the station consume a power $u^v(t)$ (kW); this energy is controllable and we assume that it can be switched between two modes corresponding to two distinct airflows;
- the trains produce a recoverable power $b(t)$ (kW) on the line;
- the difference $u^r(t) = d(t) + u^v(t) + u^b(t) - b(t)$ is either the trains braking power in excess ($u^r(t) \leqslant 0$) that will be wasted (if there is not enough demand, trains brake mechanically instead of electrically), or the power in shortage ($u^r(t) \geqslant 0$) that will be drawn on the national grid to satisfy the demand of the station and, possibly, to charge the battery.

3.2.3 Air quality model

In this study we choose to focus only on pariculate matters, based on the French health agency ANSES report [50], ferrous PM10 is the limiting factor in subway stations. In

[70], the authors use a bi-compartiment model in an office building in order to model the deposition/resuspension phenomenon. Due to the lack of data in subway stations to calibrate the surface dynamic model, we consider a simple air mass conservation model, as presented in [74], to model the dynamics of the particulate matters concentration in the subway station air. As in [74], we assume that the floor is always saturated in dust particles so as to ignore the particles surface dynamics. Moreover, we assume that trains arriving in station produce particles by wearing brake pads and wheels, but also by resuspending particles from the floor. We use the model presented in [74] to model the relation between trains arrivals and particles generation in the air.

The dynamical equation for the PM10 concentration in the station is

$$\frac{dc}{dt} = \alpha n(t)^2 + \Big(\frac{\rho_v}{v}u^v(t) + \beta n(t)\Big)\Big(c^o(t) - c(t)\Big) - \delta c(t)\ , \tag{3.3}$$

with the following notations:

- $c(t)$ $(\mu g/m^3)$, PM10 concentration in the station air;
- $c^o(t)$ $(\mu g/m^3)$, PM10 concentration outside the station;
- v (m^3), volume of the station assimilated as a single zone;
- $n(t)$ (h^{-1}), number of arriving trains per hour;
- α $(\mu gh/m^3)$, apparent generation rate of particles by braking trains;
- δ (h^{-1}), apparent deposition rate of particules;
- β, apparent train contribution rate to natural ventilation;
- ρ_v (m^3/kwh), energy efficiency of the ventilations.

3.2.4 Considerations on numerical simulations

The equations (3.1) and (3.3) form a system of ordinary differential equations. We tested[2] that a forward Euler resolution with $T_0 = 24h$ and $\Delta = 2$ min coincides with a 5th order Tsitouras method using adaptative timestepping with a mean error of 0.06±0.09%. This makes it possible to simulate the energy system, driven by given ventilation and battery control strategies, using a simple discrete time dynamical model. We choose this 2 minutes time step as it coïncides with the frequency of subways, and consequently, of braking energy bursts.

3.3 Optimization problem statement

Once we dispose of the energy system dynamical model, we can envisage to simulate different management strategies and to compare them. They are compared with respect to the daily costs that they induce, while respecting constraints. To make this statement more formal and precise, we now formulate a mathematical optimization problem, under the form of a stochastic optimal control problem.

[2]We used the Julia [52] package DifferentialEquations.jl [71].

3.3.1 Decisions are taken at discrete times

By contrast with the energy system model developed in Sect. 3.2, where time is continuous, we adopt a discrete time frame because decisions are made at discrete steps. Indeed, we consider a subway station grid equipped with a hierarchical control architecture, as in most microgrids [64], that needs time to compute and implement a decision. Decisions are produced every $\Delta = 2$ minutes, over an horizon $T_0 = 24h$; then, they are sent to local controllers that make decisions at a faster pace.

To make the connection with the variable indexed by continuous time in Sect. 3.2, we adopt the following convention: for any variable x, we put $x_t = x(t\Delta)$ for $t = 0, \dots, T = \frac{T_0}{\Delta}$. In other words, x_t denotes the value of the variable x at the beginning of the time interval $[t, t + \Delta[$. This dicretization is compatible with the one discussed in §3.2.4.

3.3.2 Uncertainties are modelled as random variables

We write random variables in capital bold letters, like $\boldsymbol{Z}$, to distinguish them from deterministic variables z.

We model energy demand $\boldsymbol{D}_t$ and trains braking energy production $\boldsymbol{B}_t$, defined when stating the balance equation (3.2), as random variables. We do the same for the number $\boldsymbol{N}_t$ of trains arrivals per hour and for the outside air quality $\boldsymbol{C}_t^o$, both defined when stating the dynamical equation (3.3) for the PM10 concentration in the station.

In the end, we define, for $t = 0, \dots, T$, the vector of uncertainties at time step t

$$\boldsymbol{W}_t = (\boldsymbol{D}_t,\ \boldsymbol{B}_t,\ \boldsymbol{N}_t,\ \boldsymbol{C}_t^o)^\top \ . \tag{3.4}$$

We call $\boldsymbol{W}_t$ the *noise* at time t, that is, the uncertainties materialized at the end of the time interval $[t - \Delta, t)$. The noise $\boldsymbol{W}_t$ takes value in the set $\mathbb{W}_t = \mathbb{R}^4$.

3.3.3 Control variables are modelled as random variables

As time goes on, the noise variables $\boldsymbol{W}_t$ are progressively unfolded and made available to the decision-maker. This is why, as decisions depend on observations in a stochastic optimal control problem, decision variables are random variables: the variables in Sect. 3.2 will now become random variables in capital bold letters.

At time step t, at the beginning of the time interval $[t, t + \Delta[$, the decision-maker takes two decisions: the battery charge/discharge power $\boldsymbol{U}_t^b$ and the ventilation power $\boldsymbol{U}_t^v$. Then, at the end of the time interval $[t, t + \Delta[)$, the decision-maker selects the power $\boldsymbol{U}_{t+1}^r$, drawn from the national grid, to react to the uncertainties $\boldsymbol{D}_{t+1}$ (demand) and $\boldsymbol{B}_{t+1}$ (braking energy) and to ensure the supply demand balance in the grid. This is made possible by a controlled DC/DC converter and supercapacitors that are not modelled in this problem. From the (balance equation) constraint (3.2):

$$\boldsymbol{U}_{t+1}^r = \boldsymbol{D}_{t+1} + \boldsymbol{U}_t^v + \boldsymbol{U}_t^b - \boldsymbol{B}_{t+1} \ . \tag{3.5}$$

We group the two decision/control variables in a vector:

$$\boldsymbol{U}_t = (\boldsymbol{U}_t^b,\ \boldsymbol{U}_t^v) \ . \tag{3.6}$$

We call $\mathbb{U}_t = \mathbb{R}^2$ the set in which the controls take their values.

3.3.4 Non-anticipativity constraints for control variables

To express the fact that the decision-maker (here the EMS) cannot anticipate on the future realizations of the noise, we introduce $\mathcal{F}_t$, the sigma algebra generated by all the past noises up to time t:

$$\mathcal{F}_t = \sigma(\boldsymbol{W}_0, \dots, \boldsymbol{W}_t) \; . \tag{3.7}$$

The increasing sequence $(\mathcal{F}_0, \dots, \mathcal{F}_T)$ is the natural filtration used to model the information flow of the problem. The algebraic non-anticipativity constraint

$$\sigma(\boldsymbol{U}_t) \subset \mathcal{F}_t \tag{3.8}$$

expresses the fact that the decision can only be made knowing no more than the past uncertainties [54, chap. 4].

We say that the controls satisfying (3.8) are $\mathcal{F}_t$-measurable. Throughout the paper, a random variable $\boldsymbol{Z}_t$ indexed by t is, by convention, $\mathcal{F}_t$-measurable, that is, $\sigma(\boldsymbol{Z}_t) \subset \mathcal{F}_t$.

3.3.5 State and dynamics

In the energy system model developed in Sect. 3.2, the equations (3.1) and (3.3) form a system of ordinary differential equations. This is why we introduce two state variables, the state of charge s_t and the PM10 concentration c_t, making thus a two-dimensional *state* variable

$$\boldsymbol{X}_t = (\boldsymbol{S}_t, \; \boldsymbol{C}_t)^\top \; . \tag{3.9}$$

We call $\mathbb{X}_t = \mathbb{R}^2$ the state space where the state takes its values.

By sampling the continuous time differential equations (3.1) and (3.3) at discrete time steps, and by considering that the control variables are piecewise constant between two steps, we can define a discrete time dynamics $f_t : \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1} \to \mathbb{X}_{t+1}$. It is such that

$$\boldsymbol{X}_{t+1} = f_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) = \begin{pmatrix} f_t^s(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) \\ f_t^c(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) \end{pmatrix} \tag{3.10}$$

where

$$f_t^s(x_t, u_t, w_{t+1}) = s_t + \Delta\Big(\rho_c (u_t^b)^+ + \rho_d^{-1} (u_t^b)^-\Big) \; , \tag{3.11a}$$

and

$$f_t^c(x_t, u_t, w_{t+1}) = c_t - \Delta\delta c_t + \Delta\alpha n_{t+1}^2 + \Delta\Big(\frac{\rho_v}{v} u_t^v + \beta n_{t+1}\Big)\Big(c_{t+1}^o - c_t\Big) \; . \tag{3.11b}$$

3.3.6 Bound constraints on the state and control variables

As stated when writing the dynamics of the state of charge in (3.1), the state of charge has to be kept bounded

$$\underline{s} \leqslant \boldsymbol{S}_t \leqslant \overline{s} \; . \tag{3.12}$$

The ventilation airflow can swith between two values, leading to the binary constraint

$$\boldsymbol{U}_t^v \in \{\underline{u^v}, \; \overline{u^v}\} \; , \tag{3.13}$$

and the charge/discharge power is limited, leading to the box constraint

$$\underline{u^b} \leqslant \boldsymbol{U}_t^b \leqslant \overline{u^b} \ . \tag{3.14}$$

The bound constraints (3.12)–(3.13)–(3.14), on the state and control variables, can be summed in the synthetic expression

$$(\boldsymbol{X}_t, \boldsymbol{U}_t) \in B_t \subset \mathbb{X}_t \times \mathbb{U}_t \ . \tag{3.15}$$

3.3.7 The objective is an expected daily cost

We consider the following criterion to be minimized:

$$\mathbb{E}\Big[\sum_{t=0}^{T-1} p_{t+1}\big(\boldsymbol{U}_{t+1}^r\big)^+ + \lambda \boldsymbol{C}_{t+1}\Big] \ . \tag{3.16}$$

We now comment each term.

The term $\mathbb{E}$ stands for the mathematical expectation. By the law of large numbers, mimimizing the mathematical expectation of costs ensures that the system will perform at its best over many days.

Inside the expectation, the sum over time represents the cumulated costs. Those are a mix of two terms.

First, at every time step t, we pay the electricity consumed on the national grid between $t-\Delta$ and t. We call p_t ($€/kW$) the cost of electricity per kW between $t-\Delta$ and t, that we assume to be deterministic. Therefore we pay $p_t \times (\boldsymbol{U}_t^r)^+$ ($€$) at time t.

Second, we give a price of discomfort relative to air quality. Ideally, we would like to keep $T^{-1}\mathbb{E}\Big(\sum_{t=1}^{T} \boldsymbol{C}_t\Big)$, the expected mean of particles concentration over a day, bounded. Indeed, this is the indicator used by the World Health Organization for its PM concentration guidelines [75]. To handle this constraint, we fix a marginal price λ ($€\ m^3/\mu g$)) of discomfort associated with this ideal constraint. We have fixed this parameter by trials and errors, after solving the problem for different values of λ. The cost of discomfort is then $\lambda \times \boldsymbol{C}_t$ ($€$).

Finally, from (3.16) and (3.5), we define the *instantaneous cost* $L_t : \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1} \to \mathbb{R}$ by

$$L_t(x_t, u_t, w_{t+1}) = p_t(d_{t+1} + u_t^v + u_t^b - b_{t+1})^+ + \lambda c_{t+1} \ . \tag{3.17}$$

3.3.8 Stochastic optimal control problem formulation

The EMS problem writes as a general Stochastic Optimal Control (SOC) [54] problem in a risk neutral (expectation) setting

$$\min_{\boldsymbol{X},\boldsymbol{U}} \mathbb{E}\Big[\sum_{t=0}^{T-1} L_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) + K(\boldsymbol{X}_T)\Big] \tag{3.18a}$$

$$\text{s.t } \boldsymbol{X}_{t+1} = f_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) \tag{3.18b}$$

$$(\boldsymbol{X}_t, \boldsymbol{U}_t) \in B_t \tag{3.18c}$$

$$\sigma(\boldsymbol{U}_t) \subset \mathcal{F}_t \tag{3.18d}$$

where K is as final cost function — which is 0 in our case, as we are indifferent of the state of charge at the end of the day.

3.4 Computation of online control strategies

The non anticipativity constraint (3.18d) can be equivalently replaced by its functional counterpart [54, chap. 3, p86]:

$$\exists \pi_t : \mathbb{W}_0 \times \ldots \times \mathbb{W}_t \to \mathbb{U}_t,\ \boldsymbol{U}_t = \pi_t(\boldsymbol{W}_0, \ldots, \boldsymbol{W}_t)\ . \tag{3.19}$$

The mapping π_t is called a *strategy* (more precisely a noise dependent strategy).

In this paper, we restrict the search to solutions among the class of (augmented) *state strategies* of the form

$$\pi_t : \mathbb{X}_t \times \mathbb{W}_t \to \mathbb{U}_t,\ \boldsymbol{U}_t = \pi_t(\boldsymbol{X}_t, \boldsymbol{W}_t)\ . \tag{3.20}$$

This is indeed a restriction, as the state $\boldsymbol{X}_t$ is, by the iterated dynamics (3.10), a function of $(\boldsymbol{W}_0, \ldots, \boldsymbol{W}_t)$.

In practice, we are not interested in knowing $\pi_t(x_t, w_t)$ for all possible values of (x_t, w_t); we just want to be able to compute, on the fly, the value $u_t = \pi_t(x_t, w_t)$ when, at time t, the couple (x_t, w_t) materializes. This is why, in section 3.4.1 and section 3.4.2, we present two methods for the online implementation of strategies. Both methods compute $u_t = \pi_t(x_t, w_t)$ by solving, online, a optimization problem.

3.4.1 Model Predictive Control (MPC)

Model Predictive Control is a general control strategy that has been used extensively in numerous applications. We compare two MPC methods, namely Open Loop Feedback Control (OLFC) and Certainty Equivalent Model Predictive Control (CEC). At time step t', the OLFC algorithm takes as inputs the state x of the system and all the previous uncertainties realizations $w_0, \ldots, w_{t'}$. One way or another, it selects an number S of "scenarios" $(\widetilde{w}^s_{t'+1}, \ldots, \widetilde{w}^s_T)_{s\in\{1,\ldots,S\}}$ with associated probabilities $(p_s)_{s\in\{1,\ldots,S\}}$ and then solves the following deterministic (open loop) optimal control problem:

$$\min_{(u_{t'},\ldots,u_{T-1})} \sum_{s=1}^{S} p_s \sum_{t=t'}^{T-1} L_t(x^s_t, u_t, \widetilde{w}^s_{t+1}) + K(x^s_T) \tag{3.21a}$$

$$\text{s.t } x^s_{t+1} = f_t(x^s_t, u_t, \widetilde{w}^s_{t+1}) \tag{3.21b}$$

$$(x^s_t, u_t) \in B_t, x^s_{t'} = x \tag{3.21c}$$

From the optimal controls $(u_{t'}, \ldots, u_{T-1})$ thus obtained, the OLFC algorithm only keeps the first $(\widetilde{u}_{t'}, \ldots, \widetilde{u}_{t+N_{mpc}})$ (we call N_{mpc} the *reoptimization step* of the OLFC). Then, at time $t' + N_{mpc}$, the OLFC algorithm produces new controls by solving problem (3.21) starting at $t' + N_{mpc}$ with an updated forecast.

As it proves delicate to select decent scenarios for all the remaining time horizon (and as a bad forecast can lead to poor decisions), the online problem horizon $T-1$ in (3.21a) is often cut at $t'+h_{t'}$, with $h_{t'} \geqslant N_{mpc}$. Thus, one obtains problem (3.21) where the objective (3.21a) is replaced by $\sum_{s=1}^{S} p_s \sum_{t=t'}^{t'+h_{t'}} L_t(x^s_t, u^s_t, \widetilde{w}^s_{s+1})$. CEC is a popular version of OLFC where all the future uncertainties are replaced by a single scenario, a forecast. The sketch of the algorithm is the one of OLFC with $S = 1$ and $p_1 = 1$. CEC is often casted in the context of deterministic optimization as it requires only to solve deterministic problems. However it can be used to solve stochastic optimization problems as it provides noise dependent strategies.

3.4.2 Stochastic Dynamic Programming (SDP) based algorithms

A major difference of MPC with the SDP methods is that there is no offline computation phase. We compare two flavours of SDP algorithms that we call Online (SDPO) and Augmented (SDPA).

The offline-online SDPO algorithm encompasses two phases

a backward functional recursion performed offline; a forward online optimization by exhaustive search.

Offline, the SDPO algorithm computes a sequence of functions $\widetilde{V}_t$ by backward induction as follows:

$$\widetilde{V}_T(x) = K(x) \tag{3.22a}$$

$$\widetilde{V}_t(x) = \min_{u \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \Big[L_t(x,u,w_{t+1}) + \widetilde{V}_{t+1}\Big(f_t(x,u,w_{t+1})\Big) \Big] \mu^{of}_{t+1}(dw_{t+1}) \ . \tag{3.22b}$$

Here, each μ^{of}_{t+1} is an (offline) probability distribution on the set $\mathbb{W}_{t+1}$. The recursion is often performed by exhaustive search in discretized versions of the state and control spaces, hence requiring interpolation of the functions V_t. Indeed, $x_{t+1} = f_t(x,u,w)$ is not guaranteed to fall on a gridpoint of the discretized version of $\mathbb{X}_{t+1}$.

Online, at time t, the SDPO algorithm uses the functions V_t and solves (with possibly a refined discretization of the control space $\mathbb{U}_t$)

$$u_t \in \arg\min_{u \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \Big[L_t(x,u,w_{t+1}) + \qquad \widetilde{V}_{t+1}\Big(f_t(x,u,w_{t+1})\Big) \Big] \mu^{on}_{t+1}(w_t, dw_{t+1}) \ .$$

Here, μ^{on}_{t+1} is an (online) conditional probability distribution on the set $\mathbb{W}_{t+1}$, knowing the previous uncertainty w_t. We choose a conditional distribution depending here only on the last uncertainty realization because we use an order 1 autoregressive model in our numerical experiment. As the online conditional probability distribution μ^{on}_{t+1} depends on past uncertainties, this method produces state and noise dependent decisions in real time.

It is well known [51] that the above offline-online SDPO algorithm produces an optimal solution of the SOC problem (3.18) when i) the random variables $\boldsymbol{W}_0, \ldots, \boldsymbol{W}_T$ are stagewise independent, ii) μ^{of}_t is the probability distribution of $\boldsymbol{W}_t$, iii) $\mu^{on}_t = \mu^{of}_t$ is the (unconditional) probability distribution of $\boldsymbol{W}_t$.

As, in our energy system case, the uncertainties are very likely correlated between successive time steps, they cannot be modelled by stagewise independent noises. Consequently, the strategy provided by the offline-online SDPO algorithm is not guaranteed to be optimal.

The offline-online SDPA algorithm follows the SDPO structure, but with

the state x replaced by the couple (x,w); the uncertainty w replaced by a new uncertainty z.

The dynamics $f_t(x,u,w)$ is also replaced by a dynamics $f_t^A\Big((x,w),u,z\Big)$ of the form

$$\begin{aligned} &f_t^A : \big(\mathbb{X}_t \times \mathbb{W}_t\big) \times \mathbb{U}_t \times \mathbb{Z}_{t+1} \to \big(\mathbb{X}_{t+1} \times \mathbb{W}_{t+1}\big) \text{ where} \\ &f_t^A\Big((x,w),u,z\Big) = \Big(f_t(x,u,f^w(w,z)),\ f^w(w,z)\Big)\ . \end{aligned} \tag{3.23}$$

It is straightforward that the above offline-online SDPA algorithm produces an optimal solution of the SOC problem (3.18) when there exists a stochastic process $\boldsymbol{Z}_0,\dots,\boldsymbol{Z}_T$ such that i) the random variables $\boldsymbol{Z}_0,\dots,\boldsymbol{Z}_T$ are stagewise independent, ii) $\boldsymbol{W}_{t+1} = f^w(\boldsymbol{W}_t,\boldsymbol{Z}_{t+1})$.

The limit of this state augmentation strategy is the well known *curse of dimensionality*. The complexity of SDP grows exponentially with the number of state variables. Here, we try to handle a memory lag of one time step; but handling dependency between noises over multiple time steps would be out of reach.

3.5 Numerical results, assessment and discussion

In section 3.4, we outlined three methods to compute online strategies. Now, we detail how to simulate them on the energy system model developed in section 3.2 and how to compare their expected daily costs.

3.5.1 Common data feeding the algorithms

All the data used for the article is available on our website [3].

Reference case

We consider a subway station i) where the ventilation is operated at constant airflow 60 m^3/s ii) which is not equipped with a battery iii) which does not recover regenerative braking. With this ventilation strategy, the mean PM10 concentration over a day is 108 $\mu g/m^3$, while the maximum is 182 $\mu g/m^3$. The consumption of the station over a day is $2.160MWh$ which costs 161 €.

By choosing this reference case, our aim is to measure the daily savings made possible by investing into a battery and by adopting one of the three strategies outlined in section 3.4. This is a partial analysis, as we do not consider the costs of investment.

Braking energy scenarios for algorithms design

As stated in (3.4), the problem presents four sources of uncertainty. However, we assume that, in (3.4), the demand $\boldsymbol{D}_t$, the number $\boldsymbol{N}_t$ of trains per hour, and the outdoor particles concentration $\boldsymbol{C}_t^o$ are deterministic in our numerical experiment. Indeed, most of the uncertainty comes from the trains energy recovery and, moreover, we can have pretty accurate forecasts for the variables that we assume deterministic.

A *scenario* is any possible realization of the noise process $(\boldsymbol{W}_0,\dots,\boldsymbol{W}_T)$ written $(w_0,\dots,w_T)$. For the braking energy, we generated $5,000$ so-called *optimization scenarios* by using a rule, provided in the link in appendix A, calibrated on realistic data.

[3] https://trigaut.github.io/VentilationArticle.html

These $5,000$ optimization scenarios are the common input provided to all the optimization algorithms, so that they can be used to design the features of each algorithm.

3.5.2 Numerical implementation of the MPC algorithm

Scenarios

Knowing a realization w_t of the noise $\boldsymbol{W}_t$, we need to compute a scenarios $(\widetilde{w}^s_{t+1}, \dots, \widetilde{w}^s_T)_{s\in\{1,\dots,S\}}$ of the future uncertainties. The scenarios relies upon the following log-AR(1) model[4]

$$\log \boldsymbol{W}_{t+1} = a \log \boldsymbol{W}_t + \boldsymbol{Z}_{t+1} \ , \quad \forall t = 0, \dots, T \ , \tag{3.24}$$

with independent residual random variables $(\boldsymbol{Z}_t)_{t=1,\dots,T}$. The coefficient a and the distribution of the residuals are identified using the $5,000$ optimization scenarios. Given a number of scenarios S, we quantize the residuals producing discrete probability laws $(\boldsymbol{Z}^d_t)_{t=1,\dots,T}$ with support of size S allowing to build a scenario fan with S scenarios.

Deterministic problem resolution

MPC algorithms require to solve the deterministic problem (3.21). We present the resolution method, based on a MILP formulation, in Appendix 3.6.1.

3.5.3 Numerical implementation of the SDP algorithms

The offline-online SDPO algorithm

It requires as input the probability distributions μ^{of}_t, used to compute the functions V_t offline, and the conditional probability distributions μ^{on}_t, used to compute the controls online.

- μ^{of}_t: we fit discrete probability distributions at each time step by quantizing, using k-means algorithm, the values taken by the $5,000$ optimization scenarios at this very time step t.
- μ^{on}_t: knowing the realization w_{t-1}, we obtain the conditional probability distributions μ^{on}_t by using the formula $w_t = w^a_{t-1} \exp(z_t)$ (see (3.24)). From the $5,000$ optimization scenarios, we obtain $5,000$ values of z_t, hence $5,000$ values of w_t by $w_t = w^a_{t-1} \exp(z_t)$.

The offline-online SDPA algorithm

In addition to what is needed for the above SDPO algorithm, it requires as input the new dynamics $f^w(w, z))$ such that $\boldsymbol{W}_{t+1} = f^w(\boldsymbol{W}_t, \boldsymbol{Z}_{t+1})$. This dynamic is deduced from Equation (3.24).

[4]The log transform ensures that we produce non negative scenarios.

3.5.4 Out of sample assessment of strategies

We have generated $1,000$ so-called *assessment scenarios*, to be used *only* for the assessment phase.

We take good care to distinguish "optimization scenarios" from "assessment scenarios". They are sealed. Optimization scenarios were used to construct items entering the design of the MPC and SDP algorithms. Assessment scenarios will be used to compare the strategies produced by these algorithms. This is what we call out of sample assessment. By this sealing, no algorithm can take advantage of the assessment scenarios to be more fitted to the assessment phase.

The result of the assessment of a given strategy/algorithm is an histogram of all the $1,000$ costs obtained along the assessment scenarios.

3.5.5 Numerical results

The computer used has Core i7, 4.2Ghz $\times 8$ processor and 16 Go ram + 12 Go swap SSD memory.

No battery case

As we neglected all stochasticity except braking energy in this numerical application, all the agorithms perform similarly regarding the ventilation control without a battery and braking energy recovery. In this deterministic setting, ventilation control optimization provides 7 € of economic savings over a day without deterioration of air quality. It represents a 5% reduction of the station electricity bill.

Comparing the algorithms performance

The results[5] are summed up in Table 3.1. We measure the savings with respect to a reference case with no regenerative braking recovered and ventilation at constant maximum speed over the day.

Strategy	SDPA	SDPO	CEC	OLFC
Offline time	$3h47$	$0h06$	$0h00$	$0h00$
Online time	0.30 ms	0.04 ms	5.7 ms	54 ms
Saved money (€)	-74.1 ± 4.79	-73.0 ± 4.59	-71.4 ± 4.35	-72.4 ± 4.75
Saved energy (kWh)	-1050 ± 69.1	-970 ± 60.22	-942 ± 59.2	-960 ± 62.31
Mean PM10 ($\mu g/m^3$)	106 ± 0.11	107 ± 0.11	107 ± 0.08	106 ± 0.10

Table 3.1: Strategies performances comparison

[5]The lower the better, as we minimize costs. Results are $\pm$ the standard deviation.

We observe in Table 3.1 that all algorithms provide close results. As we look in more detail, we see that SDPA outperforms SDPO, CEC and OLFC on average for the economic savings, the mean PM10 concentration and the saved energy. However, regarding the economic savings, the differences in mean performance (of order 2 to 3 €) are lower than standard deviations (of order 4.5 €), which makes it delicate to conclude. The same analysis goes for the energy savings, although the confidence intervals overlap less.

In fact, the four algorithms can be ranked as follows: SDPA outperforms SDPO that outperforms OLFC that outperforms CEC, for the economic and energy savings (and they are comparable for air quality). To sustain this assertion, one has to look at Figure 3.2 that represents the distribution of the relative performance gap between OLFC and SDPA for the economic savings (a comparable analysis holds for the energy savings).

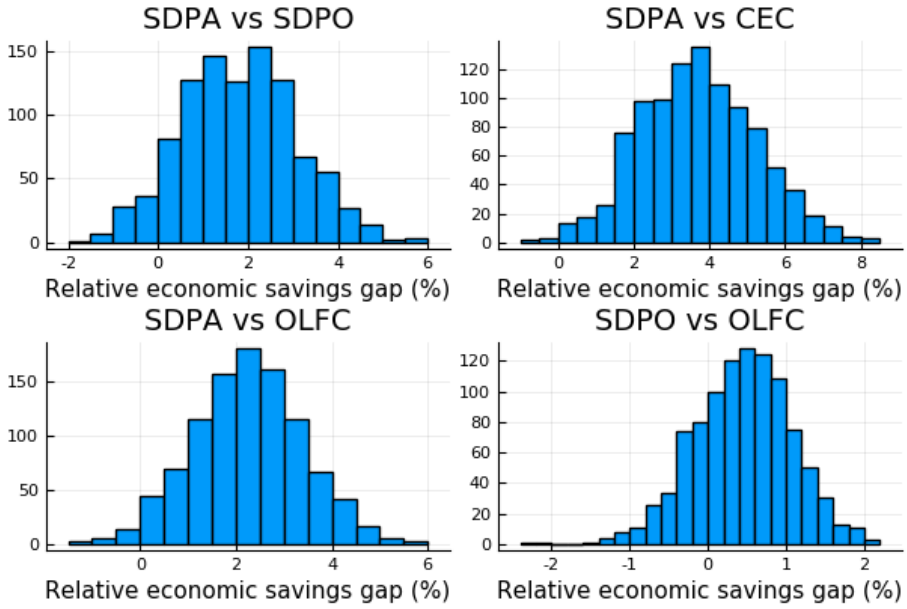

Figure 3.2: Relative savings gap between SDPA and OLFC

On Figure 3.2, the positive portion of the distribution testimony in favor of SDPA for the first three histograms and SDPO for the last one. Our analysis of the assessment scenarios leads to the following observations: i) SDPA outperforms CEC for 995 out of the $1,000$ scenarios, ii) SDPA outperforms OLFC for 979 out of the $1,000$ scenarios, iii) SDPA outperforms SDPO for 928 out of the $1,000$ scenarios, iv) SDPO outperforms OLFC for 762 out of the $1,000$ scenarios, v) SDPO outperforms CEC for all the scenarios, vi) OLFC outperforms CEC for 949 out of the $1,000$ scenarios.

Concerning the computation time, Table 3.1 shows that SDPA requires higher offline computation time than SDPO and MPC algorithms. As the online computation time for the four methods is way under 2 minutes, the three methods are implementable in real time (recall that the decision time step is 2 minutes). However, MPC algorithms differ from SDP algorithms along the following line: MPC requires to solve a MILP online, so that there is no guarantee to reach the optimum, or a feasible solution, within the prescribed 2 minutes; by contrast, both SDP algorithms only perfom an exhaustive

search over all controls in few milliseconds; they display a faster and more stable online computation which we consider valuable for applications with strong energy production bursts.

Energy and air quality results

We display and comment some energy and air quality results based on some of the $1,000$ assessment simulations.

Figure 3.3 displays the state of charge trajectories (in grey) of the battery on the $1,000$ assessment scenarios for each algorithm. We plot in blue the mean state of charge trajectory over all the scenarios and in red the 0.05 and 0.95 quantiles. We observe that the battery is more intensively operated by SDP methods, illustrating SDP's ability to recover more energy than MPC.

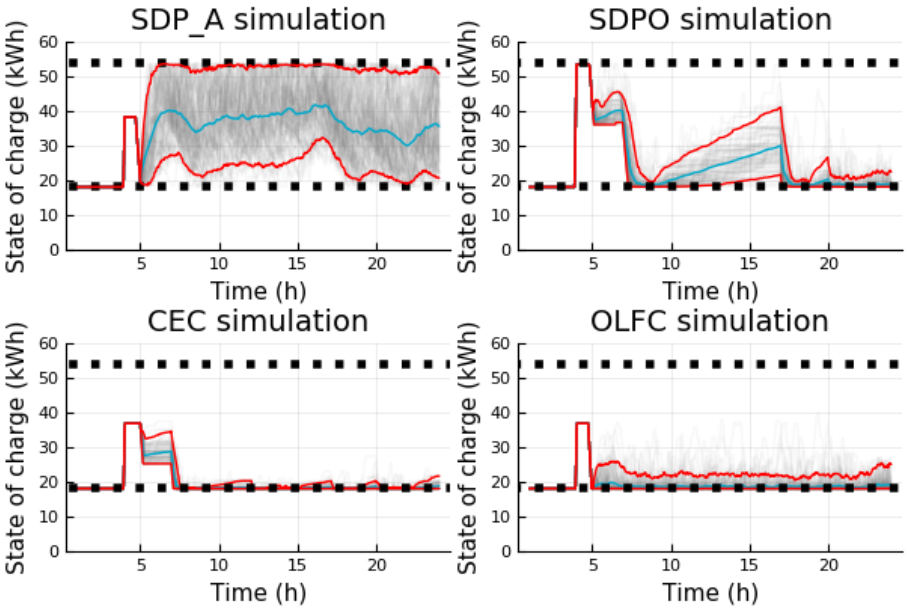

Figure 3.3: Simulations of the state of charge

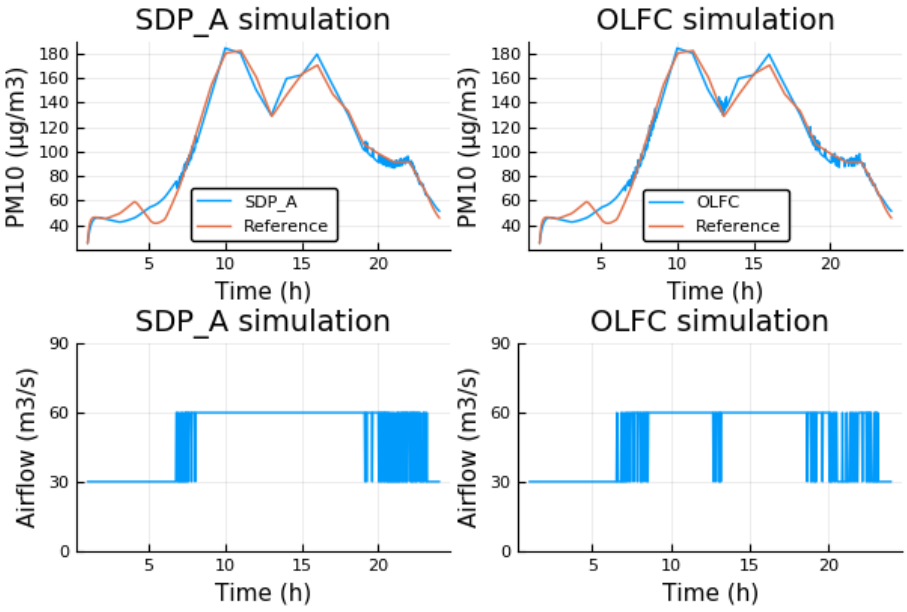

Figure 3.4: Air quality simulations

Figure 3.4 displays the controls of the ventilation (below) and the impact on the PM10 concentration (above) over 1 scenario for both SDPA and OLFC. We recall that, in the reference case, the ventilation is operated at $60m^3/s$ over the whole day. We observe that both algorithms decrease the consumption of the ventilation while maintaining a similar air quality. [6] Outdoor PM10 concentration is higher outside at night. Therefore ventilation in the reference case deteriorates indoor particles concentration between 2 and 5 a.m.

3.6 Conclusions and perspectives

We have presented a subway station energy system, with a battery recovering trains braking and smart control of the ventilations. We have investigated methods to develop and implement an Energy Management System that is able to handle uncertainties related to energy generation. We have discussed the pros and cons of two popular techniques: Stochastic Dynamic Programming (SDP) and Model Predictive Control (MPC). For such a system (with a reasonable number of state variables), we have concluded that SDP is the best choice, even if MPC is a decent alternative. This is not the case in this paper but we recall that MPC could require computationally expensive mathematical programming techniques to solve online deterministic problems.

Our numerical experiments provide encouraging results. It seems that it pays to optimize to improve the energy efficiency and air quality of subway stations. Indeed, as seen on Figure 3.4, the ventilations energy consumption can be decreased without deteriorating the air quality.

[6] Had we modeled the particles generation reduction due to braking energy recovery, we would have obtained a sharper decrease in PM10 concentration.

Our contribution is a first step towards the analysis of new subway station energy systems. It needs to be completed by an economic analysis that includes the costs of batteries and the practical installation of such systems.

3.6.1 Deterministic problem resolution

To solve the MPC deterministic problem, we use Mixed Integer Linear Programming (MILP) by minimizing over states and control variables. As the ventilation airflow u_t^v can switch between two modes, the constraint (3.11b) contains the product of a binary and a continuous variable $u_t^v \times c_t$. That kind of term can be easily linearized.

The constraint (3.11a) contains positive and negative parts of u_t^b, introducing non linearities. To circumvent the problem, we introduce two decision variables, u_t^{b+} and u_t^{b-} together with the constraint $u_t^{b+} \times u_t^{b-} = 0$. It appears that this latter constraint can be removed as it always satisfied at optimality. Indeed, there is no interest to flow through the battery to reach the demand as the battery efficiency coefficients waste power.

To solve this MILP, we use the Julia package JuMP [55] with the commercial solver Gurobi [63].

Chapter 3. Bibliography

[50] Anses. Pollution chimique de l'air des enceintes de transports ferroviaires souterrains et risques sanitaires associés chez les travailleurs. Technical report, Agence nationale de sécurité sanitaire de l'alimentation, de l'environnement et du travail, 2015.

[51] D. P. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific, 1995.

[52] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

[53] F. E. Camelli, G. Byrne, and R. Löhner. Modeling subway air flow using cfd. *Tunnelling and Underground Space Technology*, 43(Supplement C):20 – 31, 2014.

[54] P. Carpentier, J.-P. Chancelier, G. Cohen, and M. De Lara. *Stochastic Multi-Stage Optimization. At the Crossroads between Discrete Time Stochastic Control and Stochastic Programming*. Springer-Verlag, Berlin, 2015.

[55] I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[56] A. J. Gillespie, E. S. Johanson, and D. T. Montvydas. Energy storage in pennsylvania: Septa's novel and innovative integration of emerging smart grid technologies. *IEEE Vehicular Technology Magazine*, 9(2):76–86, 2014.

[57] A. González-Gil, R. Palacin, and P. Batty. Sustainable urban rail systems: Strategies and technologies for optimal management of regenerative braking energy. *Energy conversion and management*, 75:374–388, 2013.

[58] A. González-Gil, R. Palacin, P. Batty, and J. Powell. A systems approach to reduce urban rail energy consumption. *Energy Conversion and Management*, 80:509–524, 2014.

[59] D. Grange and S. Host. Pollution de l'air dans les enceintes souterraines de transport ferroviaire et santé. Technical report, Observatoire régional de santé Île-de-France, 2012.

[60] P. Haessig, T. Kovaltchouk, B. Multon, H. Ben Ahmed, and S. Lascaud. Computing an optimal control policy for an energy storage. In *EuroSciPy 2013, Bruxelles*, 2013.

[61] B. Heymann, J. F. Bonnans, P. Martinon, F. J. Silva, F. Lanas, and G. Jiménez-Estévez. Continuous optimal control approaches to microgrid energy management. *Energy Systems*, pages 1–19, 2015.

[62] B. Heymann, J. F. Bonnans, F. Silva, and G. Jimenez. A stochastic continuous time model for microgrid energy management. In *Control Conference (ECC), 2016 European*, pages 2084–2089. IEEE, 2016.

[63] G. O. Inc. Gurobi Optimizer Reference Manual, 2014. http://www.gurobi.com.

[64] D. E. Olivares, A. Mehrizi-Sani, A. H. Etemadi, C. A. Cañizares, R. Iravani, M. Kazerani, A. H. Hajimiragha, O. Gomis-Bellmunt, M. Saeedifard, R. Palma-Behnke, et al. Trends in microgrid control. *IEEE Transactions on smart grid*, 5(4):1905–1919, 2014.

[65] A. Papavasiliou, Y. Mou, L. Cambier, and D. Scieur. Application of stochastic dual dynamic programming to the real-time dispatch of storage under renewable supply uncertainty. *IEEE Transactions on Sustainable Energy*, 2017.

[66] A. Parisio, E. Rikos, and L. Glielmo. A model predictive control approach to microgrid operation optimization. *IEEE Transactions on Control Systems Technology*, 22(5):1813–1827, Sept 2014.

[67] P. Pflaum, M. Alamir, and M. Y. Lamoudi. Comparison of a primal and a dual decomposition for distributed MPC in smart districts. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pages 55–60. IEEE, 2014.

[68] W. B. Powell and S. Meisel. Tutorial on stochastic optimization in energy part ii: An energy storage illustration. *IEEE Transactions on Power Systems*, 31(2):1468–1475, March 2016.

[69] C. Prud'Homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici. Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods. *Journal of Fluids Engineering*, 124(1):70–80, Nov. 2001.

[70] J. Qian, A. R. Ferro, and K. R. Fowler. Estimating the resuspension rate and residence time of indoor particles. *Journal of the Air & Waste Management Association*, 58(4):502–516, 2008.

[71] C. Rackauckas and Q. Nie. Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1), 2017.

[72] A. N. Riseth, J. N. Dewynne, and C. L. Farmer. A comparison of control strategies applied to a pricing problem in retail. *arXiv preprint arXiv:1710.02044*, 2017.

[73] M. Vaccarini, A. Giretti, L. Tolve, and M. Casals. Model predictive energy control of ventilation for underground stations. *Energy and buildings*, 116:326–340, 2016.

[74] E. Walther, M. Bogdan, and R. Cohen. Modelling of airborne particulate matter concentration in underground stations using a two size-class conservation model. *Science of The Total Environment*, 607:1313–1319, 2017.

[75] WHO. *Air Quality Guidelines: Global Update 2005. Particulate Matter, Ozone, Nitrogen Dioxide and Sulfur Dioxide*. World Health Organization, 2006.

[76] X. Wu, X. Hu, S. Moura, X. Yin, and V. Pickert. Stochastic control of smart home energy management with plug-in electric vehicle battery energy storage and photovoltaic array. *Journal of Power Sources*, 333:203–212, 2016.

Chapter 4

Power management for a DC micro grid integrating renewables and storages

This is a joint work with Alessio Iovine, Gilney Damm, Elena De Santis and Maria Domenica Di Benedetto. It was published in *Control Engineering Practices* (Elsevier).

Chapter Abstract

A power management controller for a DC MicroGrid containing renewable energy sources, storage elements and loads is presented. The controller ensures power balance and grid stability even when some devices are not controllable in terms of their power output, and environmental conditions and load vary in time. Power balance and desired voltage level for the DC MicroGrid are considered as constraints for the controller. Simulations and an experimental setup are implemented to show the effectiveness of the proposed control action.

Contents

4.1 Introduction

MicroGrids are increasingly being considered in distributed power generation given their ability to reduce the physical and electrical distance between sources and loads ([89], [101]) and to improve grid resilience and stability when run in islanded mode. In particular, being able to connect DC power sources such as solar plants to DC loads such as LEDs and electrical vehicles (see [113]) improves substantially power losses. Hence, a large effort has been placed into the development of a efficient controllers dedicated to operate DC MicroGrids ([103], [86], [93], [87], [99], [81]). In this framework, a multilevel scheme is adopted, where economic aspects and energy or power requirements are treated at different time scales (see [103]). The power management problem is formulated as a nonlinear set of equations describing a permanent steady-state regime, neglecting the transient state ([100], [98], [85]). However, dynamic power management considering load variability and weather forecasts should be more efficient ([91], [82], [92], [109]).

The development of high-level controllers for MicroGrids has been the focus of recent research, where all the sources are supposed to be controllable in power, i.e. the controller provides a desired power output reference to each device assuming that the devices can achieve this target power output (see [90], [77], [78], [105], [104])). Unfortunately, not all MicroGrids have direct controllability of each node and the related power output. For example, in case of dealing with different storage devices in an islanded configuration, the different characteristics of each device lead to different targets: the one in charge of controlling uncertainties and unmodeled problems has not its power output imposed a priori, since it depends on the real-time perturbations taking place with respect to the ones calculated by the nominal power flow model. Nevertheless, this power output needs to be monitored and controlled, since it clearly impacts the state of charge of the device. A management controller needs to take it into account in a different way. It must to be noticed that to select the right reference for a device means to properly select the needed amount of power such device must provide or absorb.

In this paper, we consider a DC islanded MicroGrid that includes a battery, a supercapacitor, a photovoltaic system and a load. The battery acts as a reservoir of energy, while the supercapacitor counterbalances transients and unmodeled disturbances, as in [97] and [95]. As a consequence, the supercapacitor power output is not directly controllable. We propose a hierarchical control architecture to optimize the power efficiency of the MicroGrid where a higher-level power management controller provides power references to the elements of the MicroGrids while taking into account the limitations of each device and the available degrees of freedom. Indeed, since the supercapacitor has not a power reference to follow, the battery has to be controlled such to indirectly control the state of charge of the supercapacitor to never be either completely charged nor discharged. This paper extends the preliminary results described in [94], by considering power losses and converter efficiency. This extension results in a non linear optimization problem that

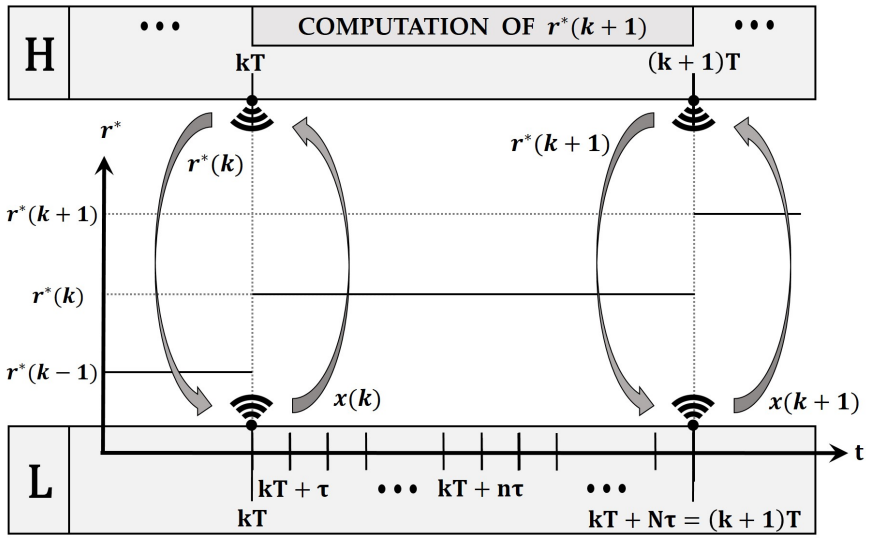

Figure 4.1: The adopted hierarchical control on two levels and the corresponding time scales.

can be solved with Mixed Integer Quadratic Programming (MIQP), or a Mixed Integer Second Order Conic Programming (MISOCP) by using a Model Predictive Controller (MPC) ([83]).

Our hierarchical scheme involves a high-level controller that deals with an optimization problem on a power management model, while the low level controller translates the power reference provided into a voltage or current level. This approach yields a fast power management controller so that it can be used for adapting the power references to time-varying conditions. Our approach has a significant advantage over present solutions that use droop control to adapt to real-time conditions, since they are inefficient in terms of optimization of the available resources and their management. Having a high update rate, the proposed scheme allows for an optimization of both the capacity of the storage devices and their stored energy [103]. Indeed, available optimization techniques operate with long time steps for the high-level controller, while the adaptation to real time conditions are left to non optimized techniques as the droop control. The references for the power management controller are supposed given by an higher Energy Management System (EMS), which is not modeled here.

Further, comparing to [94], we provide simulation results for a realistic power electronics setup using SimPowerSystems (see [108]) to simulate the DC MicroGrid with the proposed controller. The controller is implemented in embedded hardware for low-budget applications, such as home-owned photovoltaics and battery sets.

The paper is organized as follows. Section 4.2 describes the considered hierarchical control levels and in Section 4.3 the mathematical model for the power management is introduced. In Section 4.4 the optimization problem is formulated and the optimal solution is derived. Simulation and experimental results are offered in Section 4.5.

4.2 Hierarchical control structure

In this paper, we propose a hierarchical control architecture:

L) a distributed low level control system is developed for each device composing the DC grid. The control laws operate according to the device mathematical model in order to obtain the desired level of power, which is a given reference.

H) a centralized high level controller, which is the considered Power Management Controller, provides the power references for the local low level controllers. According to a power management model, it uses receding horizon techniques to predict the future states and calculates the optimal references for ensuring power balance.

The Energy Management System (EMS) at the top of the proposed hierarchical structure, which is in charge of providing the references for the lower levels, is not considered in this paper.

The two levels **L** and **H** have different time scales, as shown in Fig. 4.1. The low level controller (**L**) operates in a range varying from 10^{-6} to 10^{-3} seconds, while the high level one (**H**) has a range from 10^{0} to 10^{1} seconds. At each high level sampling time, denoted kT, the controller **H** provides the references for all low level sampling times $\{kT, kT+\tau, ..., kT+n\tau, ..., kT+N\tau\}$. Note that $T = N\tau$.

At time $k-1$ the controller **H** implements a receding horizon optimization problem of a power management model in order to predict what will be the needed power at time k from all branches of the MicroGrid in order to comply with power balance. At time k, **H** sends the optimal reference values to the local controllers, so that they can physically let the devices obtain the requested amount of power. The value at time k is based on a power management model, on the actual values of the system at time $k-1$, which is provided to the higher level controller by the devices operating local control and sent through a communication channel, and on the calculated system evolution over the considered prediction horizon of $\mathcal{N}$ time steps. Obviously, the iteration provides the references for $k+1$, $k+2$, etc.

The controllers do not share the state variables. Since **H** deals with a power management model, its state and control input are composed by energy and power levels. On the contrary, the model of **L** deals with voltages and currents. The outputs of **H** will be the references for **L**; indeed, given a power reference, the low level controller translates it into a voltage or current reference according to the device conditions.

4.2.1 Low-level control system

We consider a DC microgrid composed by a renewable source (a photovoltaic array - PV), two storages acting at different time scales, a capacitor representing the DC grid and a load (an electric vehicle for example), since this kind of structure composed of a combination of renewables, loads and different types of storage is presently attracting much interest (see [79], [96], [97]). The model of the grid is depicted in Fig. 4.2.

Given the specification for the bounds of the voltage of the capacitor C_9 and some desired reference values for the voltages of the capacitors C_1 and C_4, in [97] a stabilizing

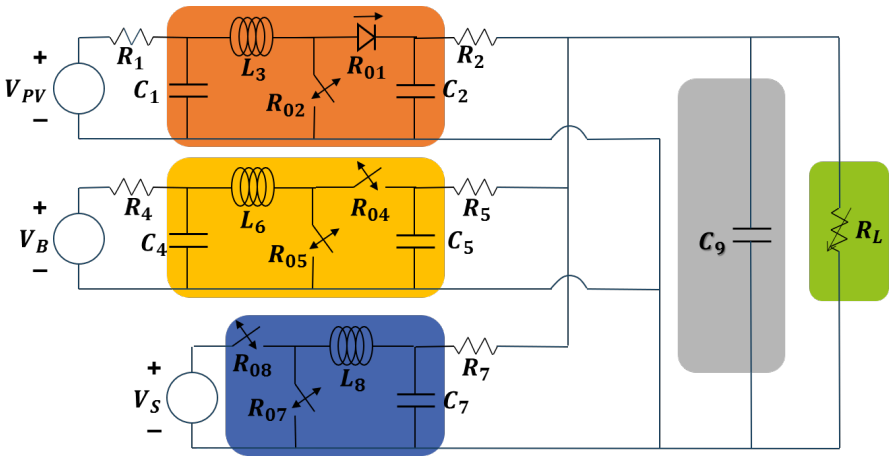

Figure 4.2: The low-level control systems

controller was designed, which ensures the asymptotic convergence to the desired reference values in the nominal case. In this paper, the high level controller is in charge of computing the low level reference values, which are therefore updated at each step k.

4.3 Power Management Model

In this section, the dynamic power management model is introduced, which is composed by equations describing energy variations in the devices and the power flow in the DC grid (see [105], [77], [104], [110]).

Fig. 4.3 depicts the MicroGrid as a set of energy nodes and power edges, from the Power Management perspective: E_B, E_S, E_{DC} are the energies stored in the battery, supercapacitor and DC grid respectively, while $D_{PV} - P_{PV}$, $D_L - P_L$, P_B^+, P_B^-, P_S^+, P_S^-, are the exchanged powers.

The two storages, a battery and a supercapacitor, have different targets. The battery can be viewed as a reservoir that acts as a buffer between the flow requested by the network and the flow supplied by the production sources: its voltage is directly controlled by the DC/DC current converter (highlighted in yellow in Fig. 4.2) applying the reference provided by the high level controller. On the contrary, the supercapacitor maintains grid voltage around a desired value: the reference value for the local controller is then not a consequence of the optimization problem but is dynamically calculated by the low level controller as a consequence of the microgrid values. The power coming in/out the supercapacitor is a consequence of the mismatch between the power produced and the power consumed. Therefore, the optimization problem will consider also the energy quantity in the supercapacitor, since the role of the supercapacitor is essential for the whole grid stability and if it is fully charged or discharged, it cannot continue its operations.

4.3.1 Energy Equations

The considered models for the battery and the supercapacitor are standard and represent them as capacitors (see [104], [106], [77]); they result in a simple impulse response model which is broadly used due to its simplicity and consequently to the low computational

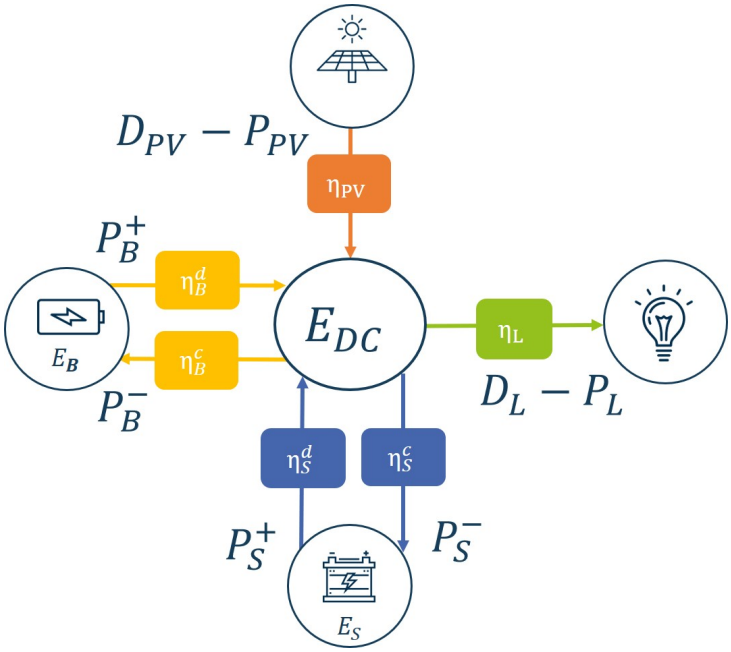

Figure 4.3: The considered framework in a Power Flow scheme represented as a set of nodes.

effort needed for its solution (see [90]). The objective is to make an approximation to deal with the energy variation in three capacitances: battery, supercapacitor and DC grid.

Then a dynamical model can be used to describe the power flow generating the energy variations in the devices. In particular, according to the scheme in Fig. 4.3, the stored energy into the DC grid, the battery and the supercapacitor will vary depending on the power flow needed for supplying the load. As described by the following equations, the energy variations in the battery and the supercapacitor depend only on the power coming in or out of the devices, while for the DC grid they depend on the power balance between the produced and the demanded power. Terms representing the efficiency of the physical devices are also introduced. The dynamical system obtained is:

$$\begin{cases} E_{DC}(k+1) = E_{DC}(k) + T\left[\eta_{PV}(D_{PV}(k) - P_{PV}(k)) - \frac{1}{\eta_L}(D_L(k) - P_L(k))\right] \\ \qquad\qquad +T\left[\eta_B^d P_B^+(k) - \frac{1}{\eta_B^c}P_B^-(k) + \eta_S^d P_S^+(k) - \frac{1}{\eta_S^c}P_S^-(k)\right] \\ E_B(k+1) = E_B(k) + T\left[-P_B^+(k) + P_B^-(k)\right] \\ E_S(k+1) = (1 - T\alpha_S)E_S(k) + T\left[-P_S^+(k) + P_S^-(k)\right] \end{cases} \tag{4.1}$$

where E_B, E_S, E_{DC} are the energies stored in the battery, supercapacitor and DC grid respectively. $D_{PV} - P_{PV}$ is the power produced by the PV array, where D_{PV} is the current available power and P_{PV} is the calculated quantity to be neglected for stability purposes; according to the same reasoning, $D_L - P_L$ is the power demanded by the load, with D_L the current demanded power and P_L the quantity to be disconnected to ensure effective feasibility of the optimization problem (load shedding).

P_B^+, P_B^-, P_S^+, P_S^-, are the powers exchanged by the battery and the supercapacitor, respectively, where the power absorbed by the storages are P_B^- and P_S^-, while the ones provided are P_B^+ and P_S^+. This difference is needed to take into account the efficiency

and the losses due to the power flow and the power constraints, which could result in different values in the charge or discharge case. The parameters η_{PV}, $\frac{1}{\eta_L}$, η_B^d, $\frac{1}{\eta_B^c}$, η_S^d and $\frac{1}{\eta_S^c}$ describe the loss in efficiency and the losses due to $D_{PV} - P_{PV}$, $D_L - P_L$, P_B^+, P_B^-, P_S^+, P_S^-, respectively. The α_S parameter allows one to consider the self-discharge ratio of the supercapacitor taking place in the considered T time interval; the really small self-discharge ratio of the battery is neglected because of the relative short time span considered in these two control levels.

The model in (4.1) is used to calculate the optimal amount of power to be demanded from the battery to correctly feed a load, considering the powers as bounded control inputs or disturbances. In particular, the load demanded power and the one coming from the renewables are seen as disturbances, and extra degrees of freedom are introduced to allow modifications on them in case of unfeasible problems. Our target is to examine a situation where it is possible to deal with the capability to operate on both such powers, taking into account future prediction of their values and physical limitations of the different storage devices.

4.3.2 Assumptions

Proper sizing of each component in a DC microgrid is an important feasibility requirement. In order to always satisfy the power requested by the load, the sizing of the PV array, battery and supercapacitor fits some conditions related to the produced power by the photovoltaic array D_{PV}, the P_B^+, P_B^-, P_S^+, P_S^-, power coming from the storages, and the D_L power absorbed by the load:

i)

Assumption 4. *the sizing of the photovoltaic array is performed according to total energy needed in a whole day;*

$$\int_0^D \eta_{PV} D_{PV}(t)\, dt \geqslant \int_0^D \frac{1}{\eta_L} D_L(t)\, dt \tag{4.2}$$

where D is equal to daytime (24 hours) and the quantities represent the worst case scenario that is considered in this framework, based on previous collected data;

ii)

Assumption 5. *the sizing of the battery and the supercapacitor are performed according to the energy balance in the T time step, needed for selecting a new reference;*

$$\left\| \int_{kT}^{(k+1)T} \left[\eta_{PV} D_{PV}(t) + \eta_B^d P_B^+(t) - \frac{1}{\eta_B^c} P_B^-(t) - \frac{1}{\eta_L} D_L(t) \right] dt \right\| \leqslant \tag{4.3}$$
$$\leqslant \frac{1}{2} \int_{kT}^{(k+1)T} [\eta_S^d P_S^+(t) - \frac{1}{\eta_S^c} P_S^-(t)]\, dt \quad \forall\, k$$

The last condition can be seen as the ability of the supercapacitor to fulfill the request to provide enough amount of power in the considered time interval; for sizing the

supercapacitor the worst scenario due to current load variations is considered, i.e. the case where the supercapacitor needs to provide/absorb the maximum available current for all the time steps.

The exact sizing of the components is considered out of the scope of this work.

4.3.3 Constraints

Here state and input constraints are defined, which represent either physical constraints (hard constraints) or targets (soft constraints):

- to ensure a certain level of power quality and to avoid problems related to the connection with the physical devices, the energy stored in the DC grid must be kept between an interval;

$$E_{DC}^{m} \leqslant E_{DC}(k) \leqslant E_{DC}^{M}, \; \forall \, k \tag{4.4}$$

  where $E_{DC}^{m}, E_{DC}^{M} \geqslant 0$;

- the energy in the battery and the supercapacitor must remain in a range of values, in order not to damage the devices;

$$E_{B}^{m} \leqslant E_{B}(k) \leqslant E_{B}^{M}, \; \forall \, k \tag{4.5}$$

$$E_{S}^{m} \leqslant E_{S}(k) \leqslant E_{S}^{M}, \; \forall \, k \tag{4.6}$$

  where $E_{B}^{m} \geqslant 0$, $E_{B}^{M} \geqslant 0$, $E_{S}^{m} \geqslant 0$, $E_{S}^{M} \geqslant 0$;

The match between power in and power out which represents power balance is already introduced by the equation describing the energy in the DC grid.

Moreover, each control law has different constraints:

- the unavailability to provide more power than the available one from the PV and the storages is introduced, and the choice not to arbitrarily increase the load if needed;

$$P_{PV}(k) \geqslant 0, \; \forall \, k \tag{4.7}$$

$$P_{L}(k) \geqslant 0, \; \forall \, k \tag{4.8}$$

- also, the power coming from the PV array and the one consumed by the load are bounded and cannot be negative;

$$D_{PV}(k) - P_{PV}(k) \geqslant 0, \; \forall \, k \tag{4.9}$$

$$D_{L}(k) - P_{L}(k) \geqslant 0, \; \forall \, k \tag{4.10}$$

- the power absorbed/provided by the battery and the supercapacitor are bounded;

$$0 \leqslant P_{B}^{+}(k) \leqslant \overline{P}_{B}^{+}, \; \forall \, k \tag{4.11}$$

$$0 \leqslant P_{B}^{-}(k) \leqslant \overline{P}_{B}^{-}, \; \forall \, k \tag{4.12}$$

$$0 \leqslant P_{S}^{+}(k) \leqslant \overline{P}_{S}^{+}, \; \forall \, k \tag{4.13}$$

$$0 \leqslant P_{S}^{-}(k) \leqslant \overline{P}_{S}^{-}, \; \forall \, k \tag{4.14}$$

  where $\overline{P}_{B}^{+} \geqslant 0$, $\overline{P}_{B}^{-} \geqslant 0$, $\overline{P}_{S}^{+} \geqslant 0$, $\overline{P}_{S}^{-} \geqslant 0$;

- saving the battery life time is a priority and to this purpose limitations on its power variation are imposed;

$$\|P_B^+(k+1) - P_B^+(k)\| \leqslant \Delta\overline{P}_B^+, \ \forall\, k \tag{4.15}$$

$$\|P_B^-(k+1) - P_B^-(k)\| \leqslant \Delta\overline{P}_B^-, \ \forall\, k \tag{4.16}$$

where $\Delta\overline{P}_B^+ \geqslant 0$, $\Delta\overline{P}_B^- \geqslant 0$;

- the power entering and leaving the battery and the supercapacitor cannot take place at the same time. Conditions similar to the ones in [105] are then implemented as

$$P_B^+(k) \times P_B^-(k) = 0, \ \forall\, k \tag{4.17}$$

$$P_S^+(k) \times P_S^-(k) = 0, \ \forall\, k \tag{4.18}$$

So, to avoid the simultaneous charging and discharging of the battery and the supercapacitor, the following binary variables S_B and S_S are introduced, such to modify the limitations on the power as

$$S_B(k) \in \{0,1\}, \ \forall\, k \tag{4.19}$$

$$S_S(k) \in \{0,1\}, \ \forall\, k \tag{4.20}$$

$$P_B^+(k) \leqslant S_B(k) \cdot \overline{P}_B^+, \ \forall\, k \tag{4.21}$$

$$P_B^-(k) \leqslant (1 - S_B(k)) \cdot \overline{P}_B^-, \ \forall\, k \tag{4.22}$$

$$P_S^+(k) \leqslant S_S(k) \cdot \overline{P}_S^+, \ \forall\, k \tag{4.23}$$

$$P_S^-(k) \leqslant (1 - S_S(k)) \cdot \overline{P}_S^-, \ \forall\, k \tag{4.24}$$

Since the supercapacitor has to ensure voltage stability with respect to the variations taking place on the grid, its power variations over time are not considered to be bounded. Indeed, even if in reality they are bounded, these bounds may be established in a worst case scenario, and according to the considered assumptions they do not impact our problem. This is the reason why there are no conditions for P_S^+ and P_S^- as (4.15) and (4.16) for P_B^+ and P_B^-.

Considering (4.1), it is then possible to rewrite the whole discrete time dynamical system as

$$x(k+1) = Ax(k) + Bu(k) + Dd(k) \tag{4.25}$$

where the state is

$$\begin{aligned} x &= [E_{DC} \ \ E_B \ \ E_S]' = \\ &= [x_1 \ \ x_2 \ \ x_3]' \end{aligned} \tag{4.26}$$

and the input and disturbance vectors are

$$\begin{aligned} u =& [P_{PV} \ \ P_L \ \ P_B^+ \ \ P_B^- \ \ P_S^+ \ \ P_S^- \ \ ]' = \\ &= [u_1 \ \ u_2 \ \ u_3 \ \ u_4 \ \ u_5 \ \ u_6]' \end{aligned} \tag{4.27}$$

$$d = [D_{PV} \;\; D_L]' = \\ = \; [d_1 \;\; d_2]' \tag{4.28}$$

The discrete time matrices A, B, D are

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 - \alpha_S T \end{bmatrix} \tag{4.29}$$

$$B = T \begin{bmatrix} -\eta_{PV} & \frac{1}{\eta_L} & \eta_B^d & -\frac{1}{\eta_B^c} & \eta_S^d & -\frac{1}{\eta_S^c} \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \quad D = T \begin{bmatrix} \eta_{PV} & -\frac{1}{\eta_L} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.30}$$

4.4 Power Management Controller

Once the mathematical model of the dynamical system has been defined, it is possible to define an optimal control problem to formalize the way the Power Management Controller **H** computes online targets for **L**.

Since the Microgrid control architecture is supposed to be able to control the system anytime without knowing in advance the future disturbances, the objective function should be in infinite horizon. Moreover, since the solar power production and the power demand have a random component, the optimization problem should be supposed to minimize an infinite sum of random instantaneous costs, denoted f, with respect to a chosen risk measure. In this paper, the selected risk measure is the expectation: this choice is explained in Section 4.5.2. Then, given the initial state, the problem would result in a stochastic optimal control one, as

$$\min_{u(\cdot)} \mathbb{E} \sum_{k=0}^{\infty} f(x(k), u(k)) \tag{4.31a}$$

$$\text{s.t } (4.4) - (4.25) \tag{4.31b}$$

In the studied case, it is reasonable to assume that, over a finite and sufficiently short time horizon, the high-level system **H** is able to compute good forecasts of the demand and solar power. For this reason, a deterministic Model Predictive Control (MPC) (see [84]) strategy seems adequate to compute control policies for problem (4.31). We could consider other methods such as Stochastic Dynamic Programming, scenario tree based Stochastic Programming or Stochastic MPC (see [107]). However, they all lack of simplicity. Indeed, the first one would suffer the curse of dimensionality and would require Approximate Dynamic Programming techniques (see [111]) that are computationally expensive in the presence of binary variables. Nevertheless, the second and third methods require to solve a large scale (many decision variables) deterministic problem at each time step, which could be too computationally expensive as well in the considered time window of 1 second.

4.4.1 Target

The target of the Power Management Controller **H** is to compute setpoints for the low level controller while receiving targets from an another upper level EMS.

In reality, a desired voltage level for the DC grid is selected; it is then translated into a desired energy level, $E^r_{DC} = x^r_1$. The same reasoning applies to the energy in the battery and supercapacitor, i.e. there exist desired levels $E^r_B = x^r_2$ and $E^r_S = x^r_3$. These targets are supposed to be computed by the EMS, but, since it is out of the scope of this paper, the following considerations are done:

1) the desired state for the battery is to be fully charged, i.e. $x^r_2 = x^M_2$;

2) the reference for the supercapacitor has to be selected to ensure the maximum capability to operate on the system; it means that it must be able to absorb/provide the maximum amount of power, and the adopted best trade-off is $x^r_3 = \frac{1}{2}\left(x^m_3 + x^M_3\right)$.

The target vector for the state is therefore

$$x^r = [E^r_{DC} \;\; E^r_B \;\; E^r_S] \tag{4.32}$$

and the error state is defined as $\widetilde{x} = x - x^r$.

4.4.2 Deterministic problem: Mixed Integer Quadratic Program

Given the initial state $x_0(k)$, which depends on the state of the low level system at time kT, the online deterministic problem addressed by the MPC strategy at each step k is the following

$$\min_{u(\cdot)} \quad \frac{1}{2}\left[\widetilde{x}(k+\mathcal{N})^T P\widetilde{x}(k+\mathcal{N}) + \sum_{i=k}^{k+\mathcal{N}-1} \widetilde{x}(i)^T Q\widetilde{x}(i) + u(i)^T R u(i)\right] \tag{4.33a}$$

$$\text{s.t} \quad \widetilde{x}(i) = x(i) - x^r(i),\ \forall\, i \tag{4.33b}$$

$$x(i+1) = Ax(i) + Bu(i) + Dd(i),\ \forall\, i \tag{4.33c}$$

$$x(k) = x_0(k), \tag{4.33d}$$

$$x_1^m \leqslant x_1(i) \leqslant x_1^M,\ \forall\, i \tag{4.33e}$$

$$x_2^m \leqslant x_2(i) \leqslant x_2^M,\ \forall\, i \tag{4.33f}$$

$$x_3^m \leqslant x_3(i) \leqslant x_3^M,\ \forall\, i \tag{4.33g}$$

$$S_B(i) \in \{0,1\},\ \forall\, i \tag{4.33h}$$

$$S_S(i) \in \{0,1\},\ \forall\, i \tag{4.33i}$$

$$0 \leqslant u_1(i) \leqslant d_1(i),\ \forall\, i \tag{4.33j}$$

$$0 \leqslant u_2(i) \leqslant d_2(i),\ \forall\, i \tag{4.33k}$$

$$0 \leqslant u_3(i) \leqslant S_B(i) \cdot \overline{P}_B^+,\ \forall\, i \tag{4.33l}$$

$$0 \leqslant u_4(i) \leqslant (1 - S_B(i)) \cdot \overline{P}_B^-,\ \forall\, i \tag{4.33m}$$

$$0 \leqslant u_5(i) \leqslant S_S(i) \cdot \overline{P}_S^+,\ \forall\, i \tag{4.33n}$$

$$0 \leqslant u_6(i) \leqslant (1 - S_S(i)) \cdot \overline{P}_S^-,\ \forall\, i \tag{4.33o}$$

$$\|u_3(i+1) - u_3(i)\| \leqslant \Delta\overline{P}_B^+,\ \forall\, i \tag{4.33p}$$

$$\|u_4(i+1) - u_4(i)\| \leqslant \Delta\overline{P}_B^+,\ \forall\, i \tag{4.33q}$$

where $P, Q, R \geqslant 0$ are weight matrices, and the matrices A, B, D are introduced in (4.29) and (4.30). $\mathcal{N}$ is the considered prediction horizon, and it is chosen in a reasonable way as a trade-off between a better performance using more data and the required computational time. Penalizations on the inputs are explained with the willing not to waste solar power or curtail load demand in case of unnecessary conditions.

The problem above is an input-state constrained optimization problem. Since we are considering a microgrid in islanded mode, Assumptions 1 and 2 are necessary, but in general not sufficient for the existence of a solution. However, it is clear that an appropriate sizing of the components always exists such that the constraints can be fulfilled over an infinite horizon of time. In fact, the PV generation induces a daily periodicity in the system. Hence, if we consider an oversizing for the battery and the supercapacitor, it is always possible to control the system in such a way that, for example, at each sunrise there is sufficient energy stored to cover the mismatch between the power coming from the PV and the power required by the load (supposed constant, but affected by a bounded additional disturbance). The question of optimizing the design of the grid, which minimizes a suitable cost functional, is of great importance and is the object of future work. Here we will assume feasibility of the optimization problem.

The resulting problem is a Mixed Integer Quadratic Program (MIQP). This kind of problem is becoming the classical one to be addressed when dealing with energy management controller, as can be seen by the numerous papers implementing it, for example [106], [104], [90], [77].

To the purpose of implementing the optimization above in a low cost hardware, it is useful to remark that the MIQP problem can be reformulated as a Mixed Integer Second Order Cone Program (MISOCP), as presented in [102]. This is motivated by the use of a conic solver for embeddable applications for the Hardware In the Loop simulation presented in Section 4.5.3.

The reference values for the low level controller at step k are computed based on the value $u^o(k)$, where $u^o(.)$ is the optimal sequence of inputs for the above problem. We are considering negligible with respect to T the time required for the computation of the optimal solution.

4.5 Simulations

In this section, simulations for the proposed model and the applied optimal control are introduced. The considered sampling time T is one second, while the simulation time is 180 seconds. A prediction horizon of 10 time steps is utilized.

4.5.1 Low level controller implementation

The output obtained by the optimal control problem at each time k will be sent from **H** to **L** to let it operate grid stability. In particular, the optimal values of P_{PV}, P_B^+ or P_B^- and P_L must be respected: the low level controller will obtain its reference values from them. As explained before, the target of the supercapacitor is to maintain a fixed grid voltage level: then it does not really need a reference power value, since it automatically operates the needed action as a response to what the other devices do, and any uncertainty or disturbance on the system. Then the value of P_S^+ and P_S^- are calculated only to let the other devices take them into account. The reference for the voltage level of the DC grid is supposed to be fixed a priori, in general by technological reasons, and as a consequence the reference coming from E_{DC} is not needed.

The considered DC MicroGrid described in Section 4.2.1 and depicted in Fig. 4.3 has been simulated in SimPowerSystems, a dedicated Matlab toolbox for modeling and simulating electrical power systems. The nonlinear control algorithms introduced in [97] are executed, with a time step of 10^{-5} seconds, according to the description of the **L** controller in Section 4.2. Based on such realistic implementation, the DC microgrid is structured as composed by a 300 kWh battery, a 8 kWh supercapacitor and a PV power plant of 50 solar panels with installed power of 1 MW. The battery has the capability to move from 30% to 90% of State Of Charge (SOC) in 90 minutes and from 90% to 30% in 30 minutes. The limit boundaries of 30% and 90% of SOC are considered in order not to reduce its lifetime.

The PV power productions are based on solar scenarios which has been generated using the python library PVLIB (see [112]), based on interpolated real measured hourly solar radiation scenarios [1]. One of the PVLIB referenced solar panel models has then been used to simulate realistic solar power scenarios with a time step of one second. The load resistance is considered constant, but the power load suffers of variations due to DC

[1]Radiation scenarios of Zambia, obtained on the open data website: https://energydata.info/dataset/zambia-solar-radiation-measurement-data-2015-2017

grid voltage variations, since it is directly connected to it. Then, also a simulation with a more complex power profile is introduced.

4.5.2 Forecast strategy

The MPC control method requires a proper forecast strategy for the near future disturbances. At any given time step k, knowing the last solar production and load, our forecast strategy is straightforward. Both future solar and load power disturbances over the MPC rolling horizon are replaced by their last observed value. Hence we consider solar and load as constant over the next few time steps. More clever forecasting strategies based on statistical models could be used, but this naive approach provides satisfactory results. With this forecast strategy, that depends only on the last observed disturbances, MPC produces current state and last disturbances feedbacks, which are simpler than the general full disturbances history feedbacks mentioned in Section 4.4.

4.5.3 Power management controller implementation

We compared two different technical implementations of the high level controller. Both involve deploying a REST API developed in Julia language (see [80]) reachable through HTTP GET requests. For the first one, the server runs on an office computer with 16Go of RAM and a Intel i7-7700k @4.2 GHz CPU. The deterministic problem (4.33) is modeled using the Julia optimization modeler JuMP (see [88]) and solved using the commercial IBM solver CPLEX (see Section 4.5.4). In the second implementation, the MIQP (4.33) is converted into its Mixed Integer Second Order Cone Program (MISOCP) equivalent formulation and modeled using JuMP. This allows to use the open source embeddable conic solver ECOS which possesses a basic implementation of a branch and bound algorithm (ECOS BB) to solve MISOCP problems. Using an ARM compatible version of Julia it is possible to deploy the REST API into a Raspberry Pi 2B (RPi) with an ARM Cortex A7 @900MHz CPU and 1GO SDRAM. We added a time limit feature to ECOS branch and bound algorithm so as to return the best found admissible suboptimal solution when a time limit is exceeded[2]. This functionality ensures that an admissible control is computed and transmitted by the RPi to the simulation computer, through ethernet, in less than one second (see Section 4.5.5).

Fig. 4.4 shows the utilized configuration for the tests: the model describing the single components of the grid is implemented in SimPowerSystems, while the one dealing with the power flow is implemented in Julia (both in the pc or in the Raspberry Pi), and they exchange information by the aforementioned ethernet connection.

To the purpose of imitating real conditions in the best way, errors in the efficiency coefficients values are introduced between the two models and control levels.

4.5.4 Simulations scenarios

Four cases will here be introduced and discussed, in order to consider all the possible interesting scenarios. A prediction horizon of 10 time steps is utilized for the case studies 1, 2 and 3, while a horizon of 5 time steps is used for case study 4, both for state variables

[2]This feature is available in the ECOS github fork: https://github.com/trigaut/ecos

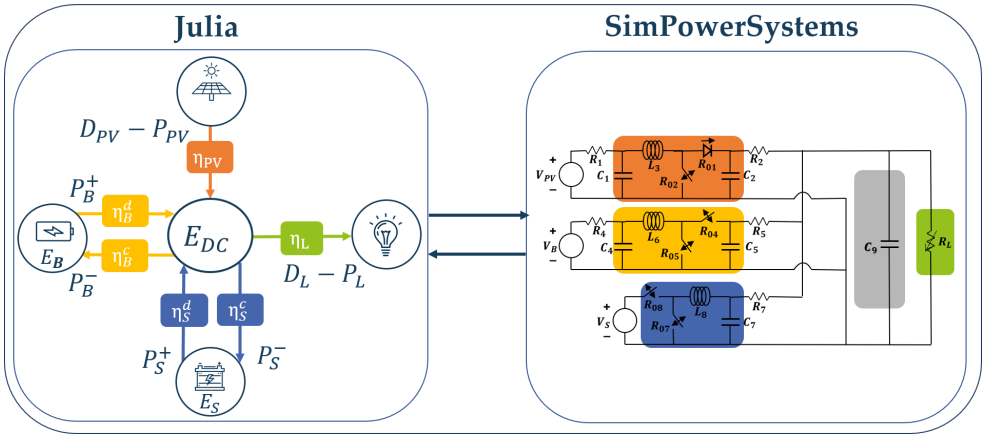

Figure 4.4: Test configuration.

Reference	Value	Reference	Value
E^*_{DC}	$1.39e^{-5}$ kWh		
E^*_B	270 kWh	SOC^*_B	90 %
E^*_S	5 kWh	SOC^*_S	62.5 %

Table 4.1: References.

and control inputs. Small slack variables have been used in order to relax constraints. Case study 1 describes the evolution of a situation where initially the supercapacitor has a level of charge which is higher than its reference, and the introduced controller properly perform in order to discharge it. Case study 2 is similar to case study 1, but will compare the evolution of the system in case there is no possibility to reduce the power coming from the PV panels. Case study 3, as opposed to the other two, will introduce a situation where the supercapacitor has a level of charge lower than the desired one. Finally, case study 4 considers a situation where the supercapacitor has a higher than desired initial state of charge, with a time varying load. For every case, the possibility to reduce the power demanded by the load if needed is considered. As mentioned in Section 5.1, the used battery and supercapacitor have 300 kWh and 8 kWh of energy, respectively, and their references and constraints are introduced in Tables 4.1, 4.2 and 4.3.

Constraint	Value
E^m_{DC}	90 % E^r_{DC}
E^M_{DC}	110 % E^r_{DC}
E^m_B	90 kWh
E^M_B	270 kWh
E^m_S	1.41 kWh
E^M_S	8 kWh

Table 4.2: Constraints for the states variables.

Constraint	Value	Constraint	Value
$\overline{P}_B^+$	240 kW	$\overline{P}_B^-$	180 kW
$\overline{P}_S^+$	1000 kW	$\overline{P}_S^-$	1000 kW
$\Delta\overline{P}_B^+$	20 kW	$\Delta\overline{P}_B^-$	20 kW

Table 4.3: Constraints for the control inputs.

Case study 1

Here a situation where initially the supercapacitor has a level of charge which is higher than its reference is considered, as can be seen in Fig. 4.9. The power management controller successfully let it reach the desired value acting on the other components of the power flow, as it is shown in Fig. 4.5, 4.7 and 4.8. Fig. 4.5 describes the behavior of P_B, which is the difference of P_B^+ and P_B^-. In particular, it is worth noticing that the control algorithm combines two actions to let the supercapacitor discharge in the fastest possible rate: indeed, with respect to the constraints of the battery, the power management controller selects both the possibility to charge more the battery and not to let the renewables produce their maximum available power. The figures clearly describe how the power management controller allows PV power reduction in the meanwhile of a good charge rate for the battery in order to discharge the supercapacitor and to let it arrive at the desired reference of SOC (and voltage). Then, considering also the smaller availability of power from renewables, the controller **H** describes a situation of discharge for the battery in order to provide the power for the load. It has to be noticed that the battery discharge and charge limits are reached; due to the small errors in the modeling of converters between the two control levels, there can be a steady state error between the desired battery power flow and the obtained one. The effectiveness of the proposed controller is verified in Fig. 4.6 indeed, even with an error due to the wrong modeling of the converter efficiency, the real power output of the supercapacitor follows the predicted optimal value even if this one is not directly implemented as reference in the low level controller. Then, the desired level of energy in the supercapacitor is reached as first target of the power management controller.

Case study 2

Here the same initial situation as in Case study 1 is considered, but to better highlight controller robustness, the possibility to reduce power produced by the PV is taken into consideration in the **H** controller calculation but it is not implemented in the **L** controller. Then, the remarkable difference introduced here is depicted in Fig. 4.13, where the desired value of $D_{PV} - P_{PV}$ is not tracked by the real value of $D_{PV} - P_{PV}^r$. As a consequence, the battery will need to take more power for more time with respect to the case 1, as depicted in Fig. 4.10 compared to Fig. 4.5. Obviously, Fig. 4.11 describing the power output of the supercapacitor will show an error between its expected value and the calculated one. Nevertheless, the loss of such degree of freedom does not impact the capability of the system to bring the level of energy of the supercapacitor to the desired one, as can be seen in Fig. 4.14.

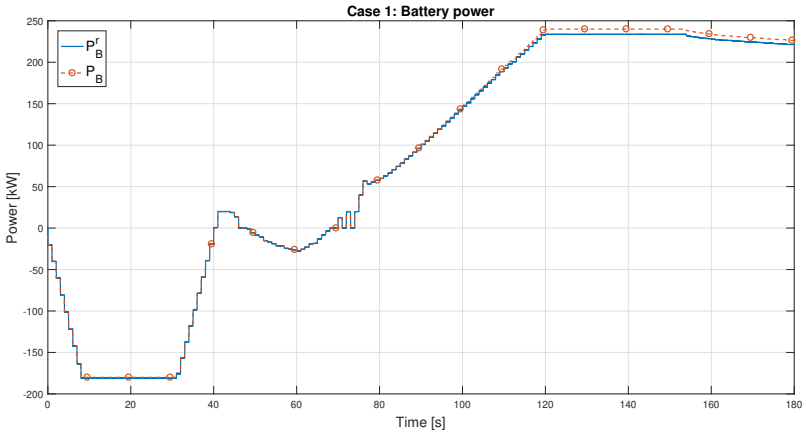

Figure 4.5: Case 1 - The calculated optimal value of P_B^+ and P_B^- is represented by P_B (dotted red line), and the P_B^r implemented by the low level controller in the SimPowerSystems simulation (blue line).

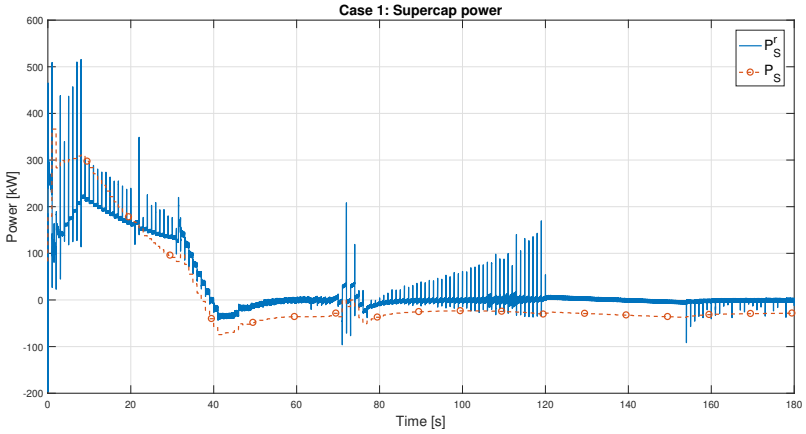

Figure 4.6: Case 1 - The calculated optimal value of P_S^+ and P_S^- is represented by P_S (dotted red line), and the P_S^r implemented by the low level controller in the SimPowerSystems simulation (blue line).

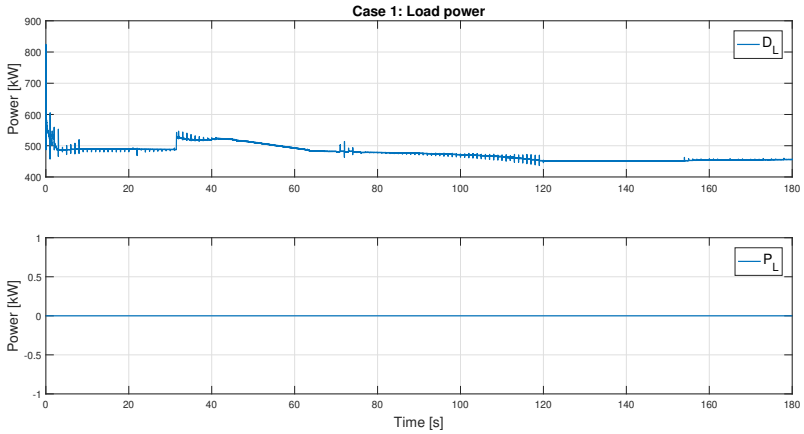

Figure 4.7: Case 1 - The power D_L demanded by the load and the calculated power P_L to be shut down from it.

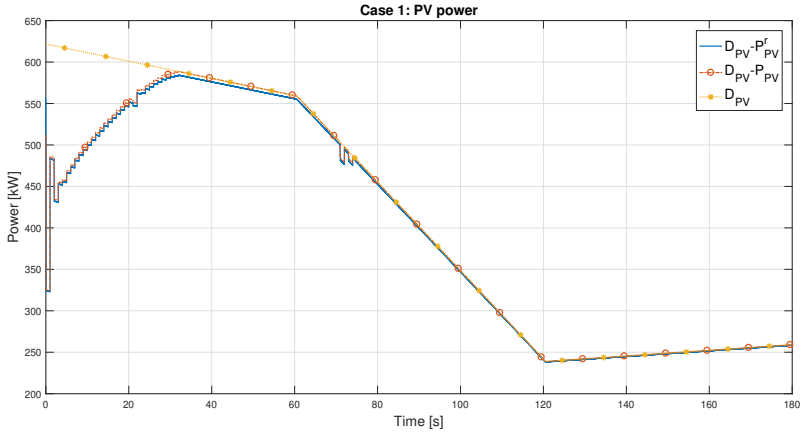

Figure 4.8: Case 1 - The available power D_{PV} from the renewables (yellow dotted line with stars), the calculated optimal reference $D_{PV} - P_{PV}$ (dotted red line with circles) and the $D_{PV} - P^r_{PV}$ implemented by the low level controller in the SimPowerSystems simulation (blue line).

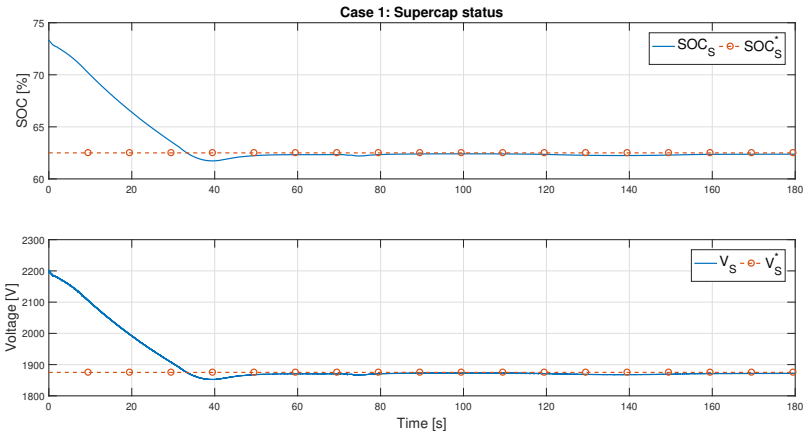

Figure 4.9: Case 1 - The SOC and voltage of the supercapacitor (blue line) with respect to their reference values (dotted red line).

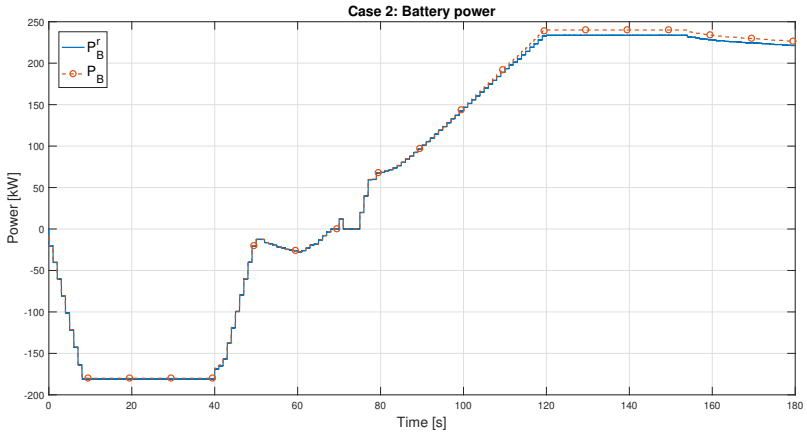

Figure 4.10: Case 2 - The calculated optimal value of P_B^+ and P_B^- is represented by P_B (dotted red line), and the P_B implemented by the low level controller in the SimPowerSystems simulation (blue line).

Case study 3

Here a situation where initially the supercapacitor has a level of charge which is lower than its reference is considered, as can be seen in Fig. 4.19. With such initial conditions, the supercapacitor has to be charged; there is no need to curtail power from the PV array, as shown by Fig. 4.18. For the same reasons, the battery will move to discharge mode and then, once the level of energy into the supercapacitor is the desired one, it will recharge itself to reach its desired level of energy. Therefore, it moves to a discharge mode as a consequence of the decrease of the PV power (see Fig. 4.15). Fig. 4.16 shows the effective tracking of the supercapacitor power output with respect to its reference, and it ends up on the reaching and the keeping of the desired level of energy (see Fig. 4.19).

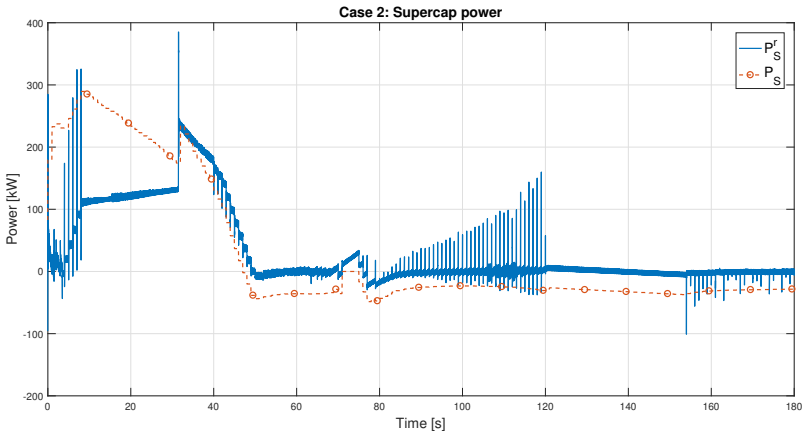

Figure 4.11: Case 2 - The calculated optimal value of P_S^+ and P_S^- is represented by P_S (dotted red line), and the P_S implemented by the low level controller in the SimPowerSystems simulation (blue line).

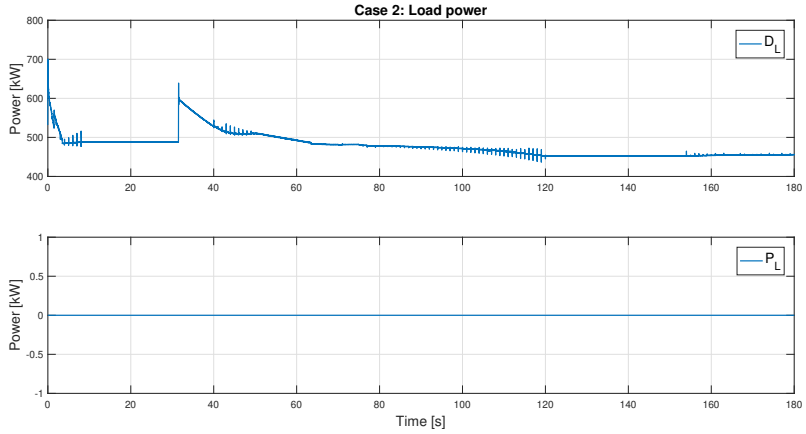

Figure 4.12: Case 2 - The power D_L demanded by the load and the calculated power P_L to be shut down from it.

Case study 4

Case study 4 has initial conditions similar to the ones of case studies 1 and 2, with the state of charge of the supercapacitor which is higher than its desired reference. To better validate the proposed optimization strategy, a more complex scenario is introduced. Indeed, both the profiles for PV power and load power are time-varying, and a pulse of about 50% of the current value is introduced in the load profile. To better underline the effectiveness of the proposed approach, the prediction horizon for state variables and control inputs is reduced to 5 time steps. Moreover, a higher penalization is given to the possibility to curtail the PV power. The proposed load profile is depicted in Fig. 4.22; in the same figure, it is shown that also in this case study there is no need to partially cut the demanded load. Fig. 4.20, 4.21 and 4.23 complete the description of the power profiles of the devices. Fig. 4.24 shows that the proposed optimization strategy fulfill the target to provide power to the load and in the meanwhile maintain a desired state

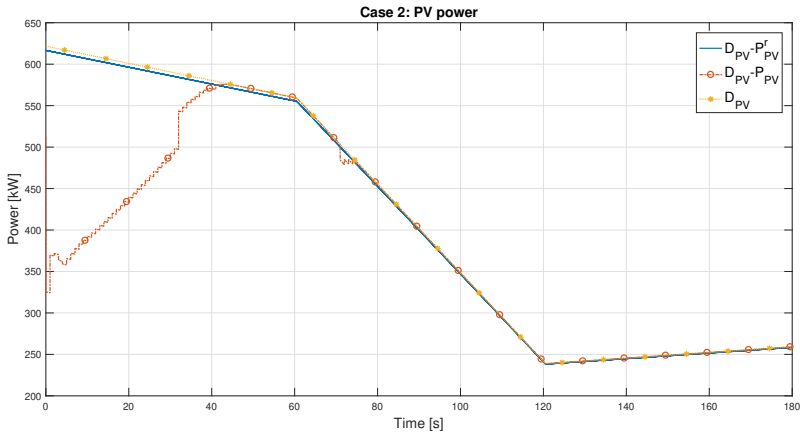

Figure 4.13: Case 2 - The available power D_{PV} from the renewables (yellow dotted line with stars), the calculated optimal reference $D_{PV} - P_{PV}$ (dotted red line with circles) and the $D_{PV} - P^r_{PV}$ implemented by the low level controller in the SimPowerSystems simulation (blue line).

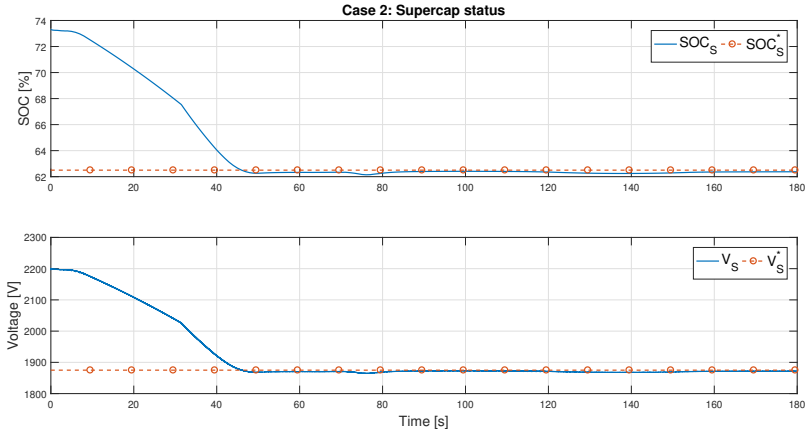

Figure 4.14: Case 2 - The SOC and voltage of the supercapacitor (blue line) with respect to their reference values (dotted red line).

of charge in the supercapacitor, even when the profiles of PV and load power generates more complex scenarios. Indeed, the energy level is not the desired one only when a high power unbalance takes place and the battery is forced not to have high variations. However, the optimization strategy succeeds to restore the desired set of circumstances.

Case studies comparison

All the considered cases show the capability of the system to perform well in the possible range of situations, even in case of errors due to modeling of parameters, as the efficiency, or of the available degrees of freedom, as in the case of unavailability to curtail PV power. Fig. 4.25 and 4.27 show the stability of the DC grid, which is ensured by the low level controllers in each case study: the variations on the value of the power load depends on the voltage DC grid variations. Except for the start-up, the voltage values are always

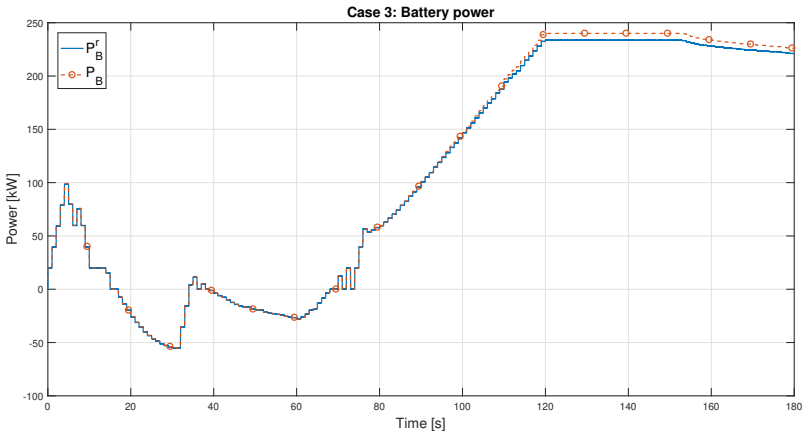

Figure 4.15: Case 3 - The calculated optimal value of P_B^+ and P_B^- is represented by P_B (dotted red line), and the P_B implemented by the low level controller in the SimPowerSystems simulation (blue line).

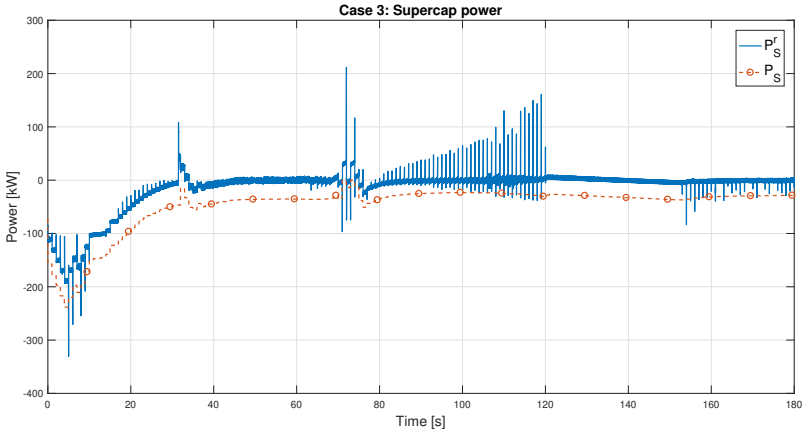

Figure 4.16: Case 3 - The calculated optimal value of P_S^+ and P_S^- is represented by P_S (dotted red line), and the P_S implemented by the low level controller in the SimPowerSystems simulation (blue line).

inside an accepted limits for the variations, i.e. the expected values plus or minus 10%.

It must to be noticed that the peaks taking place for the power P_S in transient time are due to the fast reply of the implemented low level controllers, simulated in detailed switched models, with no filters acting to reduce them. A comparison with respect to the behavior of P_B can better highlight how the target to enlarge battery lifetime is reached, since no overshoot or undershoot that can harm it are introduced in such curves due to the limitations on power variations. In real situations, other electronic devices acting as filters will be introduced in case the physical properties of the materials could not support such spikes.

A point that has to be highlighted is that, thanks to the hypothesis on the correct sizing of the different components of the DC MicroGrid, there is no need to shut down a part of the load (see Fig. 4.6, 4.11 and 4.16, 4.21).

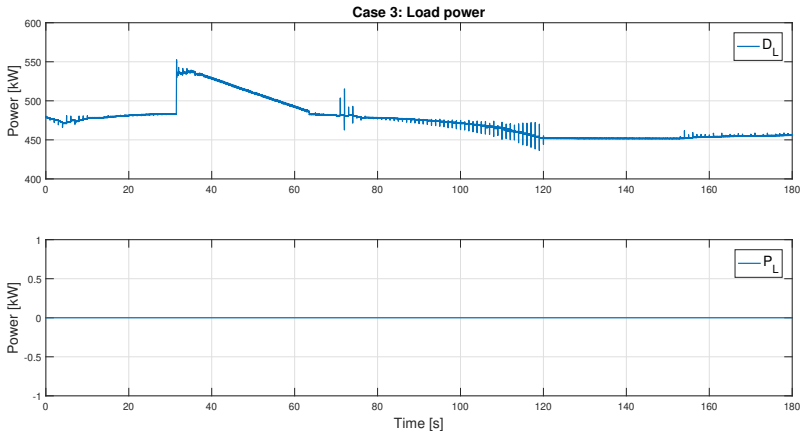

Figure 4.17: Case 3 - The power D_L demanded by the load and the calculated power P_L to be shut down from it.

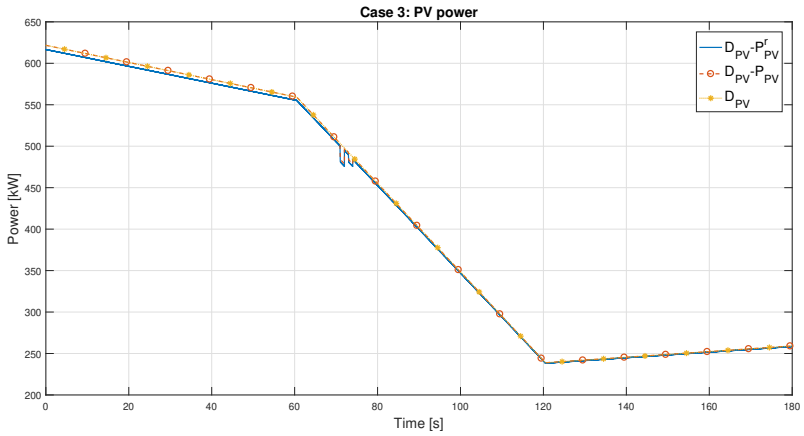

Figure 4.18: Case 3 - The available power D_{PV} from the renewables (yellow dotted line with stars), the calculated optimal reference $D_{PV} - P_{PV}$ (dotted red line with circles) and the $D_{PV} - P^r_{PV}$ implemented by the low level controller in the SimPowerSystems simulation (blue line).

Finally, the last comparison among the proposed power management controllers is the needed computation time. As explained in Section 4.5.3, the above described simulations have been implemented in a rather powerful computer, in charge of both simulations of the low and the high levels of control. Fig. 4.26 and 4.27 describe the time needed by the power management to solve the optimization problem at each call. As it is possible to see, such problem is almost everywhere solved with a computational time that can be neglected with respect to the considered step time.

4.5.5 Experimental results

As described in Section 4.5.4, the proposed power management controller performs well when it is implemented in a computer with good performances. Target here will be to show that it can be implemented also in a low cost hardware, with a reasonable loss

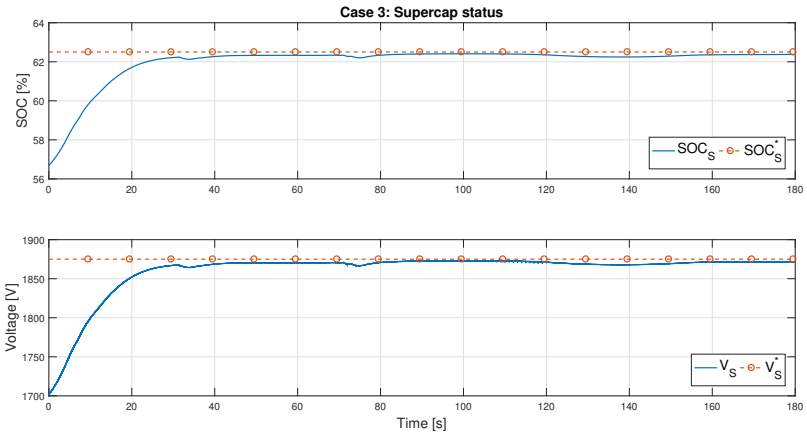

Figure 4.19: Case 3 - The SOC and voltage of the supercapacitor (blue line) with respect to their reference values (dotted red line).

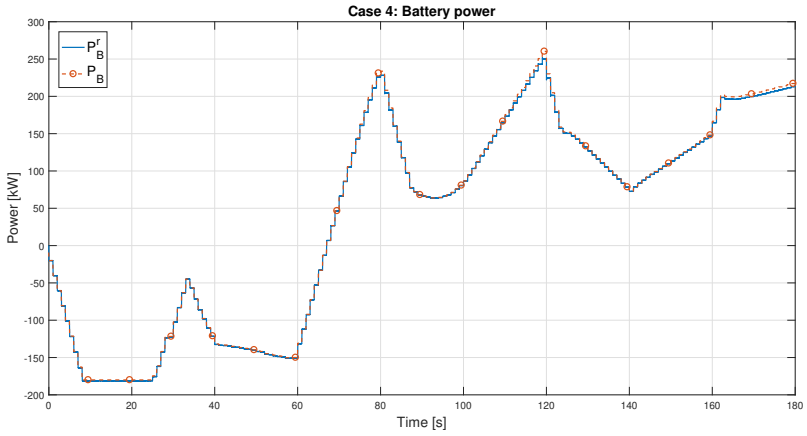

Figure 4.20: Case 4 - The calculated optimal value of P_B^+ and P_B^- is represented by P_B (dotted red line), and the P_B implemented by the low level controller in the SimPowerSystems simulation (blue line).

of performances with respect to the challenging considered time steps. As already described, the Raspberry Pi in Fig. 4.28 has been chosen to check such controller feasibility. Fig. 4.28 depicts the whole experimental setup: the Raspberry Pi implements the high level controller, and it is connected through a LAN and an ethernet cable to another computer performing simulations of the electrical grid using SimPowerSystems. The low level controllers are calculated according to the set points received from the higher controller. As mentioned in Section 4.4.2, a MIQP equivalent MISOCP problem is coded for allowing the Raspberry Pi utilization.

The optimization in the Raspberry Pi is performed in real time, while the simulation of the electrical grid in Simulink SymPowerSystems is 10 to 100 times slower than the real time. Except for the initialization step, where the high level controller values are supposed to be given, in Simulink time (hence 10 to 100 real time seconds) for each step k of the Raspberry Pi time the measurements of the state are sent to the Raspberry

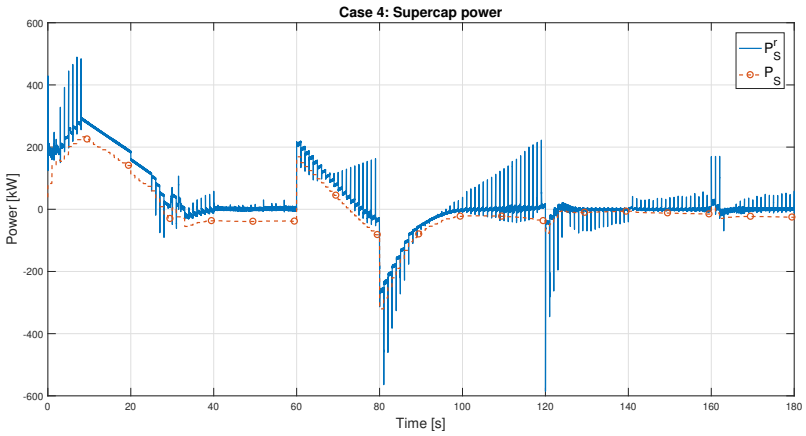

Figure 4.21: Case 4 - The calculated optimal value of P_S^+ and P_S^- is represented by P_S (dotted red line), and the P_S implemented by the low level controller in the SimPowerSystems simulation (blue line).

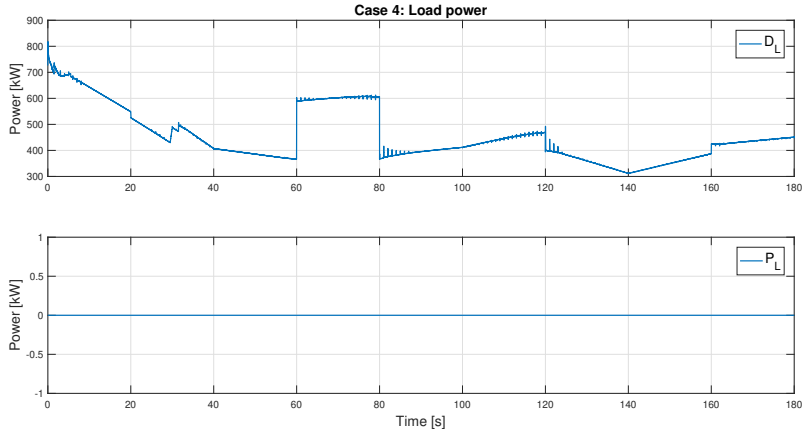

Figure 4.22: Case 4 - The power D_L demanded by the load and the calculated power P_L to be shut down from it.

Pi and a clock timer is launched. The optimization is performed in the Raspberry Pi and the obtained reference values are transmitted back to Simulink: as a consequence, the aforementioned timer is stopped and its value is stored. Using delay blocks, the provided references are then applied only after a Simulink time equivalent to the value of the clock timer, which considers the Raspberry Pi computational and transmission time. The obtained system is then emulating a real-time one. The choice to let the lower level system use the references as if the higher level controller was instantaneous has been done to the purpose to operate a proper comparison among the simulations (where the high level controller can be considered as instantaneous) and the experimental test. The choice to impose a maximum value of 0.5 s for the computational time has been done to relax the constraints on the transmission time, and for allowing not to consider bandwidth problems since the data exchanged are few bytes through an ethernet connection and the missing 0.5 s are sufficient to ensure a successful data transmission.

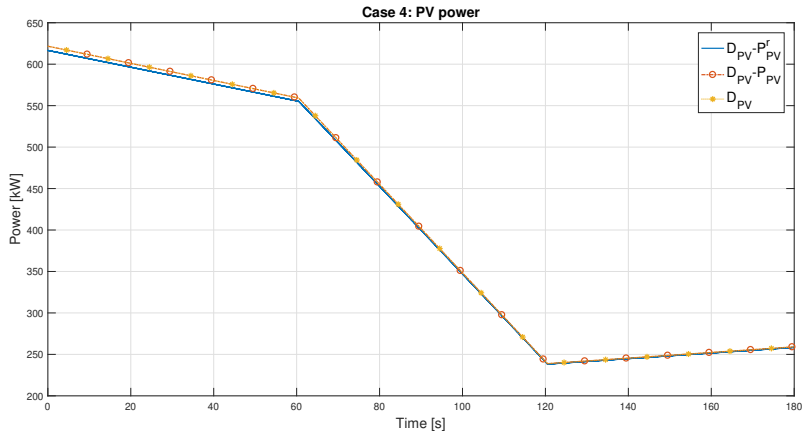

Figure 4.23: Case 4 - The available power D_{PV} from the renewables (yellow dotted line with stars), the calculated optimal reference $D_{PV} - P_{PV}$ (dotted red line with circles) and the $D_{PV} - P^r_{PV}$ implemented by the low level controller in the SimPowerSystems simulation (blue line).

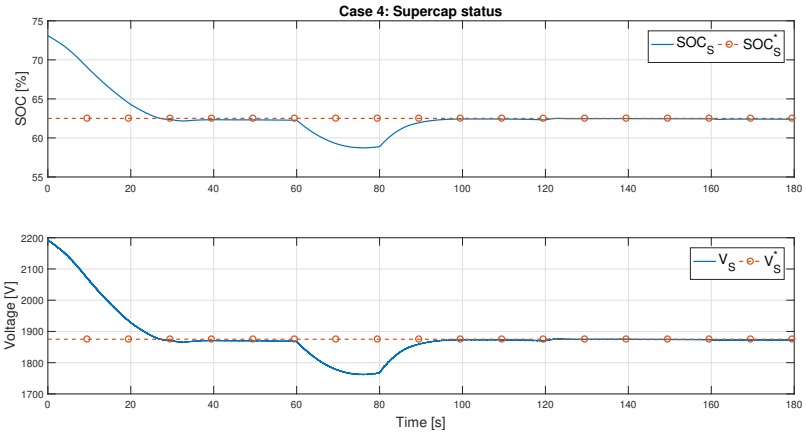

Figure 4.24: Case 4 - The SOC and voltage of the supercapacitor (blue line) with respect to their reference values (dotted red line).

The considered scenarios here are the Case Studies 1 and 3, respectively for the Experimental setup 1 and 2, in order to cover both the case of need of charge and discharge for the supercapacitor. Upper discharge bounds for the battery are set to 360 kW.

Fig. 4.29 depicts the response time of the power management; as it is possible to see, most of the time it is below the threshold of 0.5 second (in violet), which has been set as the maximum limit for the calculations. In case it is reached, the controller then sends one of the feasible solutions it has found at that time; it is important to be noticed that it probably is not the optimal one. Taking a look together at Fig. 4.29, 4.32 and 4.37, the correlation between the high step variation in the load power demand and the related higher computational time clearly appears. As for the simulation results described in Section 4.5.4, the peaks taking place for the power P_S in Fig. 4.31 and 4.36 in

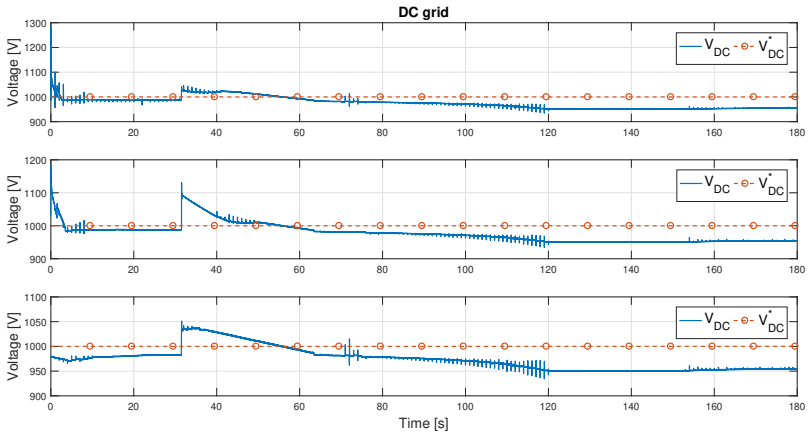

Figure 4.25: The DC grid voltage (blue lines) with respect to its reference (dotted red lines) in the case studies 1, 2 and 3, respectively.

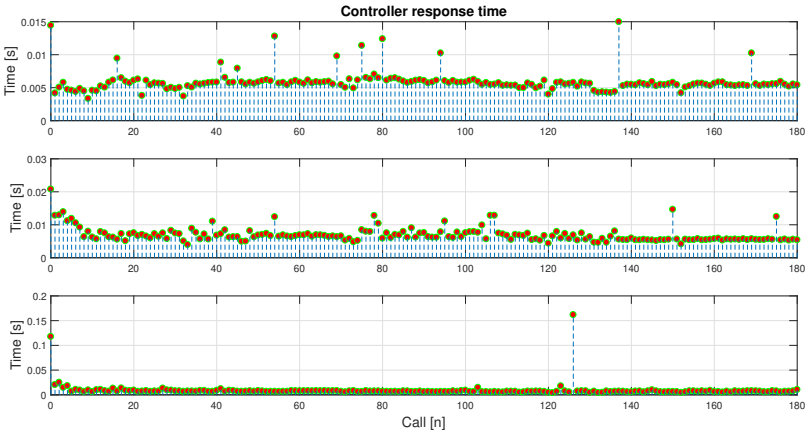

Figure 4.26: The power management controller response time for the case studies 1, 2 and 3, respectively, when it has been implemented in a computer with 16Go of RAM and a Intel i7-7700k @4.2 GHz CPU.

transient time are due to the fast reply of the implemented low level controllers, simulated in detailed switched models, with no filters acting to reduce them. The choice not to implement such filters has been done to better validate the proposed power management controller, which results robust to the fast variation due to the secondary effects as parasitic currents or perturbation acting on the DC grid. The same small perturbations acting on the DC grid and creating its voltage variation generate the perturbations on the DC load, as depicted in Fig. 4.32 and 4.37.

Then, as a consequence, the solution provided in correspondence of such requests are not optimal, as seen for example in Fig. 4.30 and 4.33, where, around the 40 second, the controller chooses to curtail PV power while a better option would have been not to do it in order to increase the power charge ratio. Also, that generates a discharge power output for the supercapacitor, which then stops its energy level reference tracking (see Fig. 4.34). Similar error rises in Experimental test 2.

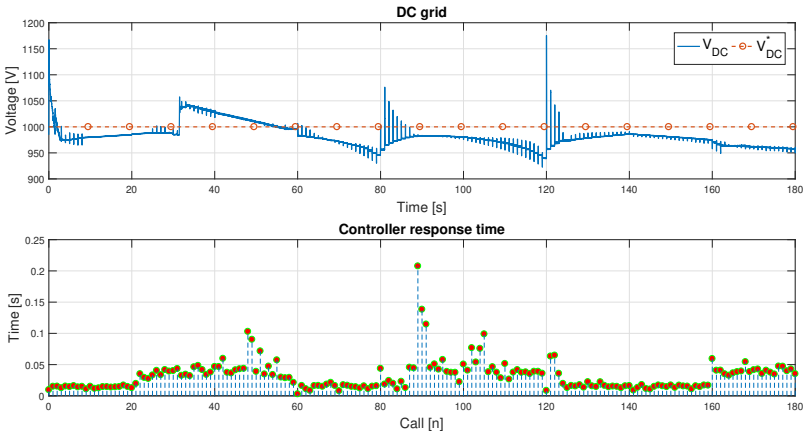

Figure 4.27: The DC grid voltage (blue lines) with respect to its reference (dotted red lines) in the case study 4 and power management controller response time .

Test	$\overline{P}_B$	$\overline{P}_S$	$\overline{P}_L$	$\overline{P}_{PV}$	SOC	CPU
Sim1	Fig. 4.5	Fig. 4.6	Fig. 4.7	Fig. 4.8	Fig. 4.9	Fig. 4.26
Sim2	Fig. 4.10	Fig. 4.11	Fig. 4.12	Fig. 4.13	Fig. 4.14	Fig. 4.26
Sim3	Fig. 4.15	Fig. 4.16	Fig. 4.17	Fig. 4.18	Fig. 4.19	Fig. 4.26
Sim4	Fig. 4.20	Fig. 4.21	Fig. 4.22	Fig. 4.23	Fig. 4.24	Fig. 4.27
Exp1	Fig. 4.30	Fig. 4.31	Fig. 4.32	Fig. 4.33	Fig. 4.34	Fig. 4.29
Exp2	Fig. 4.35	Fig. 4.36	Fig. 4.37	Fig. 4.38	Fig. 4.39	Fig. 4.29

Table 4.4: A summary of the proposed tests with respect to the figures describing them.

In the considered conditions, for both the experimental tests the power management controller is still able to manage the optimization problem such to recover energy from the renewable source, to charge/discharge the supercapacitor in order to have the desired level of energy to ensure the highest degree of controllability for the system, and to charge the battery in case of available power.

Tables 4.4 and 4.5 help the reader in making a proper comparison among the results obtained in the different proposed tests. Since the proposed situations differ in initial condition and acting disturbances, a numerical comparison is not possible: however, Table 4.4 allows for an easier comparison of the developed test on each single variable, and Table 4.5 describes how the whole targets are successfully reached in each test.

4.6 Conclusions

In this paper, a dynamic power management controller for a DC MicroGrid is introduced, to the purpose to calculate the references to be given to the lower level controllers of the real devices in charge of physically acting on the system ensuring grid stability, both in voltage and power balance sense. A receding horizon technique is utilized in order to use prediction of the disturbances acting on the system and to make the state variables reach the desired values.

	Sim1	Sim2	Sim3	Sim4	Exp1	Exp2
Problem feasibility	✓	✓	✓	✓	✓	✓
PV: stored power	✓	✓	✓	✓	✓	✓
DC load correctly fed	✓	✓	✓	✓	✓	✓
Supercap: reached reference	✓	✓	✓	✓	✓	✓
DC grid stability	✓	✓	✓	✓	✓	✓

Table 4.5: A summary of the reached targets in the different proposed tests.

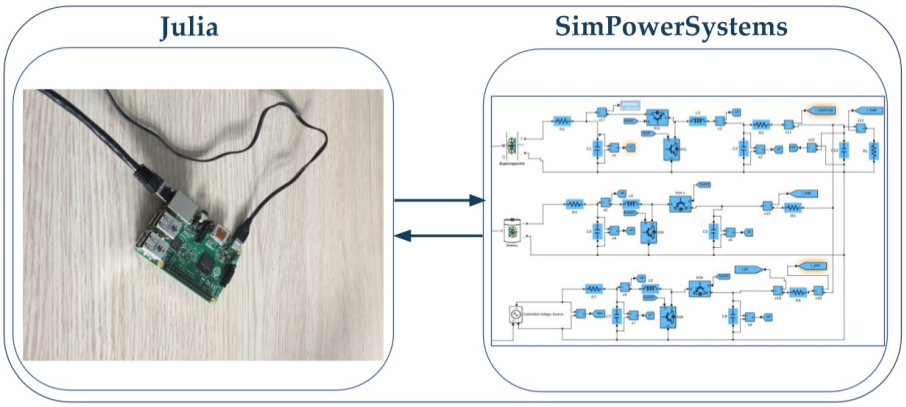

Figure 4.28: The experimental setup; a Raspberry Pi implementing the high level controller connected through LAN and an ethernet cable to another computer performing simulations of the electrical grid using SimPowerSystems and implementing the low level controllers according to the set points received from the higher controller.

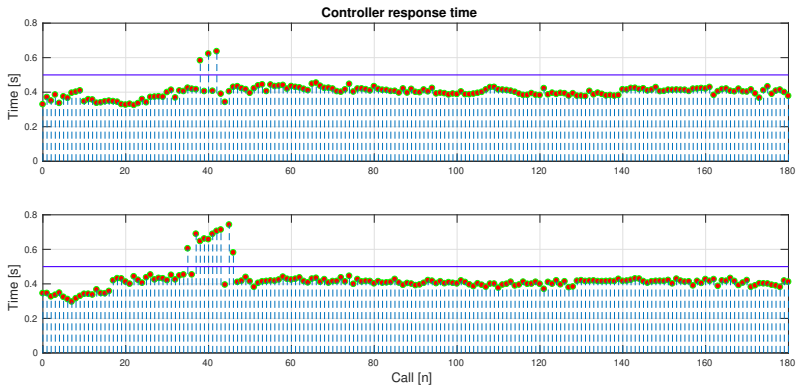

Figure 4.29: The power management controller response time for the considered cases, respectively, when it has been implemented in the Raspberry Pi.

The developed controller allows to take into account the different characteristics and constraints of several physical devices composing the grid and to use current and predicted information about power flowing to load or from renewables. The results show that the control strategy correctly fits the target to describe and predict the power flow of a DC MicroGrid and then that the obtained optimized power references can be used by the

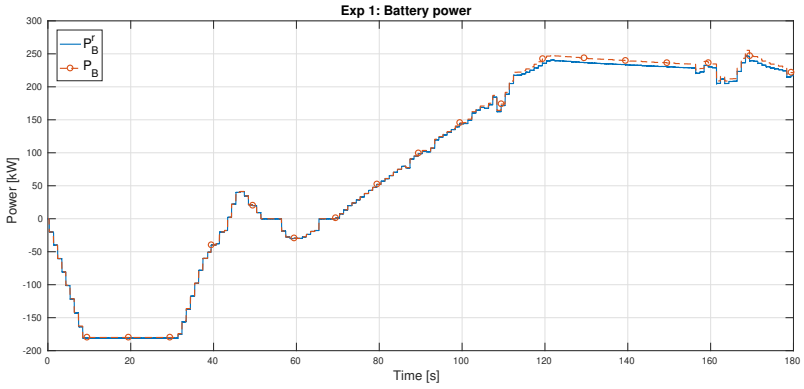

Figure 4.30: Exp 1 - The calculated optimal value of P_B^+ and P_B^- is represented by P_B (dotted red line), and the P_B implemented by the low level controller in the SimPowerSystems simulation (blue line).

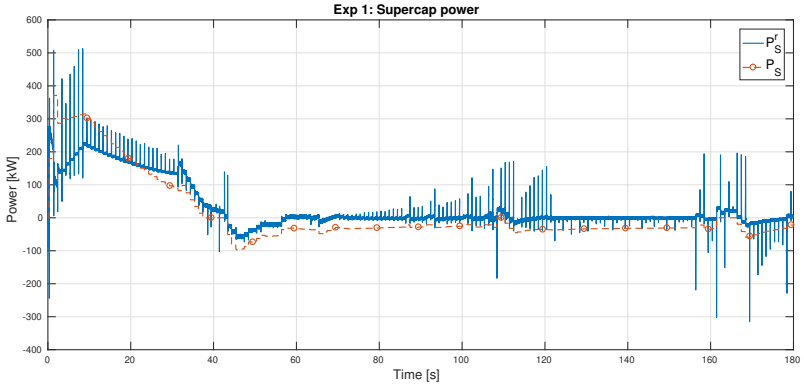

Figure 4.31: Exp 1 - The calculated optimal value of P_S^+ and P_S^- is represented by P_S (dotted red line), and the P_S implemented by the low level controller in the SimPowerSystems simulation (blue line).

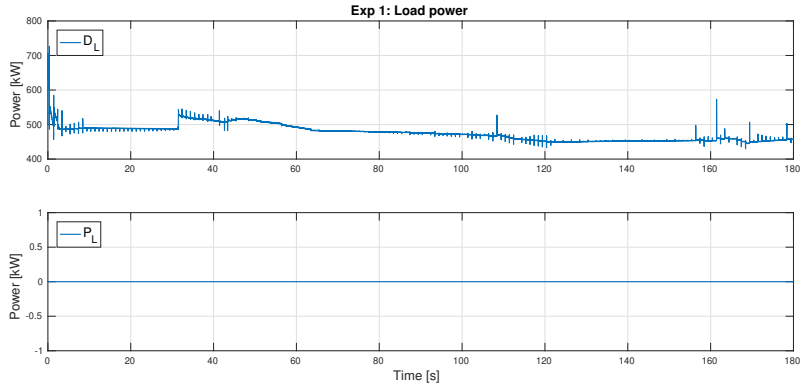

Figure 4.32: Exp 1 - The power D_L demanded by the load and the calculated power P_L to be shut down from it.

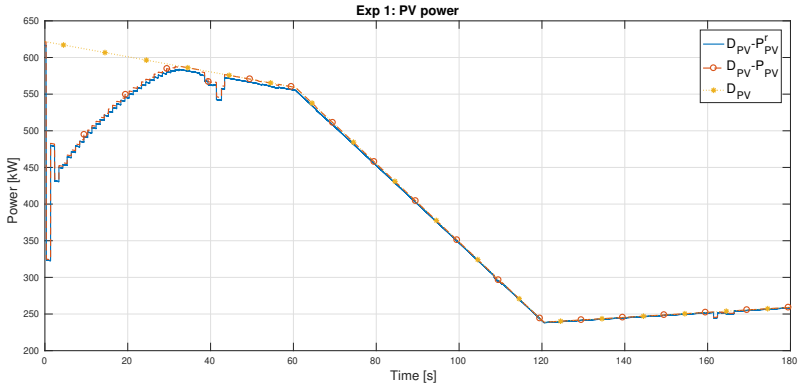

Figure 4.33: Exp 1 - The available power D_{PV} from the renewables (yellow dotted line with stars), the calculated optimal reference $D_{PV} - P_{PV}$ (dotted red line with circles) and the $D_{PV} - P^r_{PV}$ implemented by the low level controller in the SimPowerSystems simulation (blue line).

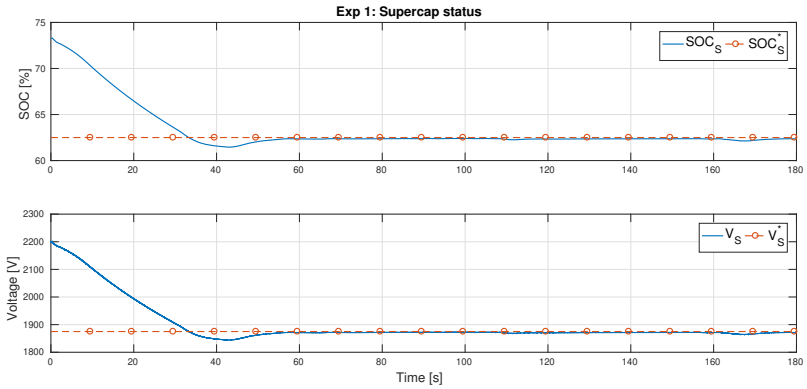

Figure 4.34: Exp 1 - The SOC and voltage of the supercapacitor (blue line) with respect to their reference values (dotted red line).

low level controller to ensure grid stability. The developed controller is shown to perform well even when it is implemented in a low cost hardware, which makes it interesting for a large variety of applications. The obtained good performances allow for a revisiting of the nowadays adopted primary level controllers, envisaging the possibility to obtain better results using the optimal control approaches that are only used at higher level.

The proposed scheme fills an important gap of intermediate level controllers happening now. Most solutions found in literature are based on simple heuristic solutions for power management that satisfies the power flow, and in general the lifespan of the battery is neglected, leaving most of the power equilibrium duty to it. The proposed scheme is fast enough to be implemented between the EMS and the low level controllers, and then it avoids current oversizing of components that is a consequence of such poor heuristic schemes. Moreover, **H** and **L** controllers together present a stable framework for the EMS that can then just deal with economic and communicating aspects, without any concern about physical ones.

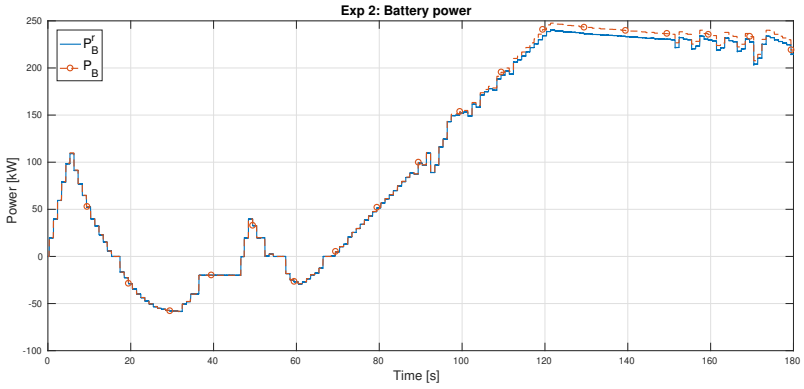

Figure 4.35: Exp 2 - The calculated optimal value of P_B^+ and P_B^- is represented by P_B (dotted red line), and the P_B implemented by the low level controller in the SimPowerSystems simulation (blue line).

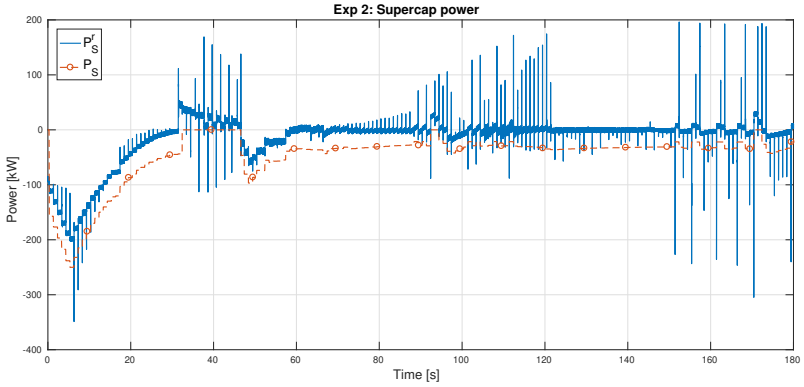

Figure 4.36: Exp 2 - The calculated optimal value of P_S^+ and P_S^- is represented by P_S (dotted red line), and the P_S implemented by the low level controller in the SimPowerSystems simulation (blue line).

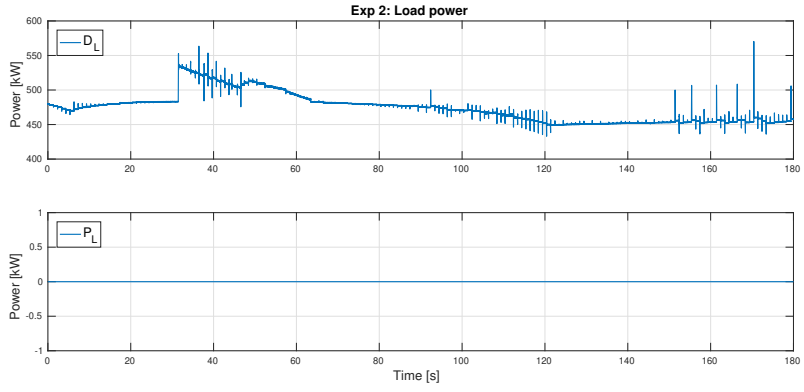

Figure 4.37: Exp 2 - The power D_L demanded by the load and the calculated power P_L to be shut down from it.

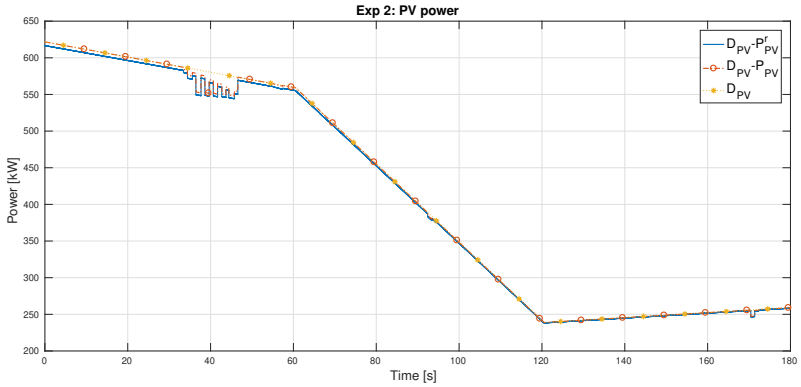

Figure 4.38: Exp 2 - The available power D_{PV} from the renewables (yellow dotted line with stars), the calculated optimal reference $D_{PV} - P_{PV}$ (dotted red line with circles) and the $D_{PV} - P^r_{PV}$ implemented by the low level controller in the SimPowerSystems simulation (blue line).

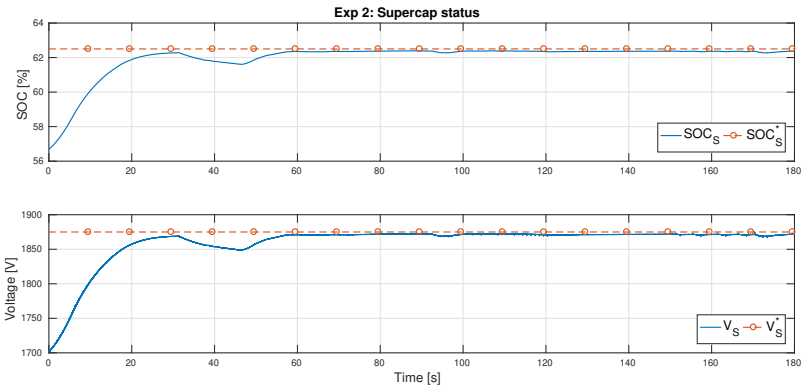

Figure 4.39: Exp 2 - The SOC and voltage of the supercapacitor (blue line) with respect to their reference values (dotted red line).

Chapter 4. Bibliography

[77] M. R. Almassalkhi and I. A. Hiskens. Model-Predictive Cascade Mitigation in Electric Power Systems With Storage and Renewables; Part I: Theory and Implementation. *IEEE Transactions on Power Systems*, 30(1):67–77, Jan 2015.

[78] M. R. Almassalkhi and I. A. Hiskens. Model-Predictive Cascade Mitigation in Electric Power Systems With Storage and Renewables; Part II: Case-Study. *IEEE Transactions on Power Systems*, 30(1):78–87, Jan 2015.

[79] R. F. Bastos, T. Dragicevic, J. M. Guerrero, and R. Q. Machado. Decentralized control for renewable DC Microgrid with composite energy storage system and UC voltage restoration connected to the grid. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 2016–2021, Oct 2016.

[80] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

[81] A. Bidram, A. Davoudi, F. L. Lewis, and J. M. Guerrero. Distributed Cooperative Secondary Control of Microgrids Using Feedback Linearization. *IEEE Transactions on Power Systems*, 28(3):3462–3470, 2013.

[82] S. Bracco, F. Delfino, F. Pampararo, M. Robba, and M. Rossi. A dynamic optimization-based architecture for polygeneration microgrids with tri-generation, renewables, storage systems and electrical vehicles. *Energy Conversion and Management*, 96:511 – 520, 2015.

[83] E. F. Camacho and C. Bordons. *Model predictive control*. Springer, 2007.

[84] E. F. Camacho, T. Samad, M. Garcia-Sanz, and I. Hiskens. Control for renewable energy and smart grids. Grand Challenges for Control, 2010.

[85] F. Delfino, R. Minciardi, F. Pampararo, and M. Robba. A Multilevel Approach for the Optimal Control of Distributed Energy Resources and Storage. *IEEE Transactions on Smart Grid*, 5(4):2155–2162, July 2014.

[86] T. Dragicevic, X. Lu, J. Vasquez, and J. Guerrero. DC Microgrids-Part II: A Review of Power Architectures, Applications, and Standardization Issues. *Power Electronics, IEEE Transactions on*, 31(5):3528–3549, May 2016.

[87] T. Dragicevic, J. Vasquez, J. Guerrero, and D. Skrlec. Advanced LVDC Electrical Power Architectures and Microgrids: A step toward a new generation of power distribution networks. *Electrification Magazine, IEEE*, 2(1):54–65, March 2014.

[88] I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[89] H. Farhangi. The path of the smart grid. *Power and Energy Magazine, IEEE*, 8(1):18–28, January 2010.

[90] M. Farina, A. Guagliardi, F. Mariani, C. Sandroni, and R. Scattolini. Model predictive control of voltage profiles in mv networks with distributed generation. *Control Engineering Practice*, 34(Supplement C):18 – 29, 2015.

[91] A. Garulli, S. Paoletti, and A. Vicino. Models and Techniques for Electric Load Forecasting in the Presence of Demand Response. *IEEE Transactions on Control Systems Technology*, 23(3):1087–1097, May 2015.

[92] W. Greenwell and A. Vahidi. Predictive Control of Voltage and Current in a Fuel Cell-Ultracapacitor Hybrid. *IEEE Transactions on Industrial Electronics*, 57(6):1954–1963, June 2010.

[93] J. Guerrero, J. Vasquez, J. Matas, L. de Vicuna, and M. Castilla. Hierarchical Control of Droop-Controlled AC and DC Microgrids; A General Approach Toward Standardization. *Industrial Electronics, IEEE Transactions on*, 58(1):158–172, Jan 2011.

[94] A. Iovine, G. Damm, E. De Santis, and M. D. Di Benedetto. Management controller for a dc microgrid integrating renewables and storages. *IFAC-PapersOnLine*, 50(1):90 – 95, 2017. 20th IFAC World Congress.

[95] A. Iovine, M. Jimenez Carrizosa, G. Damm, and P. Alou. Nonlinear control for dc microgrids enabling efficient renewable power integration and ancillary services for ac grids. *IEEE Transactions on Power Systems*, pages 1–1, 2018.

[96] A. Iovine, S. B. Siad, G. Damm, E. De Santis, and M. D. Di Benedetto. Nonlinear control of an AC-connected DC microgrid. In *Industrial Electronics Society, IECON 2016 - 42nd Annual Conference of the IEEE*, 24-27 October 2016.

[97] A. Iovine, S. B. Siad, G. Damm, E. D. Santis, and M. D. D. Benedetto. Nonlinear control of a dc microgrid for the integration of photovoltaic panels. *IEEE Transactions on Automation Science and Engineering*, 14(2):524–535, April 2017.

[98] E. Jimenez, M. J. Carrizosa, A. Benchaib, G. Damm, and F. Lamnabhi-Lagarrigue. A new generalized power flow method for multi connected DC grids. *International Journal of Electrical Power and Energy Systems*, 74:329 – 337, 2016.

[99] M. Jimenez Carrizosa, F. D. Navas, G. Damm, and F. Lamnabhi-Lagarrigue. Optimal power flow in multi-terminal HVDC grids with offshore wind farms and storage devices. *International Journal of Electrical Power and Energy Systems*, 65:291 – 298, 2015.

[100] P. Kundur, N. J. Balu, and M. G. Lauby. *Power system stability and control*. McGraw-Hill, 1994.

[101] R. H. Lasseter. Microgrids And Distributed Generation. *Intelligent Automation and Soft Computing*, 16(2):225–234, 2010.

[102] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1):193 – 228, 1998. International Linear Algebra Society (ILAS) Symposium on Fast Algorithms for Control, Signals and Image Processing.

[103] L. Meng, Q. Shafiee, G. F. Trecate, H. Karimi, D. Fulwani, X. Lu, and J. M. Guerrero. Review on Control of DC Microgrids and Multiple Microgrid Clusters. *IEEE Journal of Emerging and Selected Topics in Power Electronics*, 5(3):928–948, Sept 2017.

[104] D. E. Olivares, C. A. Canizares, and M. Kazerani. A centralized energy management system for isolated microgrids. *IEEE Transactions on Smart Grid*, 5(4):1864–1875, July 2014.

[105] D. E. Olivares, J. D. Lara, C. A. Canizares, and M. Kazerani. Stochastic-predictive energy management system for isolated microgrids. *IEEE Transactions on Smart Grid*, 6(6):2681–2693, Nov 2015.

[106] A. Parisio, E. Rikos, and L. Glielmo. A model predictive control approach to microgrid operation optimization. *IEEE Transactions on Control Systems Technology*, 22(5):1813–1827, Sept 2014.

[107] A. Parisio, E. Rikos, and L. Glielmo. Stochastic model predictive control for economic/environmental operation management of microgrids: An experimental case study. *Journal of Process Control*, 43:24–37, 2016.

[108] V. Perelmuter. *Electrotechnical Systems: Simulation with Simulink and SimPowerSystems*. CRC Press, 2012.

[109] I. Prodan and E. Zio. A model predictive control framework for reliable microgrid energy management. *International Journal of Electrical Power and Energy Systems*, 61:399 – 409, 2014.

[110] J. Sandoval-Moreno, G. Besançon, and J. J. Martinez. Model predictive control-based power management strategy for fuel cell/wind turbine/supercapacitor integration for low power generation system. In *Power Electronics and Applications (EPE), 2013 15th European Conference on*, pages 1–10, Sept 2013.

[111] H. Shuai, J. Fang, X. Ai, Y. Tang, J. Wen, and H. He. Stochastic optimization of economic dispatch for microgrid based on approximate dynamic programming. *IEEE Transactions on Smart Grid*, PP(99):1–1, 2018.

[112] J. S. Stein, W. F. Holmgren, J. Forbess, and C. W. Hansen. Pvlib: Open source photovoltaic performance modeling functions for matlab and python. In *2016 IEEE 43rd Photovoltaic Specialists Conference (PVSC)*, pages 3425–3430, June 2016.

[113] L. E. Zubieta. Are microgrids the future of energy?: DC microgrids from concept to demonstration to deployment. *IEEE Electrification Magazine*, 4(2):37–44, June 2016.

Chapter 5

Algorithms for two-time scales stochastic optimization with applications to long term management of energy storage

This is a joint work with Pierre Carpentier, Jean-Philippe Chancelier, Michel De Lara.

Chapter Abstract

In this chapter, we apply Chapter 1 formalism to design algorithms for two time scales stochastic optimization problems arising from long term storage management. Energy storage devices are of major importance to integrate more renewable energies and demand-side management in a new energy mix. However batteries remain costly even if recent market developments in the field of electrical vehicles and stationary storage tend to decrease their cost. We present a stochastic optimization model aiming at minimizing the investment and maintenance costs of batteries for a house with solar panels. For any given capacity of battery it is necessary to compute a charge/discharge strategy as well as maintenance to maximize revenues provided by intraday energy arbitrage while ensuring a long term aging of the storage devices. Long term aging is a slow process while charge/discharge control of a storage handles fast dynamics. For this purpose, we have designed algorithms that take into account this two time scales aspect in the decision making process. They are based on Chapter 1 time decomposition framework. These algorithms are applied to three numerical experiments. First one of them is used to control charge/discharge, aging and renewal of batteries for a house. Results show that it is economically significant to control aging. Second we apply and compare our algorithms one a simple charge/discharge and aging problem, that is a multistage stochastic optimization problem with many time steps. We compare our algorithms to SDP and Stochastic Dual Dynamic Programming and we observe that they are less computationally costly while displaying similar performances on the control of a storage. Finally we show how one

of our algorithm can be used for the optimal sizing of a storage taking into account charge/discharge strategy as well as aging.

Contents

5.1 Introduction

We introduce hereby the reasons to study long term management of energy storage problems and why we use a two time scales stochastic optimization framework to tackle them. Then we present existing literature on these issues.

5.1.1 Context

The integration of renewable energies is of upmost importance to ensure a clean energy production mix that can face the perpetually rising electrical demand. These energies and demand are inherently uncertain as they respectively depend on our environment and on consumers behavior. Electrical storage is used as a buffer to mitigate uncertainties in new electricity grids. Every battery requires a proper management strategy able

to make charge/discharge decisions in an uncertain setting to minimize an economical, environmental or energy criterion. These systems are costly, have a fast dynamical behavior (noticeably changing every minute) but can last multiple years. The revenues they provide and their lifetime are deeply related as a battery that is never used lasts many years but does not allow to save energy while a intensively used battery will last a much smaller amount of years but save more energy every day.

In this chapter we present a two time scales stochastic optimal control formalism to control systems with fast dynamics that affect long term behavior, as it is the case for batteries. Using well known results from discrete time stochastic optimal control and convex analysis theory, we develop two general methods to decompose that kind of problems by time blocks. Based on these theoretical methods, we develop associated numerical algorithms. We apply these algorithms to a battery charge/discharge and renewal management problem, namely two-time scales stochastic dynamic programming and its dual variant. We also combine one of our method with Stochastic Dual Dynamic Programming and Linear Programming to solve an aging aware sizing-control problem of a battery.

5.1.2 Literature review

The management of micro-grids involves different time scales as the dynamic of currents is faster than the dynamics of voltage/power which is faster than the dynamics of energy flows. For this reason, micro-grids control architecture is often divided into hierarchical levels exchanging information at different paces [132]. In this paper we focus on the energy management level (time step 1 minute) and the long term aging level (time step 1 day). We survey literature on energy storage operation and long term aging management using optimization methods.

Stochastic optimization for energy management problems

Stochastic dynamic optimization methods based on the Bellman equation [114] have often been applied to energy storage management. In [137, 126] or [124] the authors apply Stochasic Dynamic Programming with discretized state and control spaces to solve an energy management problem. This method suffers the so called *curses of dimensionality* as introduced in [114, 116, 136] or [119]. Moreover it is demonstrated in [115] that the convergence of the discretization procedure is of course dependent on the number of time stages.

A major contribution to handle a large number of energy storage for an electricity system is the well known Stochastic Dual Dynamic Programming (SDDP) algorithm [134]. This method is adapted to problems with linear dynamics and convex costs. It has been applied to energy management with battery in [133]. Other similar methods have been developed such as Mixed Integer Dynamic Approximation Scheme (MIDAS) [135] or Stochastic Dual Dynamic Integer Programming [143] for non convex problems, in particular those displaying binary variables. These algorithms performance is sensitive to the number of time steps as stated in [131] and [135].

Other classical Stochastic Programming methods are sensible to the number of time stages. Solving a multistage stochastic optimization problem on a scenario tree displays

a complexity exponential in the number of time steps [142].

We present algorithms to decompose, in time, problems displaying many time stages. The algorithms are based on a time block application of the Bellman equation [120]. The motivation is a problem displaying two decisions time scales, but time decomposition also helps to enhance classical methods and algorithms whose performance is sensitive to the number of time steps.

Energy storage aging management

Batteries are expensive equipment whose long term management strategy significantly impacts their economic profitability. The authors of [123] use an energy counting model to model and manage the aging of the battery. It corresponds to measure the equivalent number of full cycles N_{cycles} that a battery makes when it charges and discharges a given amount of energy, for instance when a 10 kWh battery charges or discharges 3 kWh, it performs a number of $\frac{3}{2\times 10}$ cycles as a full cycle exchanges the amount of energy of two times the capacity. The health of the battery is managed in a stochastic infinite horizon setting using Average-Cost Value Iteration [116, 125] that requires a stationary assumption. A more detailed model for NaS batteries is developped in [124] that takes into account depth of discharge (DoD) and temperature in addition to N_{cycles}. This model is too detailed to be embedded in an stochastic optimization energy management system. In [128] the authors first propose an abstract model of battery aging to develop continuous time deterministic optimal control methods. An overview of heuristic methods to handle storage aging in a control framework is provided in [127]. In [130] the authors compare different battery aging models in a stochastic optimal control framework.

In this paper we use the simplest aging model provided in [123]. The main goal is to design algorithms for discrete time finite horizon optimization problems with many time steps and two decision time scales, hence without a stationary assumption (contrary to [123]) and in a stochastic setting (contrary to [127]).

5.2 Stochastic optimization of an energy storage system in a microgrid over the long term

We introduce different methods to solve a "novel class" of stochastic optimization problems, namely two-time scales stochastic optimization problems. Those are optimization problems displaying stochasticity and decisions that have to be made at different paces.

5.2.1 Energy system description and notations

We consider the system sketched in Figure 5.1. This is a micro-grid with the following features:

1. an electrical load, or demand, that is uncertain (right),
2. solar panels producing uncertain renewable electricity (left),
3. a connection to the national grid if self production is not enough to provide electricity to the load (top),

4. an electrical storage to ensure supply demand balance (bottom).

All the equipment exchange electricity though a DC grid. The arrows in Figure 5.1 represent the flow of energy: it is bidirectional in the case of the storage as it can charge and discharge. The central node can be seen as a very small storage on a really fast time scale (milliseconds).

The scope of the chapter is to propose an Energy Management System (EMS) that controls both the charge/discharge and health of the battery so as to minimize the electricity consumption on the national grid while ensuring a good aging for the battery. We present here a model of this stochastic dynamical system used to design algorithms to implement the EMS.

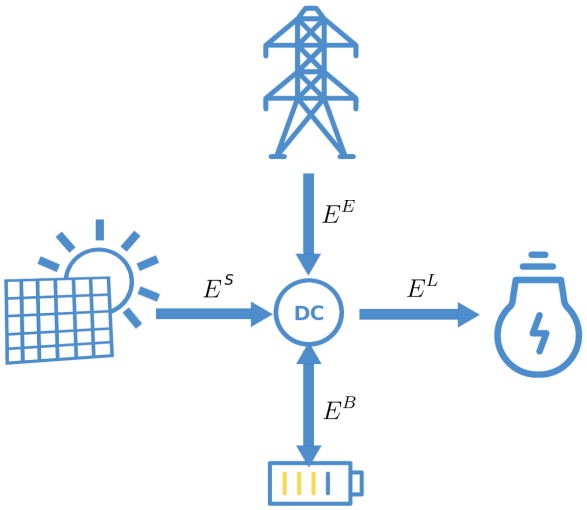

Figure 5.1: The schematic representation of the DC micro-grid to be managed

Notations for two-time scales

For a given constant time interval Δt, let $M \in \mathbb{N}^*$ such that $M+1$ is the number of time steps in a day, e.g. for $\Delta t = 60$ seconds, $M+1 = 1440$. The EMS has to make decisions on two-time scales over a given number of days $D \in \mathbb{N}^*$:

1. one battery charge/discharge decision every minute $m \in \{0, \dots, M\}$ of every day $d \in \{0, \dots, D\}$,

2. one potential renewal of the battery every day $d \in \{0, \dots, D+1\}$.

In order to take into account the two-time scales, we adopt in the sequel the following notation. A variable z will have two time indexes $z_{d,m}$ if it changes every minute m of every day d. An index (d, m) belongs to the following set

$$\mathbb{T} = \{0, \dots, D\} \times \{0, \dots, M\} \cup \{(D+1, 0)\} \,, \tag{5.1}$$

which is a totally ordered set when equipped with the *lexicographical order*

$$(d, m) < (d', m') \iff (d < d') \vee \left(d = d' \wedge m < m'\right) . \tag{5.2}$$

In the sequel, we also use the following notations for describing sequences of variables. For (d, m), and $(d, m') \in \mathbb{T}$ with $m \leqslant m'$:

- the notation $z_{d,m:m'}$ is used to refer to the sequence $(z_{d,m}, \ldots, z_{d,m'-1}, z_{d,m'})$,
- the notation $\mathbb{Z}_{d,m:m'}$ is used to refer to the cartesian product $\prod_{k=m}^{m'} \mathbb{Z}_{d,k}$.

The following time-line illustrates how time flows between two days in our model:

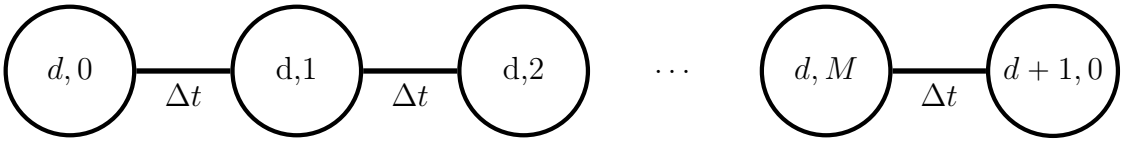

Figure 5.2: Time-line

Uncertainties are modelled as random variables

We write random variables in capital bold letters, like $\boldsymbol{Z}$, to distinguish them from deterministic variables z.

Let $d \in \{0, \ldots, D\}$ be a given day of the whole time span. Every minute $m \in \{0, \ldots, M\}$, two uncertain outcomes materialize at the end of the time interval $[(d, m-1), (d, m))$ (when $m = 0$, the time interval is $[(d-1, M), (d, 0))$, namely,

- $\boldsymbol{E}^S_{d,m}$: the solar production in kWh,
- $\boldsymbol{E}^L_{d,m}$: the electrical demand (load) in kWh.

Another uncertain outcome realizes once a day at the beginning of the time interval $[(d, 0), (d, M))$ namely,

- $\boldsymbol{P}^b_d$: the price of a battery replacement in €/kWh.

We gather all uncertainties in vectors and build a sequence of random variables $\{\boldsymbol{W}_{d,m}\}_{(d,m)\in\mathbb{T}}$ as follows. For all $d \in \{0, \ldots, D\}$ we define:

$$\boldsymbol{W}_{d,m} = \begin{pmatrix} \boldsymbol{E}^S_{d,m} \\ \boldsymbol{E}^L_{d,m} \end{pmatrix}, \text{ for } m \in \{0, \ldots, M-1\}, \quad \text{and} \quad \boldsymbol{W}_{d,M} = \begin{pmatrix} \boldsymbol{E}^S_{d,M} \\ \boldsymbol{E}^L_{d,M} \\ \boldsymbol{P}^b_d \end{pmatrix}. \tag{5.3}$$

We assume in this model that, at the end of the last minute of the day, solar energy production and demand materialize as well as the price of batteries. We call $\mathbb{W}_{d,m}$ the uncertainty space where this uncertainty takes its values.

Remark 24. *We assume in this model that the "slow" randomness $\boldsymbol{P}^b_d$ materializes during a minute at the same time as one solar and load "fast" randomness $(\boldsymbol{E}^S_{d,M}, \boldsymbol{E}^L_{d,M})$. We could add a virtual minute to avoid this.*

The decision maker take decisions based on observation of uncertainties but he cannot anticipate on future uncertainties. To describe this non-anticipativity constraint, for each $(d, m) \in \mathbb{T}$, we introduce the σ-algebra $\mathcal{F}_{d,m}$ generated by all the past noises up to time stage (d, m)

$$\mathcal{F}_{d,m} = \sigma\big(\boldsymbol{W}_{d',m'}; (d', m') \leqslant (d, m)\big) \ . \tag{5.4}$$

Throughout the chapter a random variable indexed by (d, m) refers to a $\mathcal{F}_{d,m}$-measurable random variable (the measurability being imposed by constraints or derived through dynamics equations). The filtration $\{\mathcal{F}_t\}_{t\in\mathbb{T}}$ models the information flow of the problem and the non-anticipativity constraints

$$\forall (d, m) \in \mathbb{T} \quad \sigma(\boldsymbol{U}_{d,m}) \subset \mathcal{F}_{d,m} \tag{5.5}$$

express the fact that random variables $\boldsymbol{U}_{d,m}$ are adapted to the natural filtration $\{\mathcal{F}_{d,m}\}_{(d,m)\in\mathbb{T}}$, i.e $\boldsymbol{U}_{d,m}$ only depends on uncertainties up to time (d, m). We define precisely the decision variables $\boldsymbol{U}_{d,m}$ in the next paragraph.

Decisions are modelled as random variables

As already mentioned, as time goes on, the noise variables are progressively unfolded and made available to the decision maker. This is why, as decisions depend on observations in a stochastic optimal control problem, decision variables are random variables. On a day $d \in \{0, \dots, D\}$, the EMS has to make decisions every minute $m \in \{0, \dots, M\}$ regarding the charge/discharge of the battery as well as the electricity consumption on the national grid. These decisions depend on all the randomness unfolded previously, that is, all the prices of batteries of the previous days and all the solar production and load of the previous days and minutes of the day as described in Equation (5.3)). Finally, these decisions are made at the beginning of the time interval $[(d, m), (d, m + 1))$ (when $m = M$ the time interval is $[(d, M), (d + 1, 0))$:

- $\boldsymbol{E}^E_{d,m}$: the import from the national grid in kWh;
- $\boldsymbol{E}^B_{d,m}$: the battery charge ($\geqslant 0$) or discharge ($\leqslant 0$) in kWh.

At the end of the time interval $[(d, 0), (d, M))$, the decision maker can replace the battery by a new one after observing the current price of batteries $\boldsymbol{P}^b_d$. The decision variable is again a random variable as it depends on the randomness that materialized the previous minutes of previous days:

- $\boldsymbol{R}_d$: the size of the new battery in kWh.

For all $d \in \{0, \dots, D\}$, we group all controls in vectors as follows:

$$\text{for } 0 \leqslant m < M \ , \ \boldsymbol{U}_{d,m} = \begin{pmatrix} \boldsymbol{E}^E_{d,m} \\ \boldsymbol{E}^B_{d,m} \end{pmatrix} \text{ and } \boldsymbol{U}_{d,M} = \begin{pmatrix} \boldsymbol{E}^E_{d,M} \\ \boldsymbol{R}_d \end{pmatrix} \ . \tag{5.6}$$

We assume in this model that at the last minute of the day, the national grid consumption $\boldsymbol{E}^E_{d,M}$ is chosen as well as the potential renewal of the battery $\boldsymbol{R}_d$. In order to take into account the renewal of the battery in a simplified way, we assume in the model that

there is no battery charge at the end of the day. We call $\mathbb{U}_{d,m}$ the control space where the control takes its values.

On Figure 5.1, we observe that all flows converge to a central node named "DC". At a very small time scale (milliseconds), it could be described as a small energy storage to model the voltage stability problem of the DC micro-grid. In practice, it could be implemented by a controlled DC/DC converter and super-capacitors (see [129] and Chapter 4 of this PhD thesis). We do not model this part and assume that the balance constraint (5.7) is ensured in this problem. It states that at a minute time scale we consider that voltage stability is handled and therefore that we have to ensure energy supply/demand balance at the central node. This materializes as the following constraint:

$$\boldsymbol{E}^{E}_{d,m+1} + \boldsymbol{E}^{S}_{d,m+1} = \boldsymbol{E}^{B}_{d,m} + \boldsymbol{E}^{L}_{d,m+1} \, . \tag{5.7}$$

We observe a difference of indexes between $\boldsymbol{E}^{B}_{d,m}$ and the other variables. This is due to the fact that battery charge/discharge is to be implementable on a real system. We need to be able to provide a charge/discharge target to the battery controller at the beginning of the minute. The control variable $\boldsymbol{E}^{E}_{d,m+1}$ is virtual, it is deduced when voltage stability is ensured in the grid at a lower control level. Therefore, we can remove this variable from the optimization problem and replace Equation (5.6) by the following equation:

$$\text{for } 0 \leqslant m < M \, , \;\; \boldsymbol{U}_{d,m} = \left(\boldsymbol{E}^{B}_{d,m}\right) \text{ and } \boldsymbol{U}_{d,M} = \left(\boldsymbol{R}_{d}\right) \, . \tag{5.8}$$

Charge/discharge impacts state of charge and age dynamics

We use a very simple model to describe charging and aging of the battery. We call $\rho_c \in [0,1]$ and $\rho_d \in [0,1]$ respectively the charge and discharge efficiency of the battery. On day $d \in \{0, \dots, D\}$ at minute $m \in \{0, \dots, M\}$, we call $\boldsymbol{B}_{d,m}$ the state of charge of the battery in kWh and $\boldsymbol{H}_{d,m}$ the remaining amount of exchangeable energy in the battery. As we can change a battery only once a day, we call $\boldsymbol{C}_{d}$ the capacity of the battery. For a given capacity a battery can make up to $N_c(\boldsymbol{C}_{d})$ cycles before being considered unusable. At the beginning of the life of the battery with capacity $\boldsymbol{C}_{d}$, the formula $2 \times N_c(\boldsymbol{C}_{d}) \times \boldsymbol{C}_{d}$ gives the maximum health of the battery in kWh. This is the maximum amount of exchangeable energy for the battery. A cycle represents a full charge of the battery plus a full discharge, hence two times the capacity. Every-time we charge or discharge the battery we change its state of charge according to the following dynamical equation.

$$\boldsymbol{B}_{d,m+1} = \boldsymbol{B}_{d,m} - \frac{1}{\rho_d}\boldsymbol{E}^{B-}_{d,m} + \rho_c \boldsymbol{E}^{B+}_{d,m} \, , \tag{5.9a}$$

where $(x)^{+} = 0 \wedge x$ and $(x)^{-} = 0 \wedge (-x)$. Moreover, its amount of exchangeable energy (or health) decreases according to the following dynamical equation:

$$\boldsymbol{H}_{d,m+1} = \boldsymbol{H}_{d,m} - \boldsymbol{E}^{B-}_{d,m} - \boldsymbol{E}^{B+}_{d,m} \, . \tag{5.9b}$$

When the battery health reaches zero, it cannot be used anymore. Hence we have the following health constraint

$$0 \leqslant \boldsymbol{H}_{d,m} \, . \tag{5.10}$$

We constrain the state of charge to remain between two prescribed bounds that are a percentage of the capacity $\boldsymbol{C}_d$:

$$\underline{B} \times \boldsymbol{C}_d \leqslant \boldsymbol{B}_{d,m} \leqslant \overline{B} \times \boldsymbol{C}_d \ . \tag{5.11}$$

Using Equations (5.9a) and (5.9b) repeatedly, we obtain that $\boldsymbol{B}_{d,M}$ (resp. $\boldsymbol{H}_{d,M}$) is a function of $(\boldsymbol{B}_{d,0}, \boldsymbol{U}_{d,0:M-1})$ (resp. $(\boldsymbol{H}_{d,0}, \boldsymbol{U}_{d,0:M-1})$) that we call $f_d^{\boldsymbol{B}}$ (resp. $f_d^{\boldsymbol{H}}$):

$$\boldsymbol{B}_{d,M} = f_d^{\boldsymbol{B}}\left(\boldsymbol{B}_{d,0}, \boldsymbol{U}_{d,0:M-1}\right) , \tag{5.12a}$$

$$\boldsymbol{H}_{d,M} = f_d^{\boldsymbol{H}}\left(\boldsymbol{H}_{d,0}, \boldsymbol{U}_{d,0:M-1}\right) . \tag{5.12b}$$

Battery renewal impacts state dynamics

In this paragraph, we model how the decision to renew the battery using the control $\boldsymbol{U}_{d,M}$ $(= \boldsymbol{R}_d)$ affects the slow state dynamics. If, at the end of day d, we replace the battery with capacity $\boldsymbol{C}_d$ by a new battery of capacity $\boldsymbol{R}_d$, then the capacity $\boldsymbol{C}_{d+1}$ becomes equal to $\boldsymbol{R}_d$. Otherwise the capacity remains unchanged. This gives:

$$\boldsymbol{C}_{d+1} = \begin{cases} \boldsymbol{R}_d \ , & \text{if } \boldsymbol{R}_d > 0 \ , \\ \boldsymbol{C}_d \ , & \text{otherwise .} \end{cases} \tag{5.13}$$

The renewal decision affects as well the fast variables $\boldsymbol{B}_{d,M}$ as a new battery is assumed empty,

$$\boldsymbol{B}_{d+1,0} = \begin{cases} \underline{B} \times \boldsymbol{R}_d \ , & \text{if } \boldsymbol{R}_d > 0 \ , \\ \boldsymbol{B}_{d,M} \ , & \text{otherwise ,} \end{cases} \tag{5.14}$$

and $\boldsymbol{H}_{d,M}$ as a new battery has a renewed health,

$$\boldsymbol{H}_{d+1,0} = \begin{cases} 2 \times N_c(\boldsymbol{R}_d) \times \boldsymbol{R}_d \ , & \text{if } \boldsymbol{R}_d > 0 \ , \\ \boldsymbol{H}_{d,M} \ , & \text{otherwise .} \end{cases} \tag{5.15}$$

We group these state variables at the begining of a day d under the name $\boldsymbol{X}_d$:

$$\boldsymbol{X}_d = \begin{pmatrix} \boldsymbol{C}_d \\ \boldsymbol{B}_{d,0} \\ \boldsymbol{H}_{d,0} \end{pmatrix} . \tag{5.16}$$

We call $\mathbb{X}_d$ the state space where this state takes its values. We build a mapping

$$\begin{aligned} f_d^S : \mathbb{C}_d \times \mathbb{B}_{d,M} \times \mathbb{H}_{d,M} \times \mathbb{U}_{d,M} &\to \mathbb{X}_{d+1} \\ (c, b, h, u) &\mapsto \begin{cases} (u, \underline{B}u, 2N_c(u)u) & \text{if } u > 0 \ , \\ (c, b, h) & \text{otherwise .} \end{cases} \end{aligned} \tag{5.17}$$

We thus obtain a state dynamics equation given by

$$\boldsymbol{X}_{d+1} = f_d^S\big(\boldsymbol{C}_d, \boldsymbol{B}_{d,M}, \boldsymbol{H}_{d,M}, \boldsymbol{U}_{d,M}\big) \text{ using (5.13-5.15) and (5.17)} \tag{5.18a}$$

$$= f_d^S\big(\boldsymbol{C}_d, f_d^{\boldsymbol{B}}\big(\boldsymbol{B}_{d,0}, \boldsymbol{U}_{d,0:M-1}\big), f_d^{\boldsymbol{H}}\big(\boldsymbol{H}_{d,0}, \boldsymbol{U}_{d,0:M-1}\big), \boldsymbol{U}_{d,M}\big) \text{ using (5.12)} \tag{5.18b}$$

$$= f_d\big(\boldsymbol{X}_d, \boldsymbol{U}_{d,0:M}\big) \tag{5.18c}$$

with

$$f_d\Big((c_d, b_{d,0}, h_{d,0}), u_{d,0:M}\Big) = f_d^S\Big(c_d, f_d^{\boldsymbol{B}}\big(b_{d,0}, u_{d,0:M-1}\big), f_d^{\boldsymbol{H}}(h_{d,0}, u_{d,0:M-1}), u_{d,M}\Big)\ . \tag{5.18d}$$

Remark 25. *We note that, in our model, the state dynamics does not depend directly on uncertainties* $\boldsymbol{W}_{t+1}$.

5.2.2 Stochastic optimization problem statement

We have introduced all the requested features to state a two-time scale stochastic optimal control problem dynamics. It remains to define the objective function that the EMS seeks to minimize.

The objective is a discounted expected sum over a finite horizon. We consider the following objective to be minimized:

$$\mathbb{E}\Big[\sum_{d=0}^{D} \gamma_d\Big(\boldsymbol{P}_d^b \times \boldsymbol{R}_d + \sum_{m=0}^{M-1} p_{d,m}^e \times (\boldsymbol{E}_{d,m}^B + \boldsymbol{E}_{d,m+1}^L - \boldsymbol{E}_{d,m+1}^S)^+\Big)\Big]\ . \tag{5.19}$$

We now comment each term. Over the whole daily horizon D, the decision maker wants to minimize a discounted sum of all his expenses, that is, the battery renewals and the national grid energy consumption. The first term of the sum over days $\boldsymbol{P}_d^b \times \boldsymbol{R}_d$ is the cost incurred by a battery renewal during day d. The second term $\sum_{m=0}^{M-1} p_{d,m}^e \times \boldsymbol{E}_{d,m+1}^{E\ +}$ is a sum of the national grid energy consumption every minute of the day, where $E_{d,m+1}^E$ is eliminated using Equation (5.7). We take the positive part, denoted by $^+$, assuming that an excessive production of solar energy is wasted. The sum is discounted by a chosen discount factor γ_d. In the sequel, the discount factor γ_d changes once a year to model a discount rate of $\tau = 4.5\%$ every year

$$\gamma_d = \Big(\frac{1}{1+\tau}\Big)^{\lfloor d/365 \rfloor - 1}\ . \tag{5.20}$$

Using Equations (5.3) and (5.8), we obtain that the expectation cost given by Equation (5.19) can be rewritten as

$$\mathbb{E}\Big[\sum_{d=0}^{D} L_d(\boldsymbol{X}_d, \boldsymbol{U}_d, \boldsymbol{W}_d) + K(\boldsymbol{X}_{D+1})\Big] \tag{5.21}$$

$$= \mathbb{E}\Big[\sum_{d=0}^{D} \gamma_d\Big(\boldsymbol{W}_{d,M}^3 \boldsymbol{U}_{d,M} + \sum_{m=0}^{M-1} p_{d,m}^e \times (\boldsymbol{U}_{d,m} + \boldsymbol{W}_{d,m+1}^2 - \boldsymbol{W}_{d,m+1}^1)\Big)\Big]\ ,$$

where the final cost K is null and the intraday cost L_d is given by

$$L_d : \mathbb{X}_d \times \mathbb{U}_{d,0:M} \times \mathbb{W}_{d,0:M} \to (-\infty, +\infty]\ , \tag{5.22}$$

$$(x_d, u_d, w_d) \mapsto \gamma_d\Big(w_{d,M}^3 u_{d,M} + \sum_{m=0}^{M-1} p_{d,m}^e\big(u_{d,m} + w_{d,m+1}^2 - w_{d,m+1}^1\big)\Big)\ .$$

5.3 Two algorithms for two-time scales stochastic optimal control problems

We introduce hereby a generic two-time scales stochastic optimization problem. For the sake of simplicity we assume that the constraints on states and controls, that are not a dynamic or a non anticipativity constraint, are placed in the instantaneous costs L_d using characteristic functions taking the value $+\infty$. Gathering all the above equations, we can state the optimization problem to be solved:

$$V(x) = \min_{\boldsymbol{X}_{0:D+1},\, \boldsymbol{U}_{0:D}} \mathbb{E}\left[\sum_{d=0}^{D} L_d(\boldsymbol{X}_d, \boldsymbol{U}_d, \boldsymbol{W}_d) + K(\boldsymbol{X}_{D+1})\right], \tag{5.23a}$$

$$\text{s.t } \boldsymbol{X}_{d+1} = f_d(\boldsymbol{X}_d, \boldsymbol{U}_d, \boldsymbol{W}_d)\,, \tag{5.23b}$$

$$\boldsymbol{U}_d = (\boldsymbol{U}_{d,0}, \ldots, \boldsymbol{U}_{d,m}, \ldots, \boldsymbol{U}_{d,M})\,, \tag{5.23c}$$

$$\boldsymbol{W}_d = (\boldsymbol{W}_{d,0}, \ldots, \boldsymbol{W}_{d,m}, \ldots, \boldsymbol{W}_{d,M})\,, \tag{5.23d}$$

$$\sigma(\boldsymbol{U}_{d,m}) \subset \sigma\big(\boldsymbol{W}_{d',m'}; (d',m') \leqslant (d,m)\big) \tag{5.23e}$$

$$\boldsymbol{X}_0 = x\,. \tag{5.23f}$$

The daily cost L_d is given by Equation (5.22), the final cost is equal to zero and the state dynamics between days f_d is given by Equation (5.18c). Note that the notation $\boldsymbol{X}_d$ refers to the state random variable at the minute $(d, 0)$ while the notation $\boldsymbol{U}_d$ and $\boldsymbol{W}_d$ refer respectively to random decision and uncertainty vectors containing all decisions and uncertainties of the day d.

As stated in Problem (5.23), the optimization problem is very similar to a classical discrete time stochastic optimal control problem, except for the non anticipativity constraint (5.5) that expresses the fact that the decision vector $\boldsymbol{U}_d = (\boldsymbol{U}_{d,0}, \ldots, \boldsymbol{U}_{d,M})$ at every time step d does not display the same measurability for each component (information grows every minute).

We present in this part two methods to decompose the two-time scales stochastic optimal control problem (5.23). We apply the decomposition schemes to design tractable algorithms to compute suboptimal policies and values for that kind of problems.

5.3.1 Time blocks decomposition

We introduce a daily independence assumption in order to obtain a day by day decomposition of the optimization problem (5.23), that is, a dynamic programming equation between days. We assume that the sequence of random vectors $\{\boldsymbol{W}_d\}_{d=0,\ldots,D}$ is constituted of independent random variables. However, note that we do not assume that each random vector $\boldsymbol{W}_d = (\boldsymbol{W}_{d,0}, \ldots, \boldsymbol{W}_{d,M})$ is itself composed of independent random variables.

Assumption 26. *The sequence $\{\boldsymbol{W}_d\}_{d=0,\ldots,D}$ is a sequence of independent random vectors.*

We introduce a sequence of *slow time scale value functions*, $\{V_d\}_{d\in\{0,\ldots,D+1\}}$, defined by backward induction as follows. At time $D+1$, we set

$$V_{D+1} = K\,, \tag{5.24a}$$

and then for $d \in \{0, \dots, D\}$ we define by backward induction

$$V_d(x) = \min_{\boldsymbol{X}_{d+1}, \boldsymbol{X}_d, \boldsymbol{U}_d} \mathbb{E}\left[L_d(\boldsymbol{X}_d, \boldsymbol{U}_d, \boldsymbol{W}_d) + V_{d+1}(\boldsymbol{X}_{d+1})\right], \tag{5.24b}$$

$$\text{s.t } \boldsymbol{X}_{d+1} = f_d(\boldsymbol{X}_d, \boldsymbol{U}_d, \boldsymbol{W}_d), \tag{5.24c}$$

$$\boldsymbol{U}_d = (\boldsymbol{U}_{d,0}, \dots, \boldsymbol{U}_{d,m}, \dots, \boldsymbol{U}_{d,M}), \tag{5.24d}$$

$$\boldsymbol{W}_d = (\boldsymbol{W}_{d,0}, \dots, \boldsymbol{W}_{d,m}, \dots, \boldsymbol{W}_{d,M}), \tag{5.24e}$$

$$\sigma(\boldsymbol{U}_{d,m}) \subset \sigma(\boldsymbol{X}_d, \boldsymbol{W}_{d,0:m}), \tag{5.24f}$$

$$\boldsymbol{X}_d = x. \tag{5.24g}$$

Let $d \in \{0, \dots, D\}$ be fixed. To each given pair $(x_d, \boldsymbol{X}_{d+1}) \in \mathbb{X}_d \times L^0(\Omega, \mathcal{F}, \mathbb{P}; \mathbb{X}_{d+1})$, we associate an optimization problem denoted by $\mathcal{P}_{(d,=)}\big[x_d, \boldsymbol{X}_{d+1}\big]$ and given by:

$$\mathcal{P}_{(d,=)}\big[x, \boldsymbol{X}\big] \begin{cases} \min_{\boldsymbol{X}_d, \boldsymbol{U}_d} \mathbb{E}\Big[L_d(\boldsymbol{X}_d, \boldsymbol{U}_d, \boldsymbol{W}_d)\Big], & (5.25a) \\ \quad \text{s.t } f_d(\boldsymbol{X}_d, \boldsymbol{U}_d, \boldsymbol{W}_d) = \boldsymbol{X}, & (5.25b) \\ \qquad \boldsymbol{U}_d = (\boldsymbol{U}_{d,0}, \dots, \boldsymbol{U}_{d,m}, \dots, \boldsymbol{U}_{d,M}), & (5.25c) \\ \qquad \boldsymbol{W}_d = (\boldsymbol{W}_{d,0}, \dots, \boldsymbol{W}_{d,m}, \dots, \boldsymbol{W}_{d,M}), & (5.25d) \\ \qquad \sigma(\boldsymbol{U}_{d,m}) \subset \sigma(\boldsymbol{X}_d, \boldsymbol{W}_{d,0:m}), & (5.25e) \\ \qquad \boldsymbol{X}_d = x. & (5.25f) \end{cases}$$

The value of the optimization Problem (5.25) is denoted by $\phi_{(d,=)}\big(x, \boldsymbol{X}\big)$ and we call this optimization problem *the intraday optimization problem* with equality target. Adopting standard conventions, the value function $\phi_{(d,=)}$ will take the value $+\infty$, when Problem (5.25) does not have an admissible solution for a given pair $(x, \boldsymbol{X})$.

Proposition 27. *Under Assumption 26, the value function V solution of optimization problem (5.23) coincides with the value function V_0 given by the Bellman equation (5.24). Moreover, the sequence of value functions given by Equation (5.24) coincides with the sequence of mappings given by the following backward induction:*

$$V_{D+1} = K \tag{5.26a}$$

$$\forall x \in \mathbb{X}_d, \quad V_d(x) = \min_{\boldsymbol{X} \in L^0(\Omega, \mathcal{F}, \mathbb{P}; \mathbb{X}_{d+1})} \Big(\phi_{(d,=)}\big(x, \boldsymbol{X}\big) + \mathbb{E}\big[V_{d+1}(\boldsymbol{X})\big]\Big), \tag{5.26b}$$

$$s.t\ \sigma(\boldsymbol{X}) \subset \sigma(\boldsymbol{W}_d). \tag{5.26c}$$

Proof. Under Assumption 26, the optimal value of Problem (5.23) remains unchanged when the non anticipativity constraint (5.23e) is replaced by the constraint:

$$\sigma(\boldsymbol{U}_{d,m}) \subset \sigma(\boldsymbol{X}_d, \boldsymbol{W}_{d,0:m}). \tag{5.27}$$

Then, the fact that the backward induction (5.24) is the Bellman equation which gives the solution of Problem (5.23) is detailed in [120] (Chapter 1 of this thesis). Exploiting the linearity of the expectation and the fact that minimization can be done sequentially,

we rewrite Equation (5.24) as

$$\begin{aligned} V_d(x) &= \min_{\boldsymbol{X}_{d+1}\in L^0(\Omega,\mathcal{F},\mathbb{P};\mathbb{X}_{d+1})} \min_{\boldsymbol{U}_d,\boldsymbol{X}_d} \mathbb{E}\big[L_d(\boldsymbol{X}_d,\boldsymbol{U}_d,\boldsymbol{W}_d)\big] + \mathbb{E}\big[V_{d+1}(\boldsymbol{X}_{d+1})\big] \,, \\ &\qquad\qquad \text{s.t } (5.24c)\text{-}(5.24g) \,, \\ &= \min_{\boldsymbol{X}_{d+1}\in L^0(\Omega,\mathcal{F},\mathbb{P};\mathbb{X}_{d+1})} \Big(\phi_{(d,=)}\big(x,\boldsymbol{X}_{d+1}\big) + \mathbb{E}\big[V_{d+1}(\boldsymbol{X}_{d+1})\big]\Big) \,, \end{aligned} \tag{5.28}$$

Moreover, if $\boldsymbol{X}_{d+1} \in \operatorname{dom}\big(\phi_{(d,=)}\big(x,\cdot\big)\big)$, then $\boldsymbol{X}_{d+1}$ is given by Equation (5.24c) and thus it is a $\sigma(\boldsymbol{W}_d)$-measurable random variable. Therefore, adding Constraint (5.26c) in the optimization problem (5.28) yields the same optimization problem. This ends the proof. □

5.3.2 Stochastic targets decomposition algorithm

The numerical resolution of intraday problem (5.25) is most of the time out of reach due to the target constraint (5.25b). In order to compute approximations of the daily value functions (5.24), we present simplified versions of Problem (5.25). We introduce a relaxation of the target constraint (5.25b), turning the equality into an inequality. Furthermore that makes possible to look for deterministic targets instead of stochastic ones which simplifies the information constraint (5.26c). We apply the general results in § 5.5.1 to the slow scale Bellman equation (5.26).

Relaxed intraday optimization problem

For each $d \in \{0,\dots,D\}$, we introduce a relaxed intraday optimization problem, $\mathcal{P}_{(d,\geqslant)}\big[x,\boldsymbol{X}\big]$, which is obtained by considering the optimization problem (5.25) whith the equality target (5.25b) replaced by the following inequality target (5.29b):

$$\mathcal{P}_{(d,\geqslant)}\big[x,\boldsymbol{X}\big] \begin{cases} \min\limits_{\boldsymbol{X}_d,\boldsymbol{U}_d} \mathbb{E}\Big[L_d(\boldsymbol{X}_d,\boldsymbol{U}_d,\boldsymbol{W}_d)\Big] \,, & (5.29a) \\ \quad \text{s.t } f_d(\boldsymbol{X}_d,\boldsymbol{U}_d,\boldsymbol{W}_d) \geqslant \boldsymbol{X} \,, & (5.29b) \\ \qquad \boldsymbol{U}_d = (\boldsymbol{U}_{d,0},\dots,\boldsymbol{U}_{d,m},\dots,\boldsymbol{U}_{d,M}) \,, & (5.29c) \\ \qquad \boldsymbol{W}_d = (\boldsymbol{W}_{d,0},\dots,\boldsymbol{W}_{d,m},\dots,\boldsymbol{W}_{d,M}) \,, & (5.29d) \\ \qquad \sigma(\boldsymbol{U}_{d,m}) \subset \sigma(\boldsymbol{X}_d,\boldsymbol{W}_{d,0:m}) \,, & (5.29e) \\ \qquad \boldsymbol{X}_d = x \,. & (5.29f) \end{cases}$$

We denote by $\phi_{(d,\geqslant)}(x,\boldsymbol{X})$ the value of the relaxed optimization problem $\mathcal{P}_{(d,\geqslant)}\big[x,\boldsymbol{X}\big]$. We associate to the relaxed value function $\phi_{(d,\geqslant)}\big(x,\boldsymbol{X}\big)$ a sequence of relaxed Bellman value functions $\big\{V_{(d,\geqslant)}\big\}_{d\in\{0,\dots,D+1\}}$ defined as follows:

$$V_{(D+1,\geqslant)} = K \,, \tag{5.30a}$$

and for all $d \in \{0,\dots,D\}$, and for all $x \in \mathbb{X}_d$

$$V_{(d,\geqslant)}(x) = \min_{\boldsymbol{X}\in L^0(\Omega,\mathcal{F},\mathbb{P};\mathbb{X}_{d+1})} \Big(\phi_{(d,\geqslant)}\big(x,\boldsymbol{X}\big) + \mathbb{E}\big[V_{(d+1,\geqslant)}(\boldsymbol{X})\big]\Big) \,, \tag{5.30b}$$

$$\text{s.t } \sigma(\boldsymbol{X}) \subset \sigma(\boldsymbol{W}_d) \,. \tag{5.30c}$$

We then consider the case where, in Equation (5.30), the minimization over the space $L^0(\Omega, \mathcal{F}, \mathbb{P}; \mathbb{X}_{d+1})$ is replaced by minimization over the constants $x \in \mathbb{X}_{d+1}$; we denote by $\{V_{(d,\geqslant,\mathbb{X}_{d+1})}\}_{d\in\{0,\dots,D+1\}}$ the associated sequence of Bellman functions.

$$V_{(D+1,\geqslant,\mathbb{X}_{D+2})} = K \ , \tag{5.31a}$$

and for all $d \in \{0, \dots, D\}$, and for all $x \in \mathbb{X}_d$

$$V_{(d,\geqslant,\mathbb{X}_{d+1})}(x) = \min_{X\in\mathbb{X}_{d+1}} \Big(\phi_{(d,\geqslant)}\big(x, X\big) + V_{(d+1,\geqslant,\mathbb{X}_{d+2})}(X)\Big) \ . \tag{5.31b}$$

The undefined state space $\mathbb{X}_{D+2}$ in (5.31a) is introduced for consistency with recursive equation (5.31b). It can be any space as it is not used in Equation (5.31a).

Assumption 28. *The value functions $\{V_d\}_{d=0,\dots,D}$ are non-increasing.*

We show in Proposition 29, that under Assumption 28, the value functions $V_{(d,\geqslant,\mathbb{X}_{d+1})}$ give upper bounds to the original value functions V_d in (5.28).

Proposition 29. *The sequence of relaxed Bellman value functions $\{V_{(d,\geqslant)}\}_{d\in\{0,\dots,D+1\}}$ given by Equation (5.30) gives lower bound to the sequence of value functions $\{V_d\}_{d\in\{0,\dots,D+1\}}$ given by Equation (5.24). That is, for all $d \in \{0, \dots, D+1\}$, we have*

$$V_{(d,\geqslant)} \leqslant V_d \ . \tag{5.32}$$

Moreover, under Assumption 28 we have for all $d \in \{0, \dots, D\}$ that

$$V_d = V_{(d,\geqslant)} \leqslant V_{(d,\geqslant,\mathbb{X}_{d+1})} \ . \tag{5.33}$$

Proof. Let $d \in \{0, \dots, D-1\}$ and a pair $(x_d, \boldsymbol{X}_{d+1}) \in \mathbb{X}_d \times L^0(\Omega, \mathcal{F}, \mathbb{P}; \mathbb{X}_{d+1})$ given. We have that

$$\phi_{(d,\geqslant)}\big(x_d, \boldsymbol{X}_{d+1}\big) \leqslant \phi_{(d,=)}\big(x_d, \boldsymbol{X}_{d+1}\big) \ . \tag{5.34}$$

From Equations (5.28) and (5.30), we obtain by backward induction that for all $d \in \{0, \dots, D+1\}$

$$V_{(d,\geqslant)} \leqslant V_d \ . \tag{5.35}$$

Now, given $d \in \{0, \dots, D-1\}$, since the set of constant random variables taking values in $\mathbb{X}_{d+1}$ is a subset of $L^0(\Omega, \mathcal{F}, \mathbb{P}; \mathbb{X}_{d+1})$ we obtain that $V_{(d,\geqslant)} \leqslant V_{(d,\geqslant,\mathbb{X}_{d+1})}$. Thus, the only point to prove is that under Assumption 28 we have the equality $V_d = V_{(d,\geqslant)}$. We proceed by backward induction. At time $D+1$, the two mappings $V_{(D+1,\geqslant)}$ and V_{D+1} are both equal to K which is non-increasing. Then, let d be fixed in $\{0, \dots, D\}$ and assume that $V_{(d+1,\geqslant)} = V_{d+1}$ and that these two value functions are non-increasing. We prove that $V_{(d,\geqslant)}$ and V_d coincides using Lemma 44 which applies since $\mathbb{X}_{d+1}$ is a subset of some finite dimensional space $\mathbb{R}^{n_x}$. □

Remark 30. *Looking for deterministic targets instead of stochastic targets is made possible by the fact that we relaxed the almost sure target equality constraint (5.29b) into an inequality using the value functions monotonicity. An almost sure equality constraint requires both sides to have the same measurability, we would have to ensure that a random variable is always equal to a deterministic one which is most of the time impossible.*

Statement of the algorithm with deterministic targets and periodicity classes

In order to compute the daily value functions upper bounds $\{V_{(d,\geqslant,\mathbb{X}_{d+1})}\}_{d=0,\dots,D+1}$ via Equation (5.25), we need the value of the relaxed intraday problems $\phi_{(d,\geqslant)}\big(x_d, x_{d+1}\big)$ for all $d \in \{0,\dots,D+1\}$ and for all pairs $((x_d, x_{d+1}) \in \mathbb{X}_d \times \mathbb{X}_{d+1}$, where we recall that $\phi_{(d,\geqslant)}\big(x_d, x_{d+1}\big)$ is given by

$$\phi_{(d,\geqslant)}\big(x_d, x_{d+1}\big) = \min_{\boldsymbol{U}_d} \mathbb{E}\Big[L_d(x_d, \boldsymbol{U}_d, \boldsymbol{W}_d) \Big] , \tag{5.36a}$$

$$\text{s.t } f_d(x_d, \boldsymbol{U}_d, \boldsymbol{W}_d) \geqslant x_{d+1} , \tag{5.36b}$$

$$\sigma(\boldsymbol{U}_{d,m}) \subset \sigma(\boldsymbol{W}_{d,0:m}) . \tag{5.36c}$$

The computational cost can be significant as we need to solve a stochastic optimization problem for every pair $(x_d, x_{d+1}) \in \mathbb{X}_d \times \mathbb{X}_{d+1}$ and for every d in $\{0,\dots,D\}$. We present a simplification exploiting periodicity of the intraproblems.

Lemma 31. *Let $\mathbb{I} \subset \{0,\dots,D\}$. Assume that there exists two sets $\mathbb{X}_\mathbb{I}$ and $\mathbb{U}_\mathbb{I}$ such that for all $d \in \mathbb{I}$, $\mathbb{X}_d = \mathbb{X}_\mathbb{I}$ and $\mathbb{U}_d = \mathbb{U}_\mathbb{I}$. Assume moreover than there exists two mappings $L_\mathbb{I}$ and $f_\mathbb{I}$ such that for all $d \in \mathbb{I}$, $L_d = L_\mathbb{I}$ and $f_d = f_\mathbb{I}$. Finally assume that the random variables $\{\boldsymbol{W}_d\}_{d\in\mathbb{I}}$ are independent and identically distributed. Then, there exists a function $\phi_\mathbb{I}$ such that for all $d \in \mathbb{I}$*

$$\phi_d = \phi_\mathbb{I} . \tag{5.37}$$

Proof. The proof is immediate. □

The set $\mathbb{I}$ introduced in Lemma 31 is called a periodicity class. We call N_p the number of periodicity classes of Problem (5.23) and $(\mathbb{I}_1,\dots,\mathbb{I}_{N_p})$ the periodicity classes, that is, the sets of day indices that satisfy (5.37).

Remark 32. *When there is no periodicity, $N_p = D+1$ and the periodicity classes are singletons. In this case all the intraday problems have to be computed.*

Remark 33. *A periodicity property often appears in long term energy management problems with renewable energies, due to seasonality of natural processes such as solar production. In these cases $N_p < D+1$ and it is enough to solve only N_p intraproblems.*

The algorithm to compute daily value functions approximations, with relaxed intraday

problems, deterministic targets and periodicity classes, is presented in Algorithm 3.

Algorithm 3: Two-time scales dynamic programming with deterministic targets and periodicity classes

Data: Periodicity classes $(\mathbb{I}_1, \ldots, \mathbb{I}_{N_p})$
Result: Daily value functions approximations $(V_{(d,\geqslant,\mathbb{X}_{d+1})})_{d=0,\ldots,D+1}$
Initialization: $V_{(D+1,\geqslant,\mathbb{X}_{D+2})} = K$;
for $i = 1, \ldots, N_p$ **do**
  Let $d \in \mathbb{I}_i$;
  for $(x_d, x_{d+1}) \in \mathbb{X}_d \times \mathbb{X}_{d+1}$ **do**
    Compute $\phi_{(d,\geqslant)}(x_d, x_{d+1})$;
  end
end
for $d = D, D-1 \ldots, 0$ **do**
  for $x_d \in \mathbb{X}_d$ **do**
    Solve $V_{(d,\geqslant,\mathbb{X}_{d+1})}(x_d) = \min_{x_{d+1} \in \mathbb{X}_{d+1}} \ \phi_{(d,\geqslant)}(x_d, x_{d+1}) + V_{(d+1,\geqslant,\mathbb{X}_{d+2})}(x_{d+1})$;
  end
end

Two further simplifications for the intraday problems computation

Two particular properties can be exploited to lower further the computational burden of the sequence of intraday problems $\phi_{(d,\geqslant)}$.

Initial state, final target pair dimension reduction. We introduce a first particular property of some problems allowing to lower the computational burden of the intraproblems by reducing the dimensionality of the initial state/target pair.

Assumption 34.

1. $\mathbb{X}_d = \mathbb{X}_{d+1}$,
2. $L_d(x_d, \boldsymbol{U}_d, \boldsymbol{W}_d) = l_d(\boldsymbol{U}_d, \boldsymbol{W}_d)$,
3. $f_d(x_d, \boldsymbol{U}_d, \boldsymbol{W}_d) = x_d + g_d(\boldsymbol{U}_d, \boldsymbol{W}_d)$.

Under Assumption 34, it is enough to solve, the optimization problem for all $x \in \mathbb{X}_d - \mathbb{X}_{d+1}$ (instead of each $(x_d, x_{d+1}) \in \mathbb{X}_d \times \mathbb{X}_{d+1}$)

$$\overline{\phi}_{(d,\geqslant)}(x) = \min_{\boldsymbol{U}_d} \mathbb{E}\Big[L_d(x, \boldsymbol{U}_d, \boldsymbol{W}_d)\Big] \,, \tag{5.38a}$$

$$\text{s.t } f_d(x, \boldsymbol{U}_d, \boldsymbol{W}_d) \geqslant 0 \,, \tag{5.38b}$$

$$\sigma(\boldsymbol{U}_{d,m}) \subset \sigma(\boldsymbol{W}_{d,0:m}) \,. \tag{5.38c}$$

Convexity and stagewise independence assumption. When the state dynamics f_d are linear and costs L_d and K are convex in (x, u), we can use Stochastic Dual Dynamic Programming (SDDP) to solve the intraday problems, assuming stagewise independence of the intraday noises $(\boldsymbol{W}_{d,0}, \ldots, \boldsymbol{W}_{d,M})$. We obtain a convex polyhedral lower approximation of $\phi_{(d,\geqslant)}$. This convex polyhedral lower approximation

can be represented by a linear program hence it makes it possible to compute a piecewise linear lower approximation $\underline{V}_{(d,\geqslant,\mathbb{X}_{d+1})}$ of the daily value functions upper bounds $V_{(d,\geqslant,\mathbb{X}_{d+1})}$ using Linear Programming (LP).

Remark 35. *We are not guaranteed that* $V_{(d,\geqslant)} \leqslant \underline{V}_{(d,\geqslant,\mathbb{X}_{d+1})}$.

Now we can compute efficiently intraday problems and lower bounds for the daily value functions using a deterministic target decomposition. We present another method to compute value functions for a two-time scales stochastic optimization problem relying on a deterministic weights decomposition.

5.3.3 Stochastic adaptative weights algorithm

In this part we investigate an algorithm based on applying Fenchel-Rockafellar duality [141, 140] to the dynamic programming equation with targets (5.26), in particular to the target constraint (5.25b). This method is connected to the one developed in [128] called "adaptative weights", hence the name " Stochastic Adaptative Weights" (SAWA). We extend their results in a stochastic setting and a more general framework as we are not tied to a battery management problem. Furthermore we use well known duality results to reach similar conclusions.

We introduce the dualized intraday problems, whose value is called ψ_d for $d \in \{0, \dots, D-1\}$, such that for all $(x_d, \boldsymbol{\lambda}_{d+1}) \in \mathbb{X}_d \times L^q(\Omega, \mathcal{F}, \mathbb{P}; \Lambda_{d+1})$, where Λ_{d+1} is the dual space of $\mathbb{X}_{d+1}$ ($\Lambda_{d+1} = \mathbb{R}^{n_x}$ if $\mathbb{X}_{d+1} = \mathbb{R}^{n_x}$):

$$\psi_d(x_d, \boldsymbol{\lambda}_{d+1}) = \min_{\boldsymbol{X}_d, \boldsymbol{U}_d} \mathbb{E}\left[L_d(\boldsymbol{X}_d, \boldsymbol{U}_d, \boldsymbol{W}_d) + \langle \boldsymbol{\lambda}_{d+1}, f_d(\boldsymbol{X}_d, \boldsymbol{U}_d, \boldsymbol{W}_d)\rangle\right], \tag{5.39a}$$

$$\text{s.t } \boldsymbol{U}_d = (\boldsymbol{U}_{d,0}, \dots, \boldsymbol{U}_{d,m}, \dots, \boldsymbol{U}_{d,M}), \tag{5.39b}$$

$$\boldsymbol{W}_d = (\boldsymbol{W}_{d,0}, \dots, \boldsymbol{W}_{d,m}, \dots, \boldsymbol{W}_{d,M}), \tag{5.39c}$$

$$\sigma(\boldsymbol{U}_{d,m}) \subset \sigma(\boldsymbol{X}_d, \boldsymbol{W}_{d,0:m}), \tag{5.39d}$$

$$\boldsymbol{X}_d = x_d . \tag{5.39e}$$

We assume that for all $d \in \{0, \dots, D\}$, for any state $x_d \in \mathbb{X}_d$, control $\boldsymbol{U}_d$ and uncertainty $\boldsymbol{W}_d$, the random variable $f_d(x_d, \boldsymbol{U}_d, \boldsymbol{W}_d)$ belongs to $L^p(\Omega, \mathcal{F}, \mathbb{P}; \mathbb{X}_{d+1})$ with $1 < p < +\infty$ and $\frac{1}{p} + \frac{1}{q} = 1$.

For any state x_d, admissible control $\boldsymbol{U}_d$ and uncertainty $\boldsymbol{W}_d$, $f(x_d, \boldsymbol{U}_d, \boldsymbol{W}_d)$ is measurable with respect to $\sigma(\boldsymbol{W}_d)$ due to the non anticipativity constraint (5.39d). Hence for any random variable $\boldsymbol{\lambda}_{d+1} \in L^q(\Omega, \mathcal{F}, \mathbb{P}; \Lambda_{d+1})$ we have the following equality that make it possible to restrict the measurability of dual variables [118, Chap. 5.5]:

$$\psi_d(x_d, \boldsymbol{\lambda}_{d+1}) = \psi_d\Big(x_d, \mathbb{E}\Big[\boldsymbol{\lambda}_{d+1} | \sigma(\boldsymbol{W}_d)\Big]\Big) . \tag{5.40}$$

Then, we introduce the following daily value functions,

$$\underline{V}_{D+1} = K , \tag{5.41a}$$

and, for all $d \in \{0, \dots, D\}$, and for all $x_d \in \mathbb{X}_d$,

$$\underline{V}_d(x_d) = \sup_{\boldsymbol{\lambda}_{d+1} \in \Lambda_{d+1}} \psi_d(x_d, \boldsymbol{\lambda}_{d+1}) - \mathbb{E}\Big[\underline{V}^\star_{d+1}(\boldsymbol{\lambda}_{d+1})\Big] \,, \tag{5.41b}$$

$$\text{s.t } \sigma(\boldsymbol{\lambda}_{d+1}) \subset \sigma(\boldsymbol{W}_d) \,, \tag{5.41c}$$

where $\underline{V}^\star_{d+1}$ is the Fenchel transform of the function $\underline{V}_{d+1}$. We prove, in the next proposition, that the value function $\underline{V}_d$ gives a lower bound to the value function V_d.

Lemma 36. *For every $d \in \{0, \dots, D\}$,*

$$\underline{V}_d \leqslant V_d \,. \tag{5.42}$$

Proof. We apply Lemma 47 to $\phi_{(d,=)}(x_d, \cdot)$ and $\mathbb{E}\Big[V_{d+1}\Big]$. □

Proposition 37. *Assume that K is convex and that for $d \in \{0, \dots, D\}$ the instantaneous costs L_d are jointly convex in x and u and that the dynamics f_d are jointly linear in x and u. If moreover* $\mathrm{ri}\Big(\mathrm{dom}\big(\phi_{(d,=)}(x_d, \cdot)\big) - \mathrm{dom}\big(\mathbb{E}\big[V_{d+1}\big]\big)\Big) \neq \varnothing$*, then we have the equality*

$$V_d = \underline{V}_d \,. \tag{5.43}$$

Proof. Under the convexity assumptions we are ensured that for all d, $\phi_{(d,=)}$ and V_d are convex. We apply Proposition 48 to $\phi_{d,=}(x_d, \cdot)$ and $\mathbb{E}\Big[V_{d+1}\Big]$ to obtain that, for all $d \in \{0, \dots, D\}$, and for all $x_d \in \mathbb{X}_d$:

$$\underline{V}_d(x_d) = \sup_{\boldsymbol{\lambda}_{d+1}} \psi_d(x_d, \boldsymbol{\lambda}_{d+1}) - \Big(\mathbb{E}\Big[\underline{V}_{d+1}\Big]\Big)^\star(\boldsymbol{\lambda}_{d+1}) \,, \tag{5.44a}$$

where

$$\Big(\mathbb{E}\big[\underline{V}_{d+1}\big]\Big)^\star(\boldsymbol{\lambda}_{d+1}) = \sup_{\boldsymbol{X} \in L^p(\Omega, \mathcal{F}, \mathbb{P}; \mathbb{X}_{d+1})} \langle \boldsymbol{\lambda}_{d+1}, \boldsymbol{X} \rangle - \mathbb{E}\big[\underline{V}_{d+1}\big](\boldsymbol{X}) \,, \tag{5.45}$$

$$\text{s.t } \sigma(\boldsymbol{X}) \subset \sigma(\boldsymbol{W}_d) \,. \tag{5.46}$$

Due to the property (5.40), this is equivalent to

$$\underline{V}_d(x_d) = \sup_{\boldsymbol{\lambda}_{d+1}} \psi_d(x_d, \boldsymbol{\lambda}_{d+1}) - \Big(\mathbb{E}\Big[\underline{V}_{d+1}\Big]\Big)^\star(\boldsymbol{\lambda}_{d+1}) \,, \tag{5.47a}$$

$$\text{s.t } \sigma(\boldsymbol{\lambda}_{d+1}) \subset \sigma(\boldsymbol{W}_d) \,. \tag{5.47b}$$

Finally we need to invert Fenchel transform and expectation in (5.47a) to obtain (5.41). The proof of [122, Prop. 12], using [139] and [138], can be applied straightforwardly. □

Deterministic weights simplification

It is computationally costly to compute the function ψ_d in (5.39) for every $d \in \{0, \dots, D\}$, initial state $x_d \in \mathbb{X}_d$ and stochastic weights $\boldsymbol{\lambda} \in L^q(\Omega, \mathcal{F}, \mathbb{P}; \Lambda_{d+1})$. As in §5.3.2 we relax the equality target constraint (5.25b) and restrict the computation to deterministic weights

in Λ_{d+1} which corresponds to dualize an expectation target constraint as detailed in the sequel.

We build by backward induction deterministic weights value functions:

$$V_{(D+1,\geqslant,\mathbb{E})} = K \ , \tag{5.48a}$$

and for all $d \in \{0, \dots, D\}$, and for all $x_d \in \mathbb{X}_d$,

$$V_{(d,\geqslant,\mathbb{E})}(x_d) = \sup_{\lambda_{d+1} \in \Lambda_{d+1}} \psi_d(x_d, \lambda_{d+1}) - V^{\star}_{(d+1,\geqslant,\mathbb{E})}(\lambda_{d+1}) \ , \tag{5.48b}$$

$$\lambda_{d+1} \leqslant 0 \ . \tag{5.48c}$$

Relaxing the target equality constraint (5.25b) into an inequality makes it possible to constrain to non positive weights in (5.48c).

Remark 38. *The notation $V_{(d,\geqslant,\mathbb{E})}$ with the expectation in index comes from the connection with the dualization of the target constraint in the optimization problem $\mathcal{P}_{(d,\geqslant)}\big[x_d, \boldsymbol{X}_{d+1}\big]$ where the almost sure inequality target constraint is replaced by a constraint in expectation, see (5.49b). We denote this new optimization problem by $\mathcal{P}_{(d,\geqslant,\mathbb{E})}\big[x_d, \boldsymbol{X}_{d+1}\big]$:*

$$\mathcal{P}_{(d,\geqslant,\mathbb{E})}\big[x_d, \boldsymbol{X}_{d+1}\big] \begin{cases} \min\limits_{\boldsymbol{U}_d} \quad \mathbb{E}\Big[L_d(x_d, \boldsymbol{U}_d, \boldsymbol{W}_d)\Big] \ , & \text{(5.49a)} \\ \quad s.t\ \mathbb{E}\Big[f_d(x_d, \boldsymbol{U}_d, \boldsymbol{W}_d) - \boldsymbol{X}_{d+1}\Big] \geqslant 0 \ , & \text{(5.49b)} \\ \quad \boldsymbol{U}_d = (\boldsymbol{U}_{d,0}, \dots, \boldsymbol{U}_{d,m}, \dots, \boldsymbol{U}_{d,M}) \ , & \text{(5.49c)} \\ \quad \boldsymbol{W}_d = (\boldsymbol{W}_{d,0}, \dots, \boldsymbol{W}_{d,m}, \dots, \boldsymbol{W}_{d,M}) \ , & \text{(5.49d)} \\ \quad \sigma(\boldsymbol{U}_{d,m}) \subset \sigma(\boldsymbol{W}_{d,0:m}) \ , & \text{(5.49e)} \end{cases}$$

That kind of simplification is also applied in [121].

$V_{(d,\geqslant,\mathbb{E})}$ is the value of a more constrained maximization problem with unchanged objective than $\underline{V}_d$, due to the restriction to deterministic weights. Hence we have the following inequality

$$V_{(d,\geqslant,\mathbb{E})} \leqslant \underline{V}_d \leqslant V_d \ . \tag{5.50}$$

Statement of the SAWA algorithm with deterministic weights and periodicity classes

To summarize, the algorithm to compute daily value functions approximations for relaxed intraday problems with deterministic weights is presented in Algorithm 4, once again with

periodicity classes as introduced in Lemma 31.

Algorithm 4: Two-time scales dynamic programming with weights

Data: Periodicity classes $(\mathbb{I}_1, \dots, \mathbb{I}_{N_p})$
Result: Daily value functions approximations $(V_{(d,\geqslant,\mathbb{E})})_{d=0,\dots,D+1}$
Initialization: $V_{(D+1,\geqslant,\mathbb{E})} = K$;
for $i = 1, \dots, N_p$ **do**
  Let $d \in \mathbb{I}_i$;
  for $(x_d, \lambda_{d+1}) \in \mathbb{X}_d \times \Lambda_{d+1}$ **do**
    Compute $\psi_d(x_d, \lambda_{d+1})$;
  end
end
for $d = D, D-1 \dots, 0$ **do**
  for $x_d \in \mathbb{X}_d$ **do**
    Solve $V_{(d,\geqslant,\mathbb{E})}(x_d) = \sup_{\lambda_{d+1} \in \Lambda_{d+1}} \ \psi_d(x_d, \lambda_{d+1}) - V^{\star}_{(d+1,\geqslant,\mathbb{E})}(\lambda_{d+1})$;
  end
end

Remark 39. *The whole interest of the two algorithms 3 and 4 to compute daily value functions approximations is that we can solve intraday problems in parallel, or distribute the resolution of the intraday problems across days. Moreover, we can theoretically apply any stochastic optimization method to solve the intraday problems. Without stagewise independence assumption we may use Stochastic Programming techniques (for example scenario trees) to solve the intraday problems. With the stagewise assumption we may apply Stochastic Dynamic Programming.*

5.3.4 Producing an online policy using the daily value functions

We assume that we have at disposal daily value functions $\{\widetilde{V}_d\}_{d=0,\dots,D}$ either obtained by the targets algorithm ($\widetilde{V}_d = V_{(d,\geqslant,\mathbb{X}_{d+1})}$), or by the weights algorithm ($\widetilde{V}_d = V_{(d,\geqslant,\mathbb{E})}$).

Proposition 40. *Under Assumption 28 and Proposition (29), we have the following inequality*

$$V_{(d,\geqslant,\mathbb{E})} \leqslant V_d \leqslant V_{(d,\geqslant,\mathbb{X}_{d+1})} \ . \tag{5.51}$$

Proof. It is a reminder of Equation (5.50) and Proposition 29. □

Now, for each given day $d \in \{0, \dots, D\}$ and given a current state $x_d \in \mathbb{X}_d$, we can use the daily value functions $\widetilde{V}_d$ as daily value functions approximation in order to state a new intraday problem on day d as follows:

$$\min_{\boldsymbol{X}_d, \boldsymbol{U}_d} \mathbb{E}\Big[L_d(\boldsymbol{X}_d, \boldsymbol{U}_d, \boldsymbol{W}_d) + \widetilde{V}_d\Big(f_d(\boldsymbol{X}_d, \boldsymbol{U}_d, \boldsymbol{W}_d) \Big) \Big] \ , \tag{5.52a}$$

$$\text{s.t } \boldsymbol{U}_d = (\boldsymbol{U}_{d,0}, \dots, \boldsymbol{U}_{d,m}, \dots, \boldsymbol{U}_{d,M}) \ , \tag{5.52b}$$

$$\boldsymbol{W}_d = (\boldsymbol{W}_{d,0}, \dots, \boldsymbol{W}_{d,m}, \dots, \boldsymbol{W}_{d,M}) \ , \tag{5.52c}$$

$$\sigma(\boldsymbol{U}_{d,m}) \subset \sigma(\boldsymbol{X}_d, \boldsymbol{W}_{d,0:m}) \ , \tag{5.52d}$$

$$\boldsymbol{X}_d = x_d \ . \tag{5.52e}$$

This problem can be solved by any method that provides an online policy as presented in [119] (see Chapter 2). The presence of a final cost $\widetilde{V}_d$ ensures that the long term effect on battery health of decisions made every minute is taken into account inside the intraday problem policy.

The simulation of an online policy for a stochastic optimization problem is often made offline as part of the verification process of a stochastic optimal control problem resolution. Here it would be time consuming to produce online policies using the resolution of problem (5.52) for every day of the horizon in simulation. These policies are more relevant for the real control of the system, hence the resolution of problem (5.52) can be distributed across days. We present in the next two paragraphs how to simulate two-time scales policies with targets or weights in a smaller amount of time.

Simulating a policy using targets

In the case we decomposed the problem using deterministic targets, we eventually solved intraday problems whose values are $\{\phi_{(d,\geqslant)}\}_{d=0,\dots,D}$ for every couple of initial state and deterministic target $(x_d, x_{d+1}) \in \mathbb{X}_d \times \mathbb{X}_{d+1}$. In the process, a policy for every intraday problem has been computed. For $d \in \{0,\dots,D\}$ and all $(x_d, x_{d+1}) \in \mathbb{X}_d \times \mathbb{X}_{d+1}$, we call $\pi^t_{(d,\geqslant)}(x_d, x_{d+1}) : \mathbb{W}_d \to \mathbb{U}_{d,0}\times,\dots,\mathbb{U}_{d,M}$ a policy solving $\mathcal{P}_{(d,\geqslant)}\Big[x_d, x_{d+1}\Big]$ whose value is $\phi_{(d,\geqslant)}(x_d, x_{d+1})$.

We computed the value $\phi_{(d,\geqslant)}\big(x_d, x_{d+1}\big)$ only on $\mathbb{X}_d \times \mathbb{X}_{d+1}$ as we replaced stochastic targets by deterministic ones. Therefore, we can only compute a target decision solving the problem

$$x^t_{d+1} \in \underset{x\in\mathbb{X}_{d+1}}{\arg\min} \Big(\phi_{(d,\geqslant)}\big(x_d, x\big) + V_{(d+1,\geqslant,\mathbb{X}_{d+1})}(x)\Big)\,. \tag{5.53a}$$

A deterministic target x^t_{d+1} is computed and we apply the corresponding intraday policy $\pi^t_{(d,\geqslant)}(x_d, x^t_{d+1})$ to simulate intraday decisions and states drawing a scenario w_d out of $\boldsymbol{W}_d$. The next state x_{d+1} at the beginning of day $d{+}1$ is then $x_{d+1} = f_d\Big(x_d, \pi^t_{(d,\geqslant)}(x_d, x^t_{d+1})(w_d), w_d\Big)$.

Simulating a policy using weights

In the case we decomposed the problem using deterministic weights, we eventually solved relaxed intraday problems whose values are $\{\psi_d\}_{d=0,\dots,D}$ for every couple of initial state and deterministic weight $(x_d, \lambda_{d+1}) \in \mathbb{X}_d \times \Lambda_{d+1}$. In the process, a policy for every relaxed intraday problems has been computed. For $d \in \{0,\dots,D\}$ and all $(x_d, \lambda_{d+1}) \in \mathbb{X}_d \times \Lambda_{d+1}$, we call $\pi^w_d(x_d, \lambda_{d+1}) : \mathbb{W}_d \to \mathbb{U}_{d,0}\times,\dots,\mathbb{U}_{d,M}$ a policy solving the problem whose value is $\psi_d(x_d, \lambda_{d+1})$.

At the beginning of day d in a state $x_d \in \mathbb{X}_d$, we compute a weight $\lambda^w_{d+1} \in \Lambda_{d+1}$ solving the following optimization problem

$$\lambda^w_{d+1} \in \underset{\lambda\in\Lambda_{d+1}}{\operatorname{argmax}} \Big(\psi_d(x_d, \lambda) - V^\star_{(d+1,\geqslant,\mathbb{E})}(\lambda)\Big)\,. \tag{5.54a}$$

Thanks to this deterministic weight λ^w_{d+1}, we apply the corresponding intraday policy $\pi^w_d(x_d, \lambda^w_{d+1})$ to simulate intraday decisions and states drawing a scenario w_d out of $\boldsymbol{W}_d$. The next state x_{d+1} at the beginning of day $d{+}1$ is then $x_{d+1} = f_d\Big(x_d, \pi^w_d(x_d, \lambda^w_{d+1})(w_d), w_d\Big)$.

In the next section, we present numerical experiments using the targets and weights algorithms.

5.4 Numerical experiments

In this section, we apply the previous theoretical results to three long term battery management problems. First, we describe the realistic data used for the three experiments. Then, we present the first experiment that consists in solving the battery charge/discharge, aging and renewal management problem introduced in Section 5.2, with targets decomposition. We compare the results with a daily management approach that ignores aging. Then, we present a battery aging management without renewal, with fixed capacity, over a few days. It makes it possible to apply targets and weights decomposition algorithms and to compare them to a straightforward application of Stochastic Dynamic Programming and Stochastic Dual Dynamic Programming over the whole horizon. Finally, we apply targets decomposition with SDDP for the sizing of a battery taking aging into account.

5.4.1 Experimental setup

We use a realistic instance of the problem: a house with solar panels and battery. The problem presents three sources of randomness, namely, solar panels production, electrical demand and prices of batteries per kWh.

Data to model demand and production

We assume that the house is equipped with 12 kW of solar panels. One year scenarios of solar exposure in Zambia with a time step of 1 minute are openly available[1]. Using these solar scenarios, we can generate realistic solar panels production scenarios using Python library PVlib[2]. We display in Figure 5.3 the distribution of solar panels production every hour as a boxplot.

For the demand data we obtained openly available scenarios from Ausgrid[3]. This is electrical demand data from a customer in kWh with 1 minute time step as well. We display in Figure 5.4 the hourly distribution of electrical demand.

[1] energydata.info/en/dataset/zambia-solar-radiation-measurement-data-2015-2017
[2] github.com/pvlib/pvlib-python
[3] www.ausgrid.com.au/datatoshare

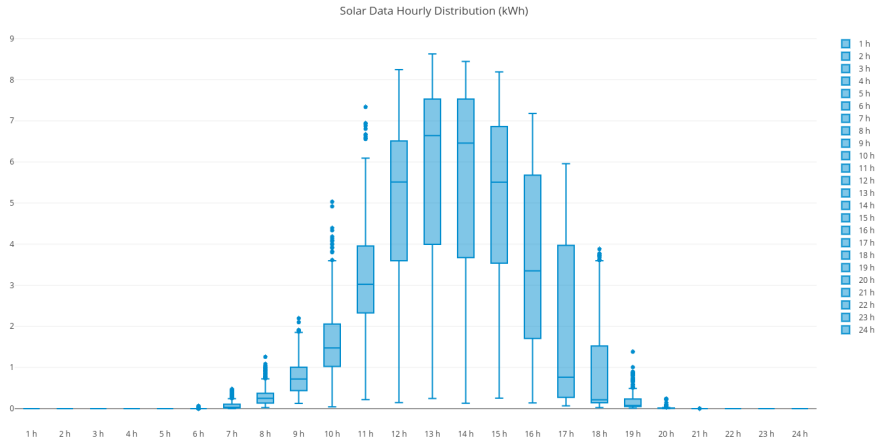

Figure 5.3: Daily solar panels production hourly distribution (kWh)

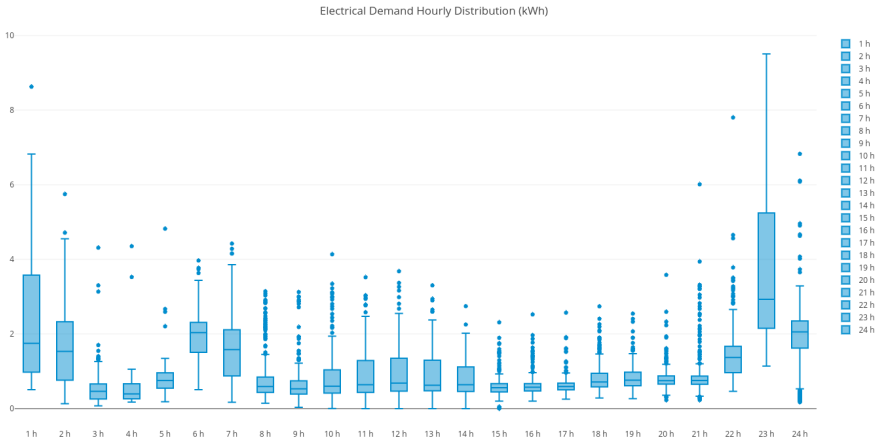

Figure 5.4: Daily electrical demand hourly distribution (kWh)

Data to model the cost of batteries and electricity

For the cost of batteries, we obtained a yearly forecast between 2010 and 2030 from Bloomberg[4]. We added an arbitrary white gaussian noise to generate synthetic random batteries prices scenarios. We display in Figure 5.5 on the left, the forecast (in blue) and the scenarios we generated (in gray). For the price of electricity, we use a realistic scenario, displayed in Figure 5.5 on the right, based on EDF blue tariff[5] for a 6 or 9 kVA subscription with peak and off-peak hours.

[4]https://www.bloomberg.com/quicktake/batteries
[5]https://particulier.edf.fr/en/home/energy-at-home/electricity/blue-tariff.html

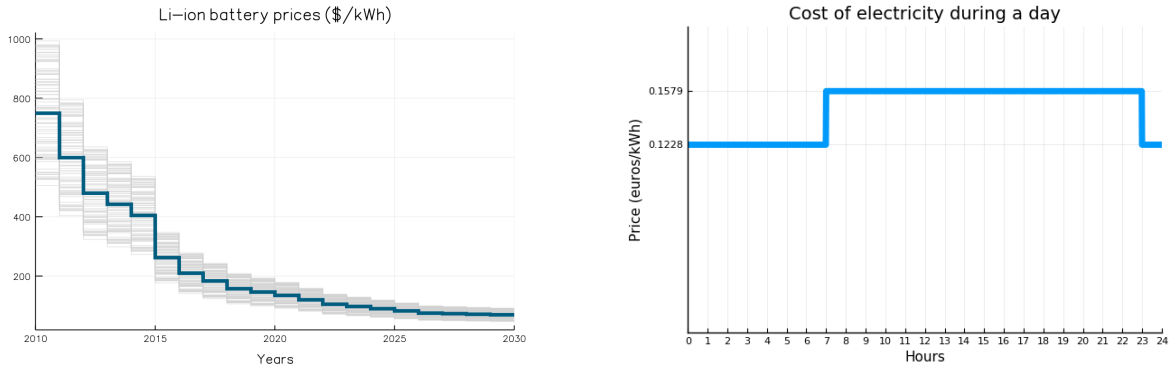

Figure 5.5: Batteries cost scenarios between 2010 and 2030, and cost of electricity

5.4.2 Long term aging and renewal of batteries

We presented the battery aging and renewal problem in Section 5.2. We recall, with the problem notations, the expression of the daily Bellman value functions for this particular problem:

$$V_d(c_d, b_{d,0}, h_{d,0}) = \min_{\boldsymbol{E}^B_{d,0:M}, \boldsymbol{R}_d} \mathbb{E}\Big[\gamma_d \sum_{m=0}^{M-1} p^e_{d,m} \times \boldsymbol{E}^E_{d,m+1} + \gamma_d \boldsymbol{P}^b_d \times \boldsymbol{R}_d + V_{d+1}(\boldsymbol{C}_{d+1}, \boldsymbol{B}_{d+1,0}, \boldsymbol{H}_{d+1,0})\Big] , \tag{5.55a}$$

$$\text{s.t } (5.9a), (5.9b) , \tag{5.55b}$$

$$(5.13), (5.14), (5.15) , \tag{5.55c}$$

$$\sigma(\boldsymbol{E}^B_{d,m}) \subset \sigma(\boldsymbol{C}_d, \boldsymbol{B}_d, \boldsymbol{H}_d, \boldsymbol{E}^S_{d,0:m}) , \tag{5.55d}$$

$$\sigma(\boldsymbol{R}_d) \subset \sigma(\boldsymbol{C}_d, \boldsymbol{B}_d, \boldsymbol{H}_d, \boldsymbol{P}^b_d) , \tag{5.55e}$$

$$\boldsymbol{C}_d = c_d , \quad \boldsymbol{B}_d = b_{d,0} , \quad \boldsymbol{H}_d = h_{d,0} . \tag{5.55f}$$

Intuitively, the daily value functions are non-increasing in the state of charge b and the state of health h because it is always preferable to have a full and healthy battery. We prove in Appendix 5.5.2 that this problem presents all the features required to apply our decomposition algorithms, namely, that the value functions V_d are non-increasing in b_d and h_d.

Splitting slow and fast decision variables

We introduce the intraday problems for all $d \in \{0, \dots, D+1\}$:

$$\phi_{(d,\geqslant)}(c_d, b_{d,0}, h_{d,0}, \boldsymbol{B}, \boldsymbol{H}) = \min_{\boldsymbol{E}^B_{d,0:M}} \mathbb{E}\left[\sum_{m=0}^{M-1} p^e_{d,m} \times \boldsymbol{E}^E_{d,m+1}\right], \tag{5.56a}$$

$$\text{s.t } (5.9a), (5.9b) , \tag{5.56b}$$

$$\boldsymbol{B}_{d,M} \geqslant \boldsymbol{B} , \ \boldsymbol{H}_{d,m} \geqslant \boldsymbol{H} , \tag{5.56c}$$

$$\sigma(\boldsymbol{E}^B_{d,m}) \subset \sigma(\boldsymbol{C}_d, \boldsymbol{B}_d, \boldsymbol{H}_d, \boldsymbol{E}^S_{d,0:m}) , \tag{5.56d}$$

$$\boldsymbol{C}_d = c_d , \ \boldsymbol{B}_d = b_{d,0} , \ \boldsymbol{H}_d = h_{d,0} . \tag{5.56e}$$

There is a small difference in the definition of the intraday problem as compared to Equation (5.25) because we keep the cost $\mathbb{E}\gamma_d \boldsymbol{P}^b_d \times \boldsymbol{R}_d$ outside. It allows to keep the capacity dynamic (5.13) and its associated target constraint (5.25b) outside the intraday problem. All the previous results can still be applied, but this decomposition is less computationally costly. We obtain the following expression for the daily value functions with the deterministic targets simplification:

$$V_{(d,\geqslant,\mathbb{X}_{d+1})}(c_d, b_{d,0}, h_{d,0}) = \min_{\boldsymbol{R}_d, b, h} \gamma_d \phi_{(d,\geqslant)}(c_d, b_{d,0}, h_{d,0}, b, h) + \mathbb{E}\Big[\gamma_d \boldsymbol{P}^b_d \times \boldsymbol{R}_d + V_{d+1}(\boldsymbol{C}_{d+1}, \boldsymbol{B}_{d+1,0}, \boldsymbol{H}_{d+1,0})\Big] , \tag{5.57a}$$

$$\text{s.t } (5.13), (5.14), (5.15) , \tag{5.57b}$$

$$\sigma(\boldsymbol{R}_d) \subset \sigma(\boldsymbol{C}_d, \boldsymbol{B}_{d,0}, \boldsymbol{H}_{d,0}, \boldsymbol{P}^b_d) , \tag{5.57c}$$

$$\boldsymbol{C}_d = c_d , \ \boldsymbol{B}_{d,0} = b_{d,0} , \ \boldsymbol{H}_{d,0} = h_{d,0} . \tag{5.57d}$$

It falls down to choose end of the day maximum aging and minimum state of charge at the beginning of the day as well as battery renewal once price of batteries is observed. We recall that, due to the dynamics (5.13), (5.14) and (5.15), the random variables $\boldsymbol{C}_{d+1}$, $\boldsymbol{B}_{d+1,0}$ and $\boldsymbol{H}_{d+1,0}$ depend on b and h in (5.57).

Simplifying the intraday problem

In this application, we are in the situation described in Assumption 34 regarding the aging dynamics:

$$\phi_{(d,\geqslant)}(c_d, b_{d,0}, h_{d,0}, b_{d,M}, h_{d,M}) = \phi_{(d,\geqslant)}(c_d, b_{d,0}, h_{d,0} - h_{d,M}, b_{d,M}, 0) . \tag{5.58}$$

Finally, as we want to focus on the aging, we neglect the state of charge target replacing it by an empty state of charge target. In fact, we assume that the battery is empty at the end and at the beginning of everyday, which is a pessimistic assumption. We have to compute the following functions:

$$\widetilde{\phi}_{(d,\geqslant)}(c_d, h_{d,0} - h_{d,M}) = \phi_{(d,\geqslant)}(c_d, 0, h_{d,0} - h_{d,M}, 0, 0) . \tag{5.59}$$

Then, we can compute the daily value functions approximation $V_{(d,\geqslant,\mathbb{X}_{d+1})}$ by exhaustive search in discretized daily states, controls and targets spaces. In our numerical

experimentation, we use the same uncertainties model everyday for the noises and we assume that the prices of electricity are the same everyday. Then we compute only $\widetilde{\phi}_{(0,\geqslant)}$.

To compute the function $\widetilde{\phi}_{(0,\geqslant)}$, we apply the SDDP algorithm, assume stagewise independence of the noises, for every possible capacities $c_d \in \mathbb{C}$, that is, in our case for all $\mathbb{C} = \{0, 1 \ldots, 19, 20\}$. Applying SDDP for a capacity c_d, we obtain a convex polyhedral lower approximation of $\widetilde{\phi}_{(0,\geqslant)}(c_d, \cdot)$.

Numerical simulation of a two-time scales policy

To simulate a policy, we draw one scenario of solar production and electrical demand for every minutes step of the 20 years horizon. We display in Figure 5.6 left, a net production (solar production minus demand) over one week. We also draw one scenario of cost of batteries for everyday of the horizon in Figure 5.6 right. We apply the simulation strategy described in §5.3.4.

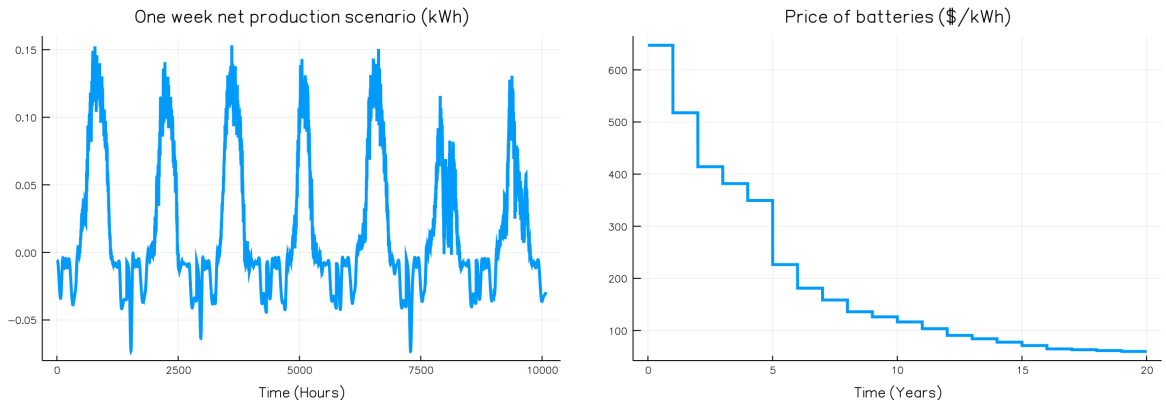

Figure 5.6: Net production scenario over one week and price of batteries scenario over 20 years

Our instance has then the following numerical features:

- Horizon: 20 years,
- Number of time steps: $10,512,000$ minutes,
- Battery capacity: between 0 and 20 kWh,
- One periodicity class: all days are similar.

Comparison of two policies

We compare two policies. One policy is computed using the value functions (5.57) with the simplification that the state of charge target is restricted to 0, that is, we do not constrain the state of charge of the battery at the end of every day. We call this strategy “aging control" or AC. The other policy is computed using the value functions (5.57) without state of charge constraints as well. However there is another simplification: the health of the battery at the end of the day has only to remain above 0, that is, there is no health target every day. This policy is called “without aging control" or WAC. We

compare hereby the two policies on one twenty years simulation. The reference case is just the electricity bill of the house along the drawn scenario and without battery at all.

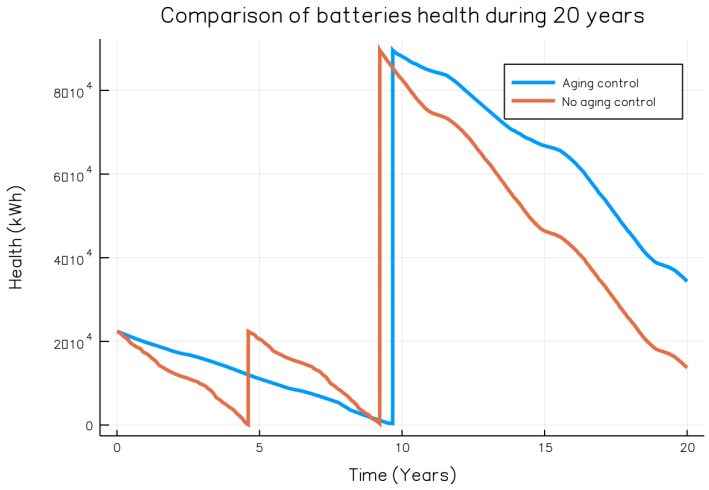

Figure 5.7: Health decrease comparison

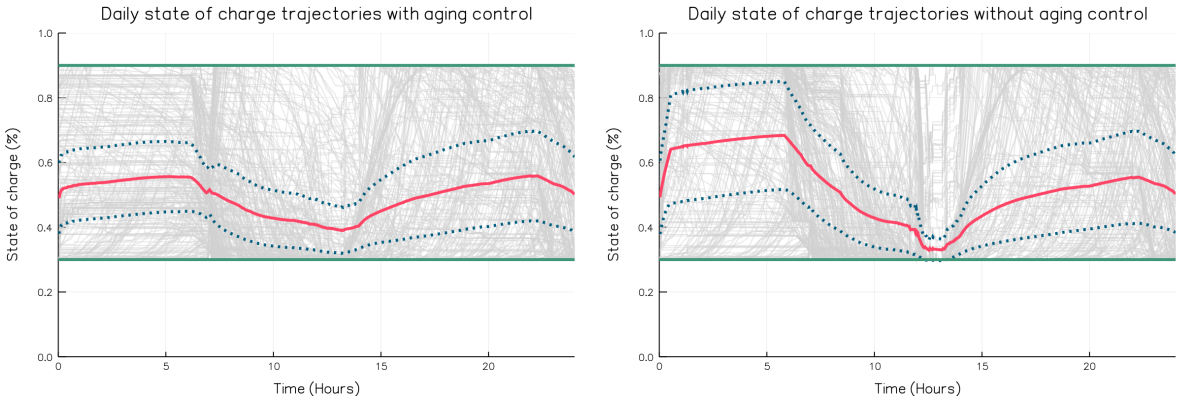

Figure 5.8: Daily state of charge trajectories comparison (kWh)

We observe unsurprisingly in Figure 5.7 that the AC strategy renews the battery one less time than the WAC strategy. We observe that both strategies buy the same first capacity, that is a 4 kWh battery. The WAC buys one more 4 kWh battery after about 5 years while the ac ones waits 10 years. At year 10, both strategies buy a 16 kWh battery that has more health at the end of the horizon with the AC strategy than it does with the WAC strategy.

We display in Figure 5.8 the impact of both strategies on the state of charge of the batteries everyday. In red is the mean state of charge. We observe unsurprisingly that the WAC strategy uses more intensively the battery everyday.

Finally, on this single simulation we obtain the following discounted expenses for electricity bill and battery purchases:

- Reference case (electricity bill without battery and solar panels): 25404 €,
- Without aging control: 24702 €, minus 3 % compared to reference,

- Aging control: 22613 €, minus 11% compared to reference.

On one simulation we observe that the best economical strategy is to buy batteries and to control their aging, against one particular battery scenario. We should make many simulations over multiple battery prices to conclude due to the stochasticity of these battery prices.

Remark 41 (On computation time)**.** *The targets decomposition algorithm is here applied because the problem is way too large to apply SDP straightforwardly (*10512000 *time steps with* 3 *state variables). The computation time would be tremendous even if SDP can be parallelized and moreover the memory needed is way above the memory of a regular computer. With the targets decomposition, we display reasonable computation times: the intraday problem* $\widetilde{\phi}_{(0,\geqslant)}$ *took around* 1 *hour to compute, the daily value functions around* 45 *minutes and a simulation around* 45 *minutes as well. We present a more rigorous discussion on complexity and computation times in the next experimentation.*

5.4.3 Decomposition methods comparison on a simple aging problem

In this part, we focus on aging of a battery with a given capacity over five days. The objective is to compare targets and weights decomposition algorithms to the results obtained using SDP, and SDDP, straightforwardly over the whole horizon. We assume that the house is equipped with a battery with a given capacity c_0 that will not change during the five days. This assumption removes the *slow scale* renewal decision variable from the previous experiment (in §5.4.2). In the application of this new paragraph, we demonstrate that the decomposition algorithms are also efficient to solve stochastic optimization problems with many time stages but a single scale

Problem instance

We present hereby the different parameters of the problem we solve.

- Horizon: 5 days.
- Time step : 15 minutes.
- Number of time steps : 480.
- Capacity: $c_0 = 13$ kWh battery.
- Initial health: $h_{0,0} = 100$ kWh of exchangeable energy.

Target daily value functions without renewal

Without battery renewal, we obtain the following daily target value functions defined by backward induction:

$$V_{(d,\geqslant,\mathbb{X}_{d+1})}(b_{d,0}, h_{d,0}) = \min_{b,h} \phi_{(d,\geqslant)}(c_0, b_{d,0}, h_{d,0}, b, h) + V_{(d+1,\geqslant,\mathbb{X}_{d+2})}(b, h) \ , \tag{5.60a}$$

where $\phi_{(d,\geqslant)}$ is the intraday problem defined in (5.56).

Once again, we decide to neglect state of charge target to focus on health here. We fix state of charge target to zero, or empty battery. We also take the same net production uncertainties model everyday of the five days horizon. As in the previous experimentation, we then compute the single intraday problem $\widetilde{\phi}_{(0,\geqslant)}(c_0,\cdot)$ with only one capacity this time. In practice, this problem is solved using SDDP provided a polyhedral lower approximation of $\widetilde{\phi}_{(0,\geqslant)}(c_0,\cdot,\cdot)$. We call the target value functions $V^T_{(d,\geqslant,\mathbb{X}_{d+1})}$:

$$V^T_{(d,\geqslant,\mathbb{X}_{d+1})}(h_{d,0}) = \min_h \widetilde{\phi}_{(d,\geqslant)}(c_0, b_{d,0}, h_{d,0} - h,) + V^T_{(d+1,\geqslant,\mathbb{X}_{d+2})}(h)\ . \tag{5.61a}$$

As the $\widetilde{\phi}_{(0,\geqslant)}(c_0,\cdot,\cdot)$ is convex polyhedral, we can solve the backward recursion (5.61a) using linear programming. Moreover, we can obtain a convex polyhedral lower approximation of $V^T_{(d,\geqslant,\mathbb{X}_{d+1})}$. In the sequel, when we refer to $V^T_{(d,\geqslant,\mathbb{X}_{d+1})}$, we refer to this polyhedral approximation.

Weights daily value functions without renewal

As we focus only on aging we do not dualize the whole intraday dynamics here. We dualize only the aging dynamic. We define the relaxed intraday problem:

$$\psi_d(c_0, h_{d,0}, \lambda_d) = \tag{5.62a}$$

$$\min_{\boldsymbol{E}^B_{d,0:M}} \mathbb{E}\left[\sum_{m=0}^{M-1} p^e_{d,m}\boldsymbol{E}^E_{d,m+1}\right] + \mathbb{E}\left[\lambda_d \times \sum_{m=0}^{M-1}\frac{1}{\rho_d}\boldsymbol{E}^{B-}_{d,m} - \rho_c\boldsymbol{E}^{B+}_{d,m}\right], \tag{5.62b}$$

$$\text{s.t } \boldsymbol{B}_{d,m+1} = \boldsymbol{B}_{d,m} - \frac{1}{\rho_d}\boldsymbol{E}^{B-}_{d,m} + \frac{1}{\rho_d}\rho_c\boldsymbol{E}^{B+}_{d,m}\ , \tag{5.62c}$$

$$\boldsymbol{B}_{d,M} \geqslant \underline{B}\ , \tag{5.62d}$$

$$\sigma(\boldsymbol{E}^B_{d,m}) \subset \sigma(\boldsymbol{B}_d, \boldsymbol{H}_d, \boldsymbol{E}^S_{d,0:m})\ , \tag{5.62e}$$

$$\boldsymbol{B}_d = \underline{B}\ ,\ \ \boldsymbol{H}_d = h_d\ . \tag{5.62f}$$

So, as λ_d deterministic, the objective turns into

$$\mathbb{E}\left[\sum_{m=0}^{M-1} p^e_{d,m}\boldsymbol{E}^E_{d,m+1} + \frac{\lambda_d}{\rho_d}\boldsymbol{E}^{B-}_{d,m} - \lambda_d\rho_c\boldsymbol{E}^{B+}_{d,m}\right]. \tag{5.63}$$

It makes it possible to solve the problem (5.62) using SDDP producing a convex polyhedral approximation of $\psi_d(c_0,\cdot,\lambda_d)$. We make the periodicity assumption as in the targets decomposition.

Then, we compute weights daily value functions solving the backward recursion:

$$V^W_{(d,\geqslant,\mathbb{E})}(h_{d,0}) = \sup_{\lambda\in\Lambda_{d+1}}\left[\psi_0(c_0,h_{d,0},\lambda) + \min_{h\in\mathbb{H}}\Big(-\lambda\cdot h + V^W_{(d+1,\geqslant,\mathbb{E})}(h)\Big)\right]. \tag{5.64}$$

In this case we computed many intraday problems value $\psi_0(c_0,\cdot,\lambda)$ for different $\lambda \in [0,2]$. We need to compute the values ψ_d only for $d = 0$ using the periodicity assumption. It appears that, above $\overline{\lambda} = 0.08$, the mapping $\lambda \to \psi_0(c_0,\cdot,\lambda)$ is constant.

Therefore, we perform the maximization (5.64) by exhaustive search in the discrete space $\{-0.08, -0.0784, \dots, -0.0016, 0\}$, that is, all the weights between -0.08 and 0 with step 0.0016, which gives 51 weights. The nested minimization over h in (5.64) is performed by exhaustive search in the discretized health space as well.

Computing minute value functions

Both targets and weights decomposition methods provide daily value functions $V^T_{(d,\geqslant,\mathbb{X}_{d+1})}$ and $V^W_{(d,\geqslant,\mathbb{E})}$. Next, we compute *intraday value functions* using these value functions as final cost in new intraday problems. For example, in the targets case, we compute the family of intraday value functions $V^T_{d,m}$ by applying the SDDP algorithm to the problem, because the final cost $V^T_{(d+1,\geqslant,\mathbb{X}_{d+2})}$ is convex polyhedral:

$$\min_{\boldsymbol{E}^B_{d,0:M}} \ \mathbb{E}\left[\sum_{m=0}^{M-1} p^e_{d,m} \times \boldsymbol{E}^E_{d,m+1} + V^T_{(d+1,\geqslant,\mathbb{X}_{d+2})}(\boldsymbol{H}_{d,M})\right], \tag{5.65a}$$

$$\text{s.t } (5.9a), (5.9b)\ , \tag{5.65b}$$

$$\sigma(\boldsymbol{E}^B_{d,m}) \subset \sigma(\boldsymbol{B}_{d,m}, \boldsymbol{H}_{d,m}, \boldsymbol{E}^S_{d,m})\ . \tag{5.65c}$$

For the weights case, we apply SDP as we computed an approximation of $V^W_{(d+1,\geqslant,\Lambda_{d+2})}$ on a grid.

Two straightforward references

We compare these two daily decomposition methods to two straightforward approaches because the problem is not too large to apply them. However we will observe that numerically these classical methods perform poorly to produce the "true" value functions. We apply SDP and SDDP to the following global problem assuming stagewise (minutes) independence of the noises:

$$\min_{\boldsymbol{E}^B_{d,0:M}} \ \mathbb{E}\left[\sum_{d=0}^{D-1}\sum_{m=0}^{M-1} p^e_{d,m} \times \boldsymbol{E}^E_{d,m+1}\right], \tag{5.66a}$$

$$\text{s.t } (5.9a), (5.9b)\ , \tag{5.66b}$$

$$\boldsymbol{B}_{d+1,0} = \boldsymbol{B}_{d,M}\ ,\ \ \boldsymbol{H}_{d+1,0} = \boldsymbol{H}_{d,M}\ , \tag{5.66c}$$

$$\sigma(\boldsymbol{E}^B_{d,m}) \subset \sigma(\boldsymbol{B}_{d,m}, \boldsymbol{H}_{d,m}, \boldsymbol{E}^S_{d,m})\ . \tag{5.66d}$$

With both algorithms, we obtain a family of intraday value functions respectively called $V^{SDP}_{d,m}$ and $V^{SDDP}_{d,m}$. To be consistent with the two previous methods, at the beginning of each day in a given state $b_{d,0}, h_{d,0}$ we force the value function V^{SDP}_0 to satisfy the equality $V^{SDP}_{d,0}(b_{d,0}, h_{d,0}) = V^{SDP}_{d,0}(0, h_{d,0})$ in order to ignore the state of charge at the end and beginning of each day as well. We do the same for SDDP.

Numerical results

We now present numerical results comparing the four methods. Figure 5.9 presents in-sample simulation results with the four algorithms along one scenario. It means that

we have drawn one scenario from the uncertainties distribution we used to compute the daily value functions. We observe that the aging of the battery as well as its state of charge over the 5 days seems to be the same for all algorithms. We observed the same fact on $10,000$ scenarios, that are not displayed here.

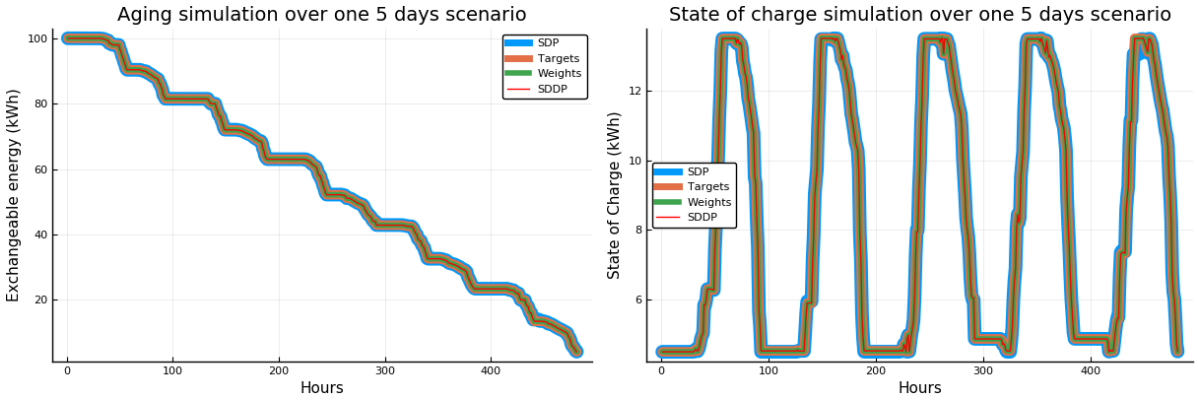

Figure 5.9: Aging and state of charge simulation over 5 days with different methods

Figure 5.10 presents the distribution of the difference of costs between each pair of algorithms along $10,000$ scenarios. All the histograms are centered around 0, hence it seems that all algorithms perform equivalently with a small win for the weights decomposition over the targets decomposition. The mean difference is approximately zero between all methods.

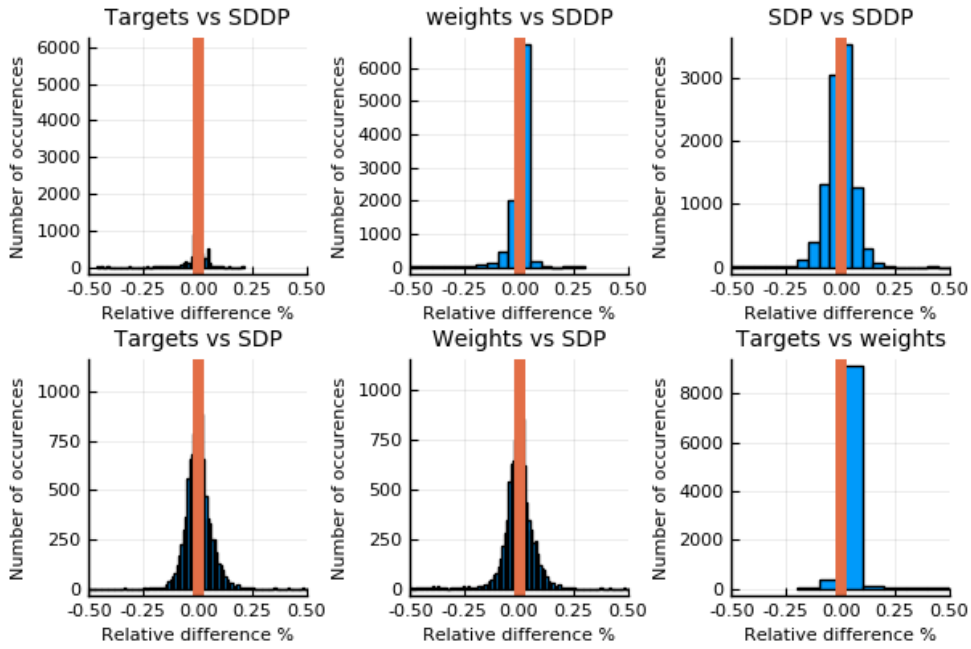

Figure 5.10: Simulation costs comparison

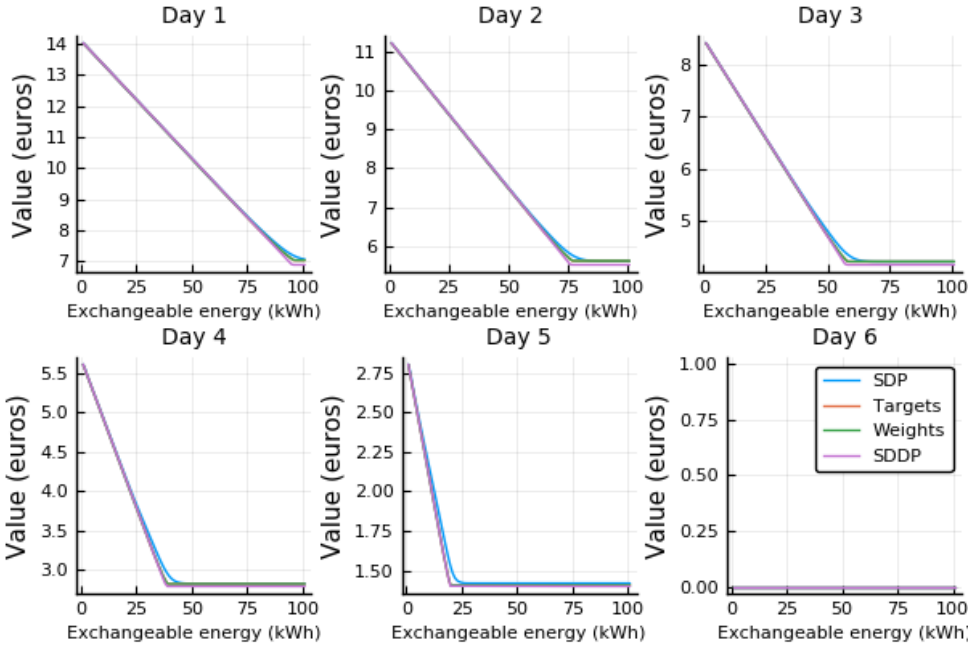

Figure 5.11: Daily value functions comparison

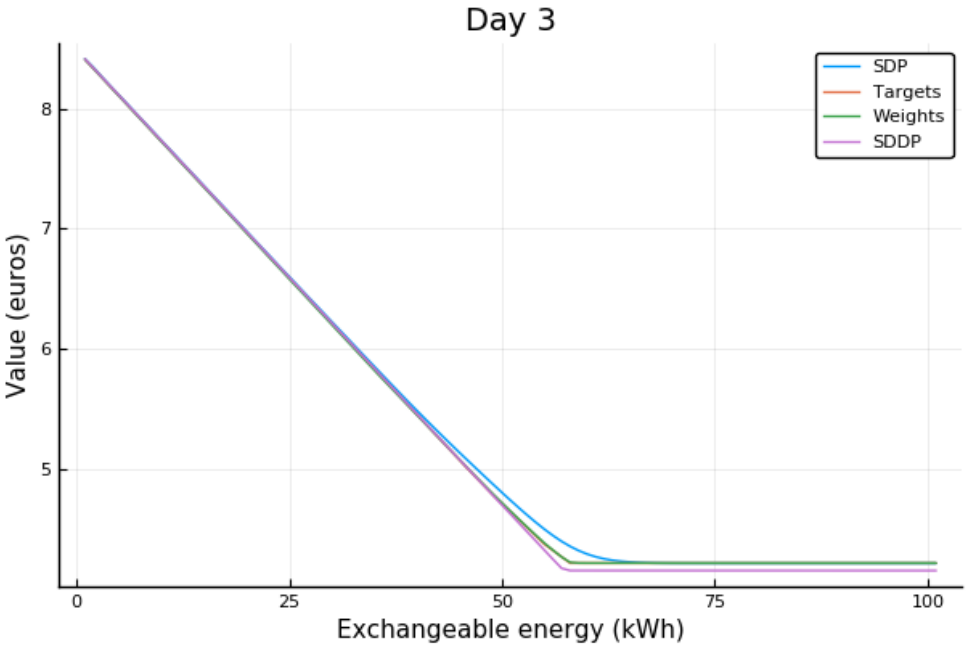

Figure 5.12: Focus on daily value function on day number 3

Figure 5.11 displays the six daily value functions, the sixth one being the final cost equal to zero. We observe that all methods compute approximately the same daily value functions. On Figure 5.12 we focus on day 3 and we observe the following order between value functions: $V_{3,0}^{SDDP} \leqslant V_{(3,\geqslant,\mathbb{E})}^{W} \leqslant V_{(3,\geqslant,\mathbb{X}_4)}^{T} \leqslant V_{3,0}^{SDP}$. We observe the same thing on all the days. This is consistent with the fact that SDDP provides a lower approximation of the true value functions, while SDP, because of discretization of state and control spaces

and interpolation, provides an upper approximation. It is surprising to find the value functions computed using targets and weights decomposition between these two bounds even with the deterministic weights and targets simplification. We recall that the "true" value functions V_d satisfy the inequality:

$$V^W_{(d,\geqslant,\mathbb{E})} \leqslant V_d \leqslant V^T_{(d,\geqslant,\mathbb{X}_{d+1})} \,. \tag{5.67}$$

	SDP	**Weights**	**SDDP**	**Targets**
Intraday resolution (SDDP)	$\varnothing$	51×14 sec	$\varnothing$	14 sec
Daily values functions	$\varnothing$	0.15 sec	$\varnothing$	0.59 sec
Minute values functions	22.5 min	5×4.5 min	3.6 min	5×14 sec
Total CPU time	22.5 min	34.4 min	3.6 min	84.6 sec
Total time (with parallelisation)	22.5 min	5.0 min	3.6 min	24.6 sec
Gap ($200 \times \frac{mc-v}{mc+v}$)	0.91 %	0.32 %	0.90 %	0.28 %

Table 5.1: Algorithms numerical results comparison

Table 5.1 presents the computation times of the algorithms as well as their gap which is measured as the relative difference between the initial value $V_{0,0}(0, 100)$ computed by the algorithm and the mean cost obtained by simulation over 10000 scenarios from state $(0, 100)$. SDP and SDDP do not display intraday resolution and daily value functions times as they are applied directly to the global problem, computing both daily and minute value functions.

- We observe that daily value functions computation for targets and weights algorithms is really fast, but that the intraday problems resolution for weights is costly. This is due to the exhaustive search in the weights space (of cardinal 51 here). This time is significantly lower in the targets case because SDDP already explores the initial state space.

- We observe that targets and weights algorithms have the best gaps. We could improve the one of SDDP but we did not manage to improve it significantly after more than 1 hour. The convergence of SDDP (measured with the gap) is sensitive to the number of time stages [131].

- The time required to compute value functions in the weights case is the same as SDP, as it computes 5 times a 5 time smaller SDP. However the weights algorithm permits to parallelize this phase or even to distribute it accross days, which is impossible with SDP.

Finally, we observe that the targets algorithm, where all the intraday problems are solved using SDDP is the fastest algorithm. It has the best gap, displays approximately the same costs and value functions as the other algorithms. Moreover, the resolution of the intraday problems with final cost can be parallelized or distributed across days. In fact it is a way to accelerate the resolution of the problem with multiple applications of SDDP, instead of one straighforward application, when a convex problem displays a high number of time steps, monotonicity and some good properties. That kind of approach could be appealing for the algorithm Mixed Integer Dynamic Approximation Scheme [135] as it is really sensible to the number of time steps and relies on monotonicity as well.

5.4.4 Sizing of a battery using targets decomposition and Stochastic Dual Dynamic Programming

In this part, we use the lower convex polyhedral approximations of the intraday target problems values $\phi_{(d,\geqslant)}$ provided by SDDP, to compute an optimal capacity for the house.

A sizing problem without battery renewal

We apply the targets decomposition scheme, introduced in §5.3.2 to the same aging problem without battery renewals to compute the net present value for a given battery capacity. We introduce the intraday problems:

$$\phi_{(d,\geqslant)}(c_d, h_{d,0} - h_{d,M}) = \min_{\boldsymbol{E}^B_{d,0:M}} \mathbb{E}\left[\sum_{m=0}^{M-1} p^e_{d,m} \times \boldsymbol{E}^E_{d,m+1}\right], \tag{5.68a}$$

$$\text{s.t } (5.9a), (5.9b) , \tag{5.68b}$$

$$\boldsymbol{B}_{d,M} \geqslant \underline{B} , \ \ \boldsymbol{H}_{d,m} \geqslant 0 , \tag{5.68c}$$

$$\boldsymbol{C}_{d,m} = \boldsymbol{C}_{d,m+1} , \tag{5.68d}$$

$$\sigma(\boldsymbol{E}^B_{d,m}) \subset \sigma(\boldsymbol{C}_d, \boldsymbol{B}_{d,0}, \boldsymbol{H}_{d,0}, \boldsymbol{E}^S_{d,0:m}) , \tag{5.68e}$$

$$\boldsymbol{C}_{d,0} = c_d , \ \ \boldsymbol{B}_{d,0} = \underline{B} , \ \ \boldsymbol{H}_{d,0} = h_{d,0} - h_{d,M} . \tag{5.68f}$$

The intraday problem remains unchanged compared to the previous experiment 5.4.2 except that we augment the state with a constant dynamic for the capacity, see (5.68d). We recover the value $\phi_{(d,\geqslant)}$ as we can eliminate trivially the capacity state using (5.68d). This trick makes it possible to compute a lower convex polyhedral approximation $\widetilde{\phi}_{(d,\geqslant)}$ by applying the SDDP algorithm with a grid of initial states.

It is then possible to apply SDDP to the daily target value functions recursion without battery renewal:

$$V_{(d,\geqslant,\mathbb{X}_{d+1})}(c_d, h_{d,0}) = \min_h \gamma_d \phi_{(d,\geqslant)}(c_d, h_{d,0} - h) + V_{(d+1,\geqslant,\mathbb{X}_{d+2})}(c_d, h_{d,0} - h) . \tag{5.69a}$$

The application of a two-time scale SDDP is relevant only if daily value functions are required. Here, only the sizing of the battery matters; hence we can just solve a linear program to compute the optimal capacity as presented in the next section.

We worked on an industrial case where the economic profit was not the only objective. We had to determine also the rate of self consumption and self production for a given capacity. For this purpose, we needed to compute daily value functions so as to produce a simulation policy to evaluate these rates using Monte Carlo simulation.

Computation of the optimal capacity

We obtained a convex polyhedral approximation of $\phi_{(d,\geqslant)}$ for every relevant $d \in \{0, \dots, D\}$ possibly using periodicity. We can then compute an optimal capacity at the first day 0

by solving the following linear program for a given price of batteries per kWh p_0^b:

$$\min_{c\in[\underline{c},\overline{c}]} \min_{h_{0,0},\dots,h_{D+1,0}} \quad p_0^b \times c + \sum_{d=0}^{D} \gamma_d \phi_{(d,\geqslant)}(c, h_{d,0} - h_{d+1,0}) \ , \tag{5.70a}$$

$$\text{s.t } h_{0,0} = 2N_{cycles} \times c \ , \tag{5.70b}$$

$$h_{d,0} \geqslant h_{d+1,0} \ . \tag{5.70c}$$

This is a linear program using the convex lower polyhedral approximation of $\phi_{(d,\geqslant)}$. The constraint (5.70c) makes it possible to lower the size of the search space but is implicit as a negative initial age $h_{d,0} - h_{d+1,0}$ leads to $\phi_{(d,\geqslant)}(c, h_{d,0} - h_{d+1,0}) = +\infty$.

Numerical results

We compute the value of Problem (5.68) using SDDP with 500 iterations in 1.7 minute. We test convergence by MonteCarlo simulation and we reach a gap of 0.1% between the upper bound, computed by Monte Carlo, and the lower bound, computed by SDDP. Then, we were able to solve Problem (5.70) for a given horizon and price of batteries in 1.7 second for a 1 year horizon and 71 seconds for 12 years horizon using CPLEX. We present numerical results as contour plots on Figure 5.13 and 5.14. The first one presents the optimal battery capacity (black is 0 kWh, yellow is 20 kWh) as a function of the investment horizon D and the prices of batteries p_0^b. The second presents the corresponding discounted benefit (black is 0 €, yellow is above 2100 €) as a function of horizon and prices as well. The third one presents the corresponding expected lifetime for the battery (black is 0 years, yellow is 7 years), as a function of horizon and prices. We observe that a battery would be economically interesting if we consider the investment over at least 2 years and below 150 euros per kWh, below the optimal battery has capacity 0 kWh. Over a 12 years horizon, the best capacity is the largest one, that is 20 kWh capacity, and the expected net benefit would be 2100 euros. We finally observe on Figure 5.14 that even the largest battery is not expected to last more than 7 years. This is consistent with the plateau we observe on Figure 5.13 above 7 years.

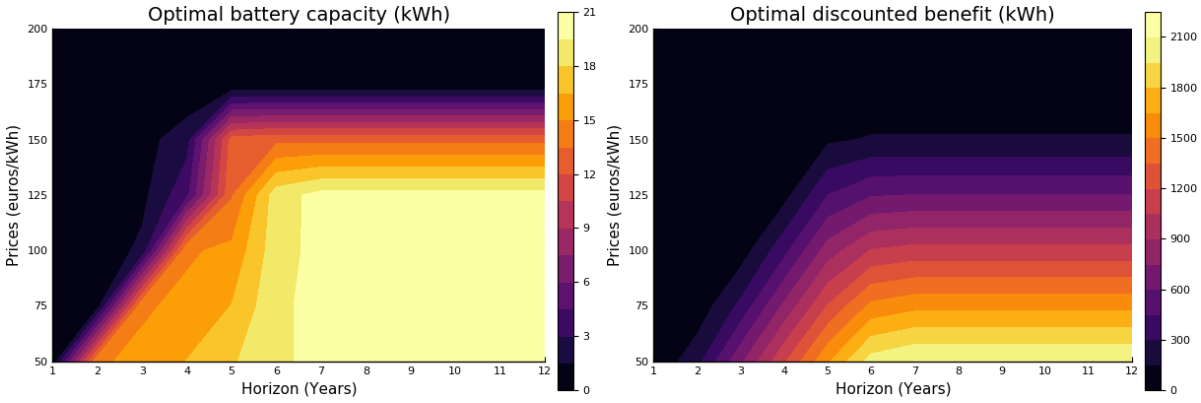

Figure 5.13: Optimal battery capacity and benefit as a function of prices and horizon

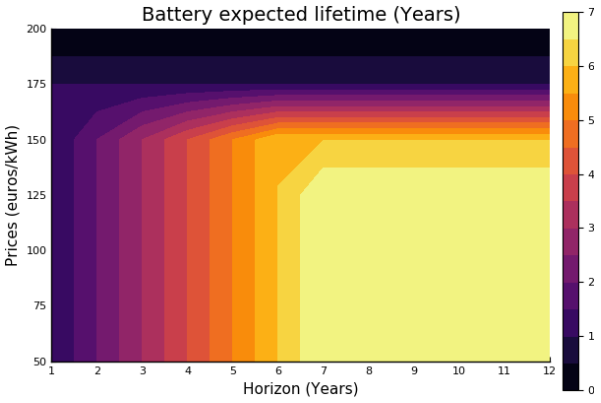

Figure 5.14: Expected battery lifetime as a function of prices and horizon

Conclusion

We introduced a two-time scales stochastic optimization problem for a battery charge/discharge, aging and renewal management problem. The motivation for two-time scales modeling originated from the existence of decisions that have to be made every minute (charge/dicharge) and decisions that have to be made once a day (renewal). We presented two algorithmic methods to compute daily value functions to solve these problems with an important number of time steps and decisions on different time scales. We applied these algorithms to three realistic applications for the original problem, a simple aging problem and a sizing problem. We conclude that these methods make it possible to solve problems with different time scales as well as an important number of time steps. Moreover, with the aging application (in §5.4.3) we observe that these methods make it possible to decompose a single time scale long problem in time. Our two algorithms perform better on a long problem displaying periodicity than a straightforward use of SDP or SDDP. It might generally be useful in order to improve algorithms whose convergence is sensitive to the number of time steps. It makes it possible for example to parallelize SDP over time, not states, and to speed up the convergence of SDDP.

Algorithms such as Mixed Integer Dynamic Approximation Scheme produce $T \times \epsilon$ optimal solutions for multistage stochastic optimization problems with binary variables, where T is the number of stages and ϵ a user tuned parameter. It might be interesting to apply such time decomposition methods to improve the quality of the solution and to speed up the convergence of these algorithms.

Theses methods are to be compared to value iteration and policy iteration applied to infinite horizon problems. Value iteration and policy iteration require both a stationary assumption. Our methods relax this assumption, proving efficient for problems displaying periodicity and monotonicity.

Finally, these methods could make it possible to mix Stochastic Programming and Stochastic Dynamic Programming methods. Stochastic Programming and scenario decomposition methods display a complexity exponential in the number of time steps. Targets and weights decomposition could be a way to make these methods more tractable on problems with an important number of time steps by splitting into subproblems with less time steps.

5.5 Appendix

In this section, we present theoretical results to decompose two-time scales stochastic optimization problems as well as proofs of monotonicity and convexity of a battery management problem.

5.5.1 An abstract optimization problem

Let $(n_v, n_u) \in \mathbb{N}^* \times \mathbb{N}^*$ be given and two subsets $\mathbb{V} \subset \mathbb{R}^{n_v}$ and $\mathbb{U} \subset \mathbb{R}^{n_u}$ equipped with the element-wise partial order $\leqslant$. Let two proper extended real valued functions $l : \mathbb{U} \to (-\infty, +\infty]$ and $V : \mathbb{V} \to (-\infty, +\infty]$ and a mapping $f : \mathbb{U} \to \mathbb{V}$. We study different ways to solve or approximate the following optimization problem:

$$v = \inf_{u \in \mathcal{U}} \quad \Big(l(u) + V(f(u)) \Big) \tag{5.71}$$

where $\mathcal{U}$ is a subset of $\mathbb{U}$. We call equation (5.71) an *abstract Bellman equation.*

Remark 42. *We study this general equation for its application in two-time scales dynamic programming. We assume that a decision maker has to make one decision every minute. The function l represents a cost incurred daily by the decision made every minute. V models a cost of the future incurred by the decision made every minute that change a state through a dynamic equation f. The decision maker wants to minimize a compromise between these two costs.*

In the whole section, we make a monotonicity assumption to decompose the optimization problem (5.71).

Assumption 43. *V is a non decreasing function.*

Decomposition by targets

For all $\alpha \in \mathbb{V}$ we introduce the following parametrized problems:

$$L_{(=)}(\alpha) = \inf_{u \in \mathcal{U} \cap f^{=\alpha}} l(u) \quad \text{and} \quad L_{(\leqslant)}(\alpha) = \inf_{u \in \mathcal{U} \cap f^{\leqslant \alpha}} l(u) \ , \tag{5.72}$$

where the level sets $f^{=\alpha}$ and $f^{\leqslant \alpha}$ are respectively given by

$$f^{=\alpha} = \{u \in \mathbb{U} | f(u) = \alpha\} \tag{5.73}$$

and

$$f^{\leqslant \alpha} = \{u \in \mathbb{U} | f(u) \leqslant \alpha\} \ . \tag{5.74}$$

In the next lemma, we use the value functions $L_{(=)}$ and $L_{\leqslant}$ to obtain lower bounds to the optimization problem (5.71).

Lemma 44. *The value $v_{(=)}$ and $v_{\leqslant}$ of the two optimization problems*

$$v_{(=)} = \inf_{\alpha \in \mathbb{V}} \big(L_{(=)}(\alpha) + V(\alpha) \big) \ , \tag{5.75}$$

$$v_{(\leqslant)} = \inf_{\alpha \in \mathbb{V}} \big(L_{(\leqslant)}(\alpha) + V(\alpha) \big) \ , \tag{5.76}$$

give lower bounds to the optimization Problem (5.71):

$$v_{(\leqslant)} \leqslant v_{(=)} = v \ . \tag{5.77}$$

Moreover, under the monotonicity Assumption 43 we have that

$$v_{(\leqslant)} = v_{(=)} = v \ . \tag{5.78}$$

Proof. $(v_{(=)} = v)$:

$$\begin{aligned} v_{(=)} &= \inf_{\alpha \in \mathbb{V}} \big(L_{(=)}(\alpha) + V(\alpha) \big) \\ &= \inf_{\alpha \in \mathbb{V}} \inf_{u \in \mathcal{U}} \ l(u) + V(\alpha) \\ &\qquad \text{s.t } f(u) = \alpha \\ &= \inf_{u \in \mathcal{U}} \inf_{\alpha \in \mathbb{V}} \ l(u) + V(\alpha) \\ &\qquad \text{s.t } f(u) = \alpha \\ &= \inf_{u \in \mathcal{U}} l(u) + V(f(u)) = v \end{aligned}$$

$(v_{(\leqslant)} \leqslant v)$: Let $u \in \mathcal{U}$ be given and set $\alpha = f(u)$. We successively have

$$\begin{aligned} v_{(\leqslant)} &\leqslant L_{(\leqslant)}(\alpha) + V(\alpha) && \text{(by (5.76))} \\ &\leqslant l(u) + V(\alpha) && (u \text{ is admissible for } L_{(\leqslant)}(\alpha)) \\ &\leqslant l(u) + V(f(u)) && (\alpha = f(u)) \end{aligned}$$

so finally, as the inequality holds for any $u \in \mathcal{U}$:

$$v_{(\leqslant)} \leqslant \inf_{u \in \mathcal{U}} \big(l(u) + V(f(u)) \big) = v \qquad \text{(by (5.71))}$$

($v \leqslant v_{\leqslant}$ under monotonicity assumption). For any $\epsilon > 0$ let $\alpha_\epsilon \in \mathbb{V}$ be a ϵ-optimal solution for the optimization problem $v_{(\leqslant)}$ and $u_\epsilon \in \mathbb{U}$ be an ϵ-optimal solution for the optimization problem $L_{(\leqslant)}(\alpha_\epsilon)$.

$$\begin{aligned} v_{(\leqslant)} + 2\epsilon &\geqslant l(u_\epsilon) + V(\alpha_\epsilon) \\ &\geqslant l(u_\epsilon) + V(f(u_\epsilon)), \qquad (\text{ monotonicity of } V \text{and admissibility of } u_\epsilon \text{ for } L_{(\leqslant)}) \\ &\geqslant v \end{aligned}$$

The proof is complete. □

Remark 45. *The same relation holds between $v_{(=)}$ and $v_{(\geqslant)}$ in the converse case where V is non-increasing.*

Decomposition by weights using Fenchel duality

Let $(\Lambda, \leqslant)$ be a subset of $\mathbb{R}^{n_\lambda}$ equipped with the element-wise partial order $\leqslant$ and $\langle \cdot, \cdot \rangle : \Lambda \times \mathbb{V} \to [-\infty, +\infty]$ a bilinear coupling. For all $\lambda \in \Lambda$ we introduce the following relaxed version of $L_=$:

$$H(\lambda) = \inf_{u \in \mathcal{U}} \quad l(u) + \langle \lambda, f(u) \rangle \ . \tag{5.79}$$

We introduce as well the Fenchel conjugate of a function ϕ.

$$\phi^*(\lambda) = \sup_{\alpha \in \mathbb{V}} \quad \langle \lambda, \alpha \rangle - \phi(\alpha) \ . \tag{5.80}$$

Lemma 46. *The following equality holds*

$$H(\lambda) = -L_{=}^{*}(-\lambda) \ . \tag{5.81}$$

Proof. For any function ϕ we have $-\phi^{*}(-\lambda) = \inf_{\alpha\in\mathbb{V}} \quad \phi(\alpha) + \langle\lambda, \alpha\rangle$.

$$\begin{aligned} H(\lambda) &= \inf_{u\in\mathcal{U}} \ l(u) + \langle\lambda, f(u)\rangle \ , \\ &= \inf_{u\in\mathcal{U}} \inf_{\alpha\in\mathbb{V}} \ l(u) + \langle\lambda, \alpha\rangle \ , \\ &\qquad \text{s.t } \alpha = f(u) \ , \\ &= \inf_{\alpha\in\mathbb{V}} \inf_{u\in\mathcal{U}} \ l(u) + \langle\lambda, \alpha\rangle \ , \\ &\qquad \text{s.t } \alpha = f(u) \ , \\ &= \inf_{\alpha\in\mathbb{V}} \langle\lambda, \alpha\rangle + L_{=}(\alpha) = -L_{=}^{*}(-\lambda) \ . \end{aligned}$$

□

This makes it possible to apply a weak duality theorem to our original problem (5.71). Without further assumptions we can state the following lemma.

Lemma 47.

$$v \geqslant \sup_{\lambda\in\Lambda} \quad H(\lambda) - V^{*}(\lambda) \ . \tag{5.82}$$

Proof. We recall that $v = \inf_{\alpha\in\mathbb{V}} \ L_{=}(\alpha) + V(\alpha)$. l is proper so $L_{=}$ is proper as well. Applying twice Fenchel-Young inequality [117] we know, that for all $(\lambda, \alpha) \in \Lambda \times \mathbb{V}$,

- $L_{=}(\alpha) \geqslant -L_{=}^{*}(-\lambda) + \langle -\lambda, \alpha\rangle$,
- $V(\alpha) \geqslant -V^{*}(\lambda) + \langle\lambda, \alpha\rangle$.

Therefore, summing the inequalities, we obtain:

$$L_{=}(\alpha) + V(\alpha) \geqslant -L_{=}^{*}(-\lambda) - V^{*}(\lambda) = H(\lambda) - V^{*}(\lambda) \ . \tag{5.83}$$

This ends the proof. □

Proposition 48. *If $L_{=}$ and V are convex and one of the following condition holds*

- $0 \in ri(\ dom\,(L_{=}) - \ dom\,(V))$,
- *or the stronger* $dom\,(L_{=}) \cap cont(V) \neq \varnothing$,

then the following equality holds:

$$v = \sup_{\lambda\in\Lambda} \quad H(\lambda) - V^{*}(\lambda) \ . \tag{5.84}$$

Proof. We apply Fenchel duality theorem [117, 140]. □

Proposition 49. *Let $\lambda \in \Lambda$ such that the function $\langle\lambda, \cdot\rangle : \alpha \in \mathbb{V} \mapsto \langle\lambda, \alpha\rangle$ is non-decreasing. Then the following equality holds:*

$$H(\lambda) = -L_{\leqslant}^{*}(-\lambda) \ . \tag{5.85}$$

Proof. It is a direct application of Lemma 44 with $V = \langle\lambda, \cdot\rangle$. □

5.5.2 Proving monotonicity and linearity of a battery management problem

We show in this part that we can linearize a battery control problem with aging, which is useful to apply Model Predictive Control or SDDP.

We focus on the following problem where the decision variable $\boldsymbol{U}_t$ is the charge discharge of the battery at time t and $\boldsymbol{W}_t$ the uncertain net production of the grid connected to the battery. The objective is to minimize to consumption of power of the national grid, that is, the charge/discharge minus the net production. We take the positive part assuming that we cannot sell electricity to the grid. We call $x^+ = \max(0, x)$ the positive part of a variable x and $x^- = -\min(0, x)$ the negative part. We write the problem in an hazard-decision setting for the sake of simplicity; the results are the same in a decision-hazard setting. For the sake of simplicity as well, we assume that the noises $(\boldsymbol{W}_0, \dots, \boldsymbol{W}_T)$ are stagewise independent. It makes it possible to restrict the search to functions $\boldsymbol{U}_{t+1}(\boldsymbol{B}_t, \boldsymbol{H}_t, \boldsymbol{W}_{t+1})$ of the state and the next noise. The problem we study is the following:

$$\inf_{\boldsymbol{U}_{t=0,\dots,T-1}} \mathbb{E}\left[\sum_{t=0}^{T-1} c_t \times \Big(\boldsymbol{U}_{t+1} - \boldsymbol{W}_{t+1}\Big)\right]^+ , \tag{5.86a}$$

$$\text{s.t } \boldsymbol{B}_{t+1} = \boldsymbol{B}_t + \rho_c \boldsymbol{U}_{t+1}^+ - \rho_d^{-1} \boldsymbol{U}_{t+1}^- , \tag{5.86b}$$

$$\boldsymbol{H}_{t+1} = \boldsymbol{H}_t - \boldsymbol{U}_{t+1}^+ - \boldsymbol{U}_{t+1}^- , \tag{5.86c}$$

$$\underline{B} \leqslant \boldsymbol{B}_t \leqslant \overline{B} , \tag{5.86d}$$

$$\boldsymbol{H}_t \geqslant 0 , \tag{5.86e}$$

$$\underline{U} \leqslant \boldsymbol{U}_t \leqslant \overline{U} , \tag{5.86f}$$

$$\sigma(\boldsymbol{U}_t) \subset \sigma(\boldsymbol{B}_{t-1}, \boldsymbol{H}_{t-1}, \boldsymbol{W}_t) , \tag{5.86g}$$

$$\boldsymbol{B}_0 = b_0 , \quad \boldsymbol{H}_0 = h_0 . \tag{5.86h}$$

First, we prove that the value functions $V_t^\sharp$ of problem (5.86) are non increasing in state of charge b and health h. The value functions satisfy the following backward recursion

$$V_T^\sharp = 0 , \tag{5.87a}$$

$$V_t^\sharp(b, h) = \mathbb{E} V_t(b, h, \boldsymbol{W}), \ \forall (b, h) \in \mathbb{B} \times \mathbb{H} , \tag{5.87b}$$

where

$$V_t(b, h, w) = \inf_u \ c_t \times (u - w)^+ + V_{t+1}^\sharp(b + \rho_c u^+ - \rho_d^{-1} u^-, h - u^+ - u^-) , \tag{5.87c}$$

$$\text{s.t } \underline{B} - b \leqslant \rho_c u^+ - \rho_d^{-1} u^- \leqslant \overline{B} - b , \tag{5.87d}$$

$$u^+ + u^- \leqslant h , \tag{5.87e}$$

$$\underline{U} \leqslant u \leqslant \overline{U} . \tag{5.87f}$$

Lemma 50. *The value functions $\{V_t^\sharp\}_{t=0,\dots,T}$ are non-increasing.*

Proof. The last value function $V_T^\sharp$ is obviously non-increasing.

Assume that $V_{t+1}^\sharp$ is non increasing. V_t is obviously non increasing in h as decreasing h constrains the problem further and increases the objective as $V_{t+1}^\sharp$ is non increasing.

Let $b' \geqslant b$, let $\epsilon > 0$ and let u_b^ϵ an ϵ-optimal for $V_t(b,h,w)$. By definition, we have

$$c_t \times (u_b^\epsilon - w)^+ + V_{t+1}^\sharp(b + \rho_c u_b^{\epsilon+} - \rho_d^{-1} u_b^{\epsilon-}, h - u_b^{\epsilon+} - u_b^{\epsilon-}) \leqslant V_t(b,h,w) + \epsilon \; . \tag{5.88a}$$

We distinguish two cases.

$u_b^\epsilon \leqslant 0$: then u_b^ϵ is admissible for $V_t(b',h,w)$ because

$$\underline{B} - b' \leqslant \underline{B} - b \leqslant \rho_c u_b^{\epsilon+} - \rho_d^{-1} u_b^{\epsilon-} = \rho_d^{-1} u_b^\epsilon \leqslant 0 \leqslant \overline{B} - b' \; , \tag{5.88b}$$

moreover as V_{t+1} is non increasing

$$V_{t+1}^\sharp(b' + \rho_c u_b^{\epsilon+} - \rho_d^{-1} u_b^{\epsilon-}, h - u_b^{\epsilon+} - u_b^{\epsilon-}) \leqslant V_{t+1}^\sharp(b + \rho_c u_b^{\epsilon+} - \rho_d^{-1} u_b^{\epsilon-}, h - u_b^{\epsilon+} - u_b^{\epsilon-}) \; , \tag{5.88c}$$

then we have

$$V_t(b',h,w) \leqslant V_t(b,h,w) + \epsilon \; . \tag{5.88d}$$

$u_b^\epsilon > 0$: let $u_{b'}^\epsilon = \min(\rho_c^{-1} \times (\overline{B} - b'), u_b^\epsilon)$. $u_{b'}^\epsilon$ is admissible for $V_t(b',h,w)$ as

$$\underline{U} \leqslant 0 < u_{b'}^\epsilon \leqslant u_b^\epsilon \leqslant \overline{U} \; , \tag{5.88e}$$

$$\underline{B} - b' \leqslant 0 < \rho_c u_{b'}^\epsilon \leqslant \rho_c \rho_c^{-1} \times (\overline{B} - b') = B - b' \; , \tag{5.88f}$$

$$u_{b'}^{\epsilon\,+} \leqslant u_b^{\epsilon+} \leqslant h \; . \tag{5.88g}$$

Moreover we have

$$b' + \rho_c u_{b'}^\epsilon = b + \rho_c u_b^\epsilon \; , \tag{5.88h}$$

or

$$b' + \rho_c u_{b'}^\epsilon = \overline{B} \geqslant b + \rho_c u_b^\epsilon \; , \tag{5.88i}$$

so

$$b' + \rho_c u_{b'}^\epsilon \geqslant b + \rho_c u_b^\epsilon \; , \tag{5.88j}$$

so these inequalities and the fact that V_{t+1} in non increasing lead to

$$c_t \times (u_{b'}^\epsilon - w)^+ + V_{t+1}^\sharp(b + \rho_c u_{b'}^\epsilon, h - u_{b'}^\epsilon) \leqslant c_t \times (u_b^\epsilon - w)^+ + V_{t+1}^\sharp(b + \rho_c u_b^\epsilon, h - u_b^\epsilon) \tag{5.88k}$$

Finally

$$V_t(b',h,w) \leqslant V_t(b,h,w) + \epsilon \; , \tag{5.88l}$$

then we conclude that V_t is non increasing.

□

Now we would like to remove the positive and negative parts from the problem to apply SDDP or linear programming in a Model Predictive Control method.

Lemma 51. *The value functions* $\{V_t^\sharp\}_{t=0,\dots,T}$ *are convex polyhedral.*

Proof. $V_T^\sharp$ is trivially convex polyhedral. Assume that $V_{t+1}^\sharp$ is convex polyhedral. Let $\mathcal{B} : V_{t+1}^\sharp \mapsto V_t^\sharp = \mathbb{E}V_t$. We prove that this is a linear Bellman operator as it is demonstrated in [131] from [117] that in this case $V_t^\sharp$ is convex polyhedral. We introduce a new equivalent definition for V_t:

$$V_t(b,h,w) = \inf_{u_c,u_d,l} \; l + V_{t+1}^\sharp(b + \rho_c u_c - \rho_d^{-1} u_d, h - u_c - u_d) \;, \tag{5.89a}$$

$$\text{s.t } \underline{B} - b \leqslant \rho_c u_c - \rho_d^{-1} u_d \leqslant \overline{B} - b \;, \tag{5.89b}$$

$$u_c + u_d \leqslant h \;, \tag{5.89c}$$

$$0 \leqslant u_c \leqslant \overline{U}, \quad 0 \leqslant u_d \leqslant -\underline{U} \;, \tag{5.89d}$$

$$u_c \times u_d = 0 \;, \tag{5.89e}$$

$$l \geqslant 0 \;, \tag{5.89f}$$

$$l \geqslant c_t \times (u_c - u_d - w)^+ \;. \tag{5.89g}$$

In this new formulation, we introduce three non negative control variables l, u_c, u_d. The first non negative one l is used to linearize the objective by adding the constraint (5.89g). This is a classical trick. The other two (u_c, u_d) are used to replace the positive and negative parts on controls of the original problem but require to introduce the nonlinear (binary) constraint (5.89e). We show hereby that we can remove this binary constraint (5.89e).

Let (l, u_c, u_d) be an admissible solution to $V_t(b,h,w)$ without the binary constraint such that $u_c \times u_d > 0$. We distinguish two cases.

$u_c \leqslant u_d$ We introduce a new solution (l', u_c', u_d') such that $u_c' = 0$ and $u_d' = u_d - u_c$ with $l' = l \geqslant c_t \times (u_c' - u_d' - w)^+ = (u_c - u_d - w)^+$. This solution satisfies the binary constraint $u_c' \times u_d' = 0$. And this solution is admissible as

$$\begin{aligned}
&0 = u_c' \leqslant \overline{U}, 0 \leqslant u_d' \leqslant u_d \leqslant -\underline{U} \;, \\
&u_c' + u_d' = u_d - u_c \leqslant u_d + u_c \leqslant h \;, \\
&\text{and as } \rho_c \leqslant 1 \text{ and } \rho_d \leqslant 1 \;, \\
&\overline{B} - b \geqslant 0 \geqslant \rho_c u_c' - \rho_d^{-1} u_d' = \rho_d^{-1} u_c - \rho_d^{-1} u_d \geqslant \rho_c u_c - \rho_d^{-1} u_d \geqslant \underline{B} - b \;.
\end{aligned}$$

These inequalities plus the fact that $V_{t+1}^\sharp$ is non-increasing makes it possible to say that (l', u_c', u_d') is admissible and achieves the same cost.

$u_c > u_d$ We introduce a new solution (l', u_c', u_d') such that $u_c' = \min(u_c - u_d, \rho_c^- 1(\overline{B} - b))$ and $u_d' = 0$ with $l' \geqslant c_t \times (u_c' - u_d' - w)^+ \leqslant (u_c - u_d - w)^+$. So at optimality we have $l \leqslant (u_c - u_d - w)^+$. This solution satisfies the binary constraint $u_c' \times u_d' = 0$. And this solution is admissible as

$$\begin{aligned}
&0 \leqslant u_c' \leqslant u_c \overline{U}, 0 = u_d' \leqslant -\underline{U} \;, \\
&u_c' + u_d' = u_c - u_d \leqslant u_c + u_d \leqslant h \;, \\
&\overline{B} - b \geqslant \rho_c u_c' - \rho_d^{-1} u_d' \geqslant 0 \geqslant \underline{B} - b \;.
\end{aligned}$$

Moreover we have

$$b + \rho_c u_c' = b + \rho_c u_c - \rho_c u_d \geqslant b + \rho_c u_c - \rho_d^{-1} u_d \;,$$

or

$$b + \rho_c u_c = \overline{B} \geqslant b + \rho_c u_c - \rho_d^{-1} u_d \ ,$$

so as $V_{t+1}^\sharp$ is non increasing we have an admissible solution that achieves a better cost.

We conclude that, from any admissible solution without the binary constraint, we can build an admissible solution satisfying the binary constraint and achieving a lower cost. We prove recursively that this cost is strictly lower if $\rho_c < 1$ and $\rho_d < 1$. Hence, we can remove the binary constraint. The function V_t is therefore the value of a linear program where constraints are linear in the parameters b, h, w. Due to the linearity of the expectation, we conclude that $\mathcal{B}$ is a linear Bellman operator. □

Remark 52. *In the battery renewal problem, we show that the intraday problems are also non-increasing in the capacity c_d because a lower capacity constrains the problem further without changing the objective. We prove by backward induction that the daily value functions are decreasing because in the targets decomposition the instantaneous cost is decreasing and the value function as well. Moreover the problem does not have any constraint.*

Chapter 5. Bibliography

[114] R. Bellman. *Dynamic Programming*. Princeton University Press, New Jersey, 1957.

[115] D. P. Bertsekas. Convergence of discretization procedures in dynamic programming. *IEEE Transactions on Automatic Control*, 20(3):415–419, June 1975.

[116] D. P. Bertsekas. *Dynamic programming and optimal control. Vol. I*. Athena Scientific, Belmont, MA, fourth edition, 2017.

[117] J. Borwein and A. S. Lewis. *Convex analysis and nonlinear optimization: theory and examples*. Springer Science & Business Media, 2010.

[118] P. Carpentier, J.-P. Chancelier, G. Cohen, and M. De Lara. *Stochastic Multi-Stage Optimization. At the Crossroads between Discrete Time Stochastic Control and Stochastic Programming*. Springer-Verlag, Berlin, 2015.

[119] P. Carpentier, J.-P. Chancelier, M. De Lara, F. Pacaud, and T. Rigaut. A template to design online policies for multistage stochastic optimization problems. working paper, Jan. 2019.

[120] P. Carpentier, J.-P. Chancelier, M. De Lara, and T. Rigaut. Time blocks decomposition of multistage stochastic optimization problem. 2018.

[121] P. Carpentier, J.-P. Chancelier, V. Leclère, and F. Pacaud. Stochastic decomposition applied to large-scale hydro valleys management. *European Journal of Operational Research*, 270(3):1086 – 1098, 2018.

[122] J.-P. Chancelier and M. De Lara. Fenchel-moreau conjugation inequalities with three couplings and application to stochastic bellman equation. *arXiv preprint arXiv:1804.03034*, 2018.

[123] P. Haessig, H. B. Ahmed, and B. Multon. Energy storage control with aging limitation. In *PowerTech, 2015 IEEE Eindhoven*, pages 1–6. IEEE, 2015.

[124] P. Haessig, B. Multon, H. Ben Ahmed, S. Lascaud, and L. Jamy. Aging-aware NaS battery model in a stochastic wind-storage simulation framework. In *PowerTech 2013*, pages 1–6, Grenoble, France, June 2013.

[125] O. Hernández-Lerma and J. B. Lasserre. *Discrete-time Markov control processes: basic optimality criteria*, volume 30. Springer Science & Business Media, 2012.

[126] B. Heymann, J. Frédéric Bonnans, F. Silva, and G. Jimenez. A Stochastic Continuous Time Model for Microgrid Energy Management. In *ECC2016*, Aalborg, Denmark, June 2016.

[127] B. Heymann and P. Martinon. Optimal Battery Aging : an Adaptive Weights Dynamic Programming Algorithm. *Journal of Optimization Theory and Applications*, Aug. 2018.

[128] B. Heymann, P. Martinon, and F. Bonnans. Long term aging : an adaptative weights dynamic programming algorithm. working paper, July 2016.

[129] A. Iovine, S. B. Siad, G. Damm, E. D. Santis, and M. D. D. Benedetto. Nonlinear control of a dc microgrid for the integration of photovoltaic panels. *IEEE Transactions on Automation Science and Engineering*, 14(2):524–535, April 2017.

[130] R. Le Goff Latimier. *Management and Sizing of an Electric Vehicle Fleet Associated with a Photovoltaic Plant : Stochastic and Distributed Co-optimizationStationary Valorisation of Electric Vehicle Batteries taking into account their aging and availibility*. Theses, Université Paris-Saclay, Sept. 2016.

[131] V. Leclère, P. Carpentier, J.-P. Chancelier, A. Lenoir, and F. Pacaud. Exact converging bounds for stochastic dual dynamic programming via fenchel duality. 2018.

[132] D. E. Olivares, A. Mehrizi-Sani, A. H. Etemadi, C. A. Canizares, R. Iravani, M. Kazerani, A. H. Hajimiragha, O. Gomis-Bellmunt, M. Saeedifard, and R. e. a. Palma-Behnke. Trends in Microgrid Control. *IEEE Trans. Smart Grid*, 5(4):1905–1919, 2014.

[133] F. Pacaud, P. Carpentier, J.-P. Chancelier, and M. De Lara. Stochastic optimal control of a domestic microgrid equipped with solar panel and battery. preprint, Jan. 2018.

[134] M. V. Pereira and L. M. Pinto. Multi-stage stochastic optimization applied to energy planning. *Math. Program.*, 52(1-3):359–375, May 1991.

[135] A. Philpott, F. Wahid, and F. Bonnans. MIDAS: A Mixed Integer Dynamic Approximation Scheme. page 22, June 2016.

[136] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[137] T. Rigaut, P. Carpentier, J. Chancelier, M. D. Lara, and J. Waeytens. Stochastic optimization of braking energy storage and ventilation in a subway station. *IEEE Transactions on Power Systems*, pages 1–1, 2018.

[138] R. T. Rockafellar. Integrals which are convex functionals. *Pacific J. Math.*, 24:525–539, 1968.

[139] R. T. Rockafellar. Integrals which are convex functionals. II. *Pacific J. Math.*, 39:439–469, 1971.

[140] R. T. Rockafellar. *Convex analysis*. Princeton university press, 2015.

[141] R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

[142] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[143] J. Zou, S. Ahmed, and X. A. Sun. Stochastic dual dynamic integer programming. *Mathematical Programming*, Mar 2018.

Part III

Softwares and experimentations

Chapter 6

DynOpt: a generic library for stochastic dynamic optimization

This is a joint work with François Pacaud.

Chapter Abstract

We present in this chapter a stochastic dynamic optimization toolbox called DynOpt. A user interacts with DynOpt toolbox though an API which enables to build a stochastic optimization problem and solve it using dedicated algorithms. Moreover a user is able to build a specialized API for specific energy management problems he wants to solve. For instance, a user can build a battery sizing utility out of DynOpt or a house temperature controller as described in Chapter 7. This library is articulated around the concept of control policies that are distinguished by how they use online information to compute an optimal control for a stochastic dynamical system. We present the mathematical background that led to develop DynOpt. Then, we introduce the main objects for a potential user or a potential developer. Finally, we apply DynOpt on an energy storage toy problem and discuss the opportunities that it brings for real energy management applications. It is implemented as a Julia package, leveraging Julia multiple dispatch design.

Contents

6.1 Introduction and review

DynOpt.jl is a Julia [1] package developed at Efficacity [2] to solve multistage stochastic optimization problems. The aim of the package is to quickly design and solve stochastic optimization problems using a high level language without having to reimplement a dedicated control algorithm for every application.

DynOpt is implemented in Julia [147], a high level general purpose programming language that is more and more popular for scientific computing. This choice of Julia language is motivated by the active community in optimization and the existence of a high quality optimization modeler, JuMP [152]. Julia provides a just-in-time compiler particularly appealing for nested for loops that are required to solve stochastic dynamic programs (SDPs) [161, 145]. Moreover Julia is designed for parallelism which is appealing to solve SDPs.

We first introduce the mathematical background of DynOpt. Then, we present the different objects and associated algorithms which have been implemented. Afterwards, we present how to use the library for energy management oriented applications. Finally, we discuss the possible short term improvements of DynOpt.

Different packages already exist to solve SDPs. StochDynamicProgramming.jl [157], SDDP.jl [151], Kokako.jl [150], StOpt [153], BOCOP [148], StructDualDynProg [3], FAST [4]... DynOpt is close to StochDynamicProgramming.jl as it implements 2 of the same algorithms. However contrary to StochDynamicProgramming.jl its structure was thought to stay very close to the mathematical background in stochastic optimal control. Compared to the other packages, DynOpt implements more algorithms with a generic interface and is more stochastic optimal control oriented. The main choice of DynOpt is to remain generic for stochastic optimal control. It allows to easily express and solve these problems without an extensive understanding of stochastic optimization. The focus was not primarily on performances as most of DynOpt algorithms are to be run offline, where computation time is not critical. DynOpt outputs are however sufficiently efficient to compute controls for real systems in few milliseconds.

The structure of DynOpt is inspired by the work in suboptimal control and approximate dynamic programming of [145, 160, 161] and the second chapter of this PhD thesis.

[1] https://julialang.org
[2] https://www.efficacity.com
[3] https://github.com/JuliaStochOpt/StructDualDynProg.jl
[4] https://web.stanford.edu/ lcambier/fast/

6.2 Mathematical background: a template to design online policies

In this section we detail the kind of problems which can be solved using DynOpt as well as policies which is a notion of admissible solution. The problems we focus on display stochasticity, that is random variables. In the sequel, we write random variables in capital bold letters, like $\boldsymbol{Z}$, to distinguish them from deterministic variables z. .

6.2.1 Stochastic optimal control problems and solutions

We introduce a general Stochastic Optimal Control (SOC) problem in a risk neutral (expectation) setting following the formalism introduced in [149]

$$\min_{\boldsymbol{X},\boldsymbol{U}} \mathbb{E}\left[\sum_{t=0}^{T-1} L_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) + K(\boldsymbol{X}_T)\right], \tag{6.1a}$$

$$\text{s.t } \boldsymbol{X}_{t+1} = f_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) , \tag{6.1b}$$

$$B_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) \leqslant 0 , \tag{6.1c}$$

$$\sigma(\boldsymbol{U}_t) \subset \sigma(\boldsymbol{W}_0, \dots, \boldsymbol{W}_t) . \tag{6.1d}$$

These are the kind of problems that DynOpt is dedicated to solve. We detail in subsection 6.2.1 the different ingredients of this problem. Formally, these optimization problems are infinite dimensional and hard to solve, because solutions are random variables. In some very particular cases, it is possible to solve them exactly using numerical algorithms. In most real life cases this is however out of reach. DynOpt makes it possible to compute admissible, and sometimes optimal, solutions and assesses their quality.

Ingredients

We detail the ingredients appearing in Problem (6.1).

Time The index $t \in \mathbb{N}$ in (6.1) materializes a time stage. It belongs to a discrete set $\{0, \dots, T\}$ where $T \in \mathbb{N}$ is called the horizon of the problem.

Exogenous random variables The sequence of random variables $\left(\boldsymbol{W}_0, \dots, \boldsymbol{W}_T\right)$ is a discrete stochastic process. Each variable $\boldsymbol{W}_t$ is an *exogenous uncertainty* influencing the system to be controlled and takes values in a set $\mathbb{W}_t$.

State variables The sequence $\left(\boldsymbol{X}_0, \dots, \boldsymbol{X}_T\right)$ is a sequence of *state* variables describing the state of the system at each time step. It can be the level of water in a dam or the state of charge of a battery. They are random because they are influenced by the exogenous random variables $\boldsymbol{W}_t$, e.g rain in a dam. However, they are endogenous because they are also influenced by decisions $\boldsymbol{U}_t$ as described below. A variable $\boldsymbol{X}_t$ takes values in the set $\mathbb{X}_t$.

Decisions and non anticipativity constraint The sequence of random variables $\left(\boldsymbol{U}_0, \ldots, \boldsymbol{U}_{T-1}\right)$ is a sequence of *decision* variables, also named *controls*. A variable $\boldsymbol{U}_t$ is random because it depends on all the past uncertainties realization $w_0, \ldots, w_t$. The algebraic constraint (6.1d) states that $\boldsymbol{U}_t$ is measurable w.r.t $\boldsymbol{W}_0, \ldots, \boldsymbol{W}_t$. Applying Doob-Dynckin lemma [149] it can be equivalently restated in an functional manner:

$$(6.1d) \iff \exists \pi_t : \mathbb{W}_0 \times \ldots \times \mathbb{W}_t \to \mathbb{U}_t, \quad \boldsymbol{U}_t = \pi_t(\boldsymbol{W}_0, \ldots, \boldsymbol{W}_t) \,, \tag{6.2}$$

where $\boldsymbol{U}_t$ takes values in $\mathbb{U}_t$.

Dynamics, instantaneous and final costs For $t \in \{0, \ldots, T-1\}$ the mapping $f_t : \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1} \to \mathbb{X}_{t+1}$ is called the dynamics of the system. It describes how the system in a state x_t switches to the state x_{t+1} by applying a decision u_t and an uncertainty realization w_t. At the same time the function $L_t : \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1} \to (-\infty, +\infty]$ models the payoff obtained when switching from x_t to x_{t+1}. The function $K : \mathbb{X}_T \to (-\infty, +\infty]$ is a cost associated to the final state of the system.

Constraints For $t \in \{0, \ldots, T-1\}$ the mapping $B_t : \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1} \to \mathbb{R}^{n_t^c}$ defines the n_t^c constraints of the problem at time t. It returns a vector in $\mathbb{R}^{n_t^c}$ that has to remain below 0 elementwise.

Admissible solutions

As stated previously the admissible decisions of Problem (6.1) satisfy the non-anticipativity constraint (6.1d) or its functional counter part (6.2). Hence we are looking for solutions that have the form $\{\pi_0, \ldots, \pi_{T-1}\}$ with:

$$\forall t \in \{0, \ldots, T-1\}, \quad \pi_t : \mathbb{W}_0 \times \ldots \times \mathbb{W}_t \to \mathbb{U}_t \,. \tag{6.3}$$

The mapping π_t is called a *policy*, more precisely a fully *noise dependent policy*.

In pratice, it is often out of reach to compute a fully noise dependent policy due to the size of the cartesian space $\mathbb{W}_0 \times \ldots \times \mathbb{W}_t$.

Therefore, we restrict the search to solutions among the class of *noises dependent state feedback* policies of the form

$$\pi_t : \mathbb{X}_t \times \mathbb{W}_{t_0(t)} \times \ldots \times \mathbb{W}_t \to \mathbb{U}_t \,; \tag{6.4}$$

where $t_0(t) \in \{0, \ldots, t\}$. That kind of policies takes into account only some past uncertainties as well as the state of the system. This is indeed a restriction, as the state $\boldsymbol{X}_t$ is, by the iterated dynamics (6.1b), a function of $(\boldsymbol{W}_0, \ldots, \boldsymbol{W}_t)$.

In fact when the exogenous uncertainties $\{\boldsymbol{W}_0, \ldots, \boldsymbol{W}_T\}$ are stagewise independent it is enough to restrict the search to state feedbacks $\pi_t : \mathbb{X}_t \to \mathbb{U}_t$. However in many cases this so-called Markovian property is not ensured.

In practice, we are not interested in knowing $\pi_t(x_t, w_{t_0(t)}, \ldots, w_t)$ for all possible values of $(x_t, w_{t_0(t)}, \ldots, w_t)$; we just want to be able to compute, on the fly, the value $u_t = \pi_t(x_t, w_{t_0(t)}, \ldots, w_t)$ when, at time t, the tuple $(x_t, w_{t_0(t)}, \ldots, w_t)$ materializes.

This is why, in subsection 6.2.2 and subsection 6.2.3, we present two classes of methods for the online implementation of strategies. Both classes compute $u_t = \pi_t(x_t, w_{t_0(t)}, \dots, w_t)$ by solving, online, an optimization problem.

We refer to Chapter 1 of this manuscript for a more formal description of the concept of admissible solution.

Hazard decision policies

In some cases the non-anticipativity constraint (6.1d) is relaxed to take into account one more uncertainty. Hazard-Decision solutions have the following form

$$\pi_t : \mathbb{X}_t \times \mathbb{W}_{t_0(t)} \times \ldots \times \mathbb{W}_t \times \mathbb{W}_{t+1} \to \mathbb{U}_t \ . \tag{6.5}$$

These are useful to describe decisions that can react immediately to uncertainty realization or to linearize a dynamical equation.

6.2.2 Cost-to-go policies

Cost to go based policies proceed in two parts.

- Offline, the cost-to-go based algorithms compute a sequence of functions $\widetilde{V}_t$ by backward induction as follows:

$$\forall x \in \mathbb{X}_T, \widetilde{V}_T(x) = K(x) \tag{6.6a}$$

$$\forall x \in \mathbb{X}_t, \widetilde{V}_t(x) = \min_{u \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \Big[L_t(x, u, w_{t+1}) + \widetilde{V}_{t+1}\Big(f_t(x, u, w_{t+1})\Big)\Big] \mu^{of}_{t+1}(dw_{t+1}) \ . \tag{6.6b}$$

  Here, each μ^{of}_{t+1} is an (offline) probability distribution on the set $\mathbb{W}_{t+1}$. The functions $\widetilde{V}_t$ are called costs-to-go or value functions in the sequel.

- Online, at time t, the computation of a cost-to-go policy uses the functions $\widetilde{V}_t$ and solves

$$u_t \in \arg\min_{u \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \Big[L_t(x, u, w_{t+1}) + \widetilde{V}_{t+1}\Big(f_t(x, u, w_{t+1})\Big)\Big] \mu^{on}_{t+1}(w_{t_0(t)}, \dots, w_t, dw_{t+1}) \ . \tag{6.7}$$

  Here, μ^{on}_{t+1} is an (online) conditional probability distribution on the set $\mathbb{W}_{t+1}$, knowing the previous uncertainties $w_{t_0(t)}, \dots, w_t$.

  As an example we choose a conditional distribution depending only on the last uncertainty realization when we use an order 1 auto-regressive model.

Remark 53. *In the markovian case where the uncertainties are stagewise independent, the online distribution is the same as the offline one, with no past uncertainties dependence. Bellman's principle of optimality guarantees that it produces an optimal policy.*

6.2.3 Lookahead policies

A lookahead policy method has most often no offline computation part. The most general form of lookahead policies, with discretized uncertainties, is called Stochastic Programming, we refer the reader to [163] for a general definition. Currently in DynOpt only a subclass of lookahead policies, which is called open loop feedback control [144], is implemented.

At time step t', a lookahead policy takes as inputs the state x of the system and some previous uncertainties realizations $w_{t_0(t')}, \dots, w_{t'}$. One way or another, it selects a number S of "scenarios" $(\widetilde{w}^s_{t'+1}, \dots, \widetilde{w}^s_T)_{s\in\{1,\dots,S\}}$ with associated probabilities $(p_s)_{s\in\{1,\dots,S\}}$ and then solves the following deterministic optimal control problem:

$$\min_{(u_{t'},\dots,u_{T-1})} \sum_{s=1}^{S} p_s \sum_{t=t'}^{T-1} L_t(x^s_t, u_t, \widetilde{w}^s_{t+1}) + K(x^s_T) \ , \tag{6.8a}$$

$$\text{s.t } x^s_{t+1} = f_t(x^s_t, u_t, \widetilde{w}^s_{t+1}) \ , \tag{6.8b}$$

$$B_t(x^s_t, u_t, w^s_{t+1}) \leqslant 0 \ , \tag{6.8c}$$

$$x^s_{t'} = x \ . \tag{6.8d}$$

From the optimal controls $(u_{t'}, \dots, u_{T-1})$ thus obtained, the lookahead policy only keeps the first $(u_{t'}, \dots, u_{t+N_{lp}})$ (we call N_{lp} the *reoptimization step* of the lookahead policy). Then, at time $t' + N_{lp}$, the OLFC algorithm produces new controls by solving problem (6.8) starting at $t' + N_{lp}$ with updated noises scenarios.

6.3 Modeling language and algorithms

We present in the section the different objects and functionalities implemented in DynOpt. We describe as well three different algorithms to compute costs-to-go for cost-to-go policies and how to implement a lookahead policy.

6.3.1 Formulation of a problem in DynOpt

We present hereby the core objects of DynOpt to define a stochastic optimization problem as well as their associated resolution algorithms and policies. First we introduce the open source dependencies required to run DynOpt.

Underlying technologies

We list here all the dependencies of DynOpt, i.e Julia packages, which are all open source.

- JuMP [5] and MathProgBase [6]: some algorithms require to solve mathematical programs. For this reason one of the main dependence of DynOpt is JuMP, a Julia mathematical programming modeler similar to AMPL, Pyomo or GAMS. A JuMP user can express and solve many kinds of optimization problems as long as a dedicated solver is installed on the machine. JuMP is built upon MathProgBase, an abstract layer to interface with solvers.

[5] https://github.com/JuliaOpt/JuMP.jl
[6] https://github.com/JuliaOpt/MathProgBase.jl

- Interpolations [7] some algorithms require interpolations of arrays, for example when a function is only computed on a grid while we need to evaluate it in a continuous set.

- StatsBase [8]: stochastic optimization requires tools to handle random variables. StatsBase provides basic supports for statistics in Julia.

- Clustering [9]: to handle random variables numerically in DynOpt they are regularly quantized. Clustering provides a k-means algorithm used to quantize the discrete random variables provided by the user.

- ProgressMeter [10]: some algorithms have a deterministic number of iterations. ProgressMeter is used to display a progress bar for such algorithms.

- CutPruners [11]: SDDP algorithm presented in paragraph 6.3.2, models costs-to-go as polyhedral functions. Cut pruning allows to remove cuts that are not *active* from polyhedral functions.

- NearestNeighbors [12] and Distances [13] some algorithms require to find the nearest neighbor to a point in a list in a list of points. NearestNeighbors and Distances allows to perform efficiently this task using k-dimensional trees.

DynamicOptimizationModel object

The main structure or type of DynOpt is a DynamicOptimizationModel. It contains all the features required to define a stochastic optimal control problem as detailed in subsection 6.2.1.

[7] https://github.com/JuliaMath/Interpolations.jl
[8] https://github.com/JuliaStats/StatsBase.jl
[9] https://github.com/JuliaStats/Clustering.jl
[10] https://github.com/timholy/ProgressMeter.jl
[11] https://github.com/JuliaPolyhedra/CutPruners.jl
[12] https://github.com/KristofferC/NearestNeighbors.jl
[13] https://github.com/JuliaStats/Distances.jl

```
mutable struct DynamicOptimizationModel{T}

    stages::Int #Number of decision stages

    #cost function L_t(x_t, u_t, w_{t+1})
    cost::Union{Function, Void}

    #dynamics function that returns the new state f_t( x_t, u_t, w_{t+1})
    dynamics::Union{Function, Void}

    #dynamics function that changes the new state in place
    dynamics!::Union{Function, Void}

    #constraints function returning a boolean
    constraints::Union{Function, Void}

    #constraints function returning a vector that should be lower than 0
    constraints_vector::Union{Function, Void}

    final_cost::Union{Function, Void} #final cost function K(x_T)

    x_mins::Array{T,1} #lower bounds for states
    x_maxs::Array{T,1} #upper bounds for states

    u_mins::Array{T,1} #lower bounds for controls
    u_maxs::Array{T,1} #upper bounds for controls
    u_type::Array{Symbol,1} #types of controls Cont or Bin

    #anticipativity of the controls, WaitAndSee or HereAndNow
    non_anticipativity::Array{DataType,1}

    #possible initial states of the problem
    initial_states::Union{Void, Array{Float64}}
end
```

Figure 6.1: DynamicOptimizationModel type

Stochastic processes types

A stochastic process is an abstract type in DynOpt, naturally called StochasticProcess. There are two kinds of stochastic processes that can be modeled in DynOpt. Both are structures that inherit from the StochasticProcess abstract type.

```
# A discrete law with N realizations of dimension W
struct DiscreteMarginalLaw{T}

    support::Array{T,2} # W x N support vector of the discrete probability law

    probas::StatsBase.ProbabilityWeights # N probalities

end

struct WhiteNoise <: StochasticProcess

    A vector a indepent discrete laws
    laws::Vector{DiscreteMarginalLaw}

end

struct FunctionalProcess <: StochasticProcess

    # A vector of functions returning a DiscreteMarginalLaw
    laws::Vector{Function}

    # A law at time t takes as argument
    # all the past realizations up to time t-maximum_lag
    maximum_lag::Int

end
```

Figure 6.2: Types to handle stochastic processes

A stochastic process is used when building a cost-to-go solver that requires a WhiteNoise, as cost-to-go methods require a Markovian assumption. Both WhiteNoise and FunctionalProcess can be used to build a policy as a policy can take into account stagewise dependence between random variables.

6.3.2 Computing a cost-to-go policy

To compute the family of costs-to-go $\{\widetilde{V}_t\}_{t=0,\dots,T}$ cost-to-go policies require to perform the backward recursion (6.6).

Numerically, the distributions μ_t^{of} are quantized one way or another. We call n_t the size of the quantification of μ_t^{of}, $\mathbb{W}_t^d = \{w_t^1, \dots, w_t^{n_t}\}$ the realizations of the uncertainty obtained at time t and $p_t^1, \dots, p_t^{n_t}$ the associated probabilities.

A numerical cost-to-go algorithm then solves the following backward recursion:

$$\forall x \in \mathbb{X}_T, \widetilde{V}_T(x) = K(x) \ , \tag{6.9a}$$

$$\forall x \in \mathbb{X}_t, \widetilde{V}_t(x) = \min_{u \in \mathbb{U}_t} \sum_{i=1}^{n_{t+1}} p_{t+1}^i \Big[L_t(x, u, w_{t+1}^i) + \widetilde{V}_{t+1}\Big(f_t(x, u, w_{t+1}^i) \Big) \Big] \ . \tag{6.9b}$$

Three challenges remain for the algorithm to be numerically tractable

1. the exploration of the state space $\mathbb{X}_t$ at each time step t,
2. the resolution of problem (6.9b) at each time step t,
3. the choice of function class to compute a cost-to-go $\widetilde{V}_t$ at each time step t.

We first distinguish the four cost-to-go algorithms implemented in DynOpt by the exploration method of the state space.

State space exhaustive search

The most classical implementation of cost–to-go algorithms perform an exhaustive search in the discretized spaces $\mathbb{X}_t$. Most often $\mathbb{X}_t$ is replaced by a grid $\mathbb{X}_t^d \subset \mathbb{X}_t$.

Nested for loops We present hereby the classical Stochastic Dynamic Programming algorithm [149, 161]. Let $t \in \{0, \dots, T-1\}$ and assume that we have computed $\forall x \in \mathbb{X}_{t+1}^d, \widetilde{V}_{t+1}(x_d)$. The algorithm discretizes the control space $\mathbb{U}_t$ using a grid $\mathbb{U}_t^d$ and solves the optimization problem (6.9b) for all $x \in \mathbb{X}_t^d$ by exhaustive search giving an approximate value for $\widetilde{V}_t(x)$. For every $u \in \mathbb{U}_t^d$ and every w_{t+1}^i for $i \in \{1, \dots, n_{t+1}\}$ the next state $f_t(x, u, w_{t+1}^i)$ is not guaranteed to fall on a grid point of $\mathbb{X}_{t+1}^d$. For this reason an interpolation of the finite sequence $\{\widetilde{V}_{t+1}(x)\}_{x \in \mathbb{X}_{t+1}^d}$ is performed over $\mathbb{X}_{t+1}$. In DynOpt we use the Julia package Interpolations.jl to perform this interpolation automatically.

This algorithm falls down to discretize state, control and uncertainty spaces and write four nested for loops, over time (backward), over states, controls and uncertainties. Assuming that $\mathbb{X}_t^d$ is a Cartesian product of N_X discrete spaces with X values, and likewise for $\mathbb{U}_t^d$ and $\mathbb{W}_t^d$ then the complexity of the algorithm is $O(T \times X^{N_x} \times U^{N_u} \times W^{N_w})$.

This algorithm displays three *curses of dimensionality* over the number of state N_x, control N_u and uncertainty N_w variables. We usually assume that the algorithm is not tractable above three or four state variables. Even if the loop over state variables is parallelized, which is the case in DynOpt.

Inner approximation The inner approximation stochastic dynamic programming algorithms is similar to the nested for loops algorithm except in the way the optimization problem (6.9b) is solved. It can be applied when dynamics and constraints are linear and the instantaneous and final costs are convex. It relies on manipulation of polyhedral approximation in convex optimization [146].

At time $t \in \{0, \dots, T-1\}$ the algorithm computes the convex hull of the points $\{\widetilde{V}_{t+1}(x_{t+1}^i)\}_{i \in \{1, \dots, n_{t+1}^x\}}$ where n_{t+1}^x is the cardinal of $\mathbb{X}_{t+1}^d$. That is $\widetilde{V}_{t+1}$ is replaced by

$$x \in \mathbb{X}_{t+1} \mapsto \min_{\alpha_1, \dots, \alpha_{n_{t+1}^x}} \quad \sum_{i=1}^{n_{t+1}^x} \alpha_i \widetilde{V}_{t+1}(x_{t+1}^i) \ , \tag{6.10}$$

$$\text{s.t} \quad \sum_{i=1}^{n_{t+1}^x} \alpha_i x_{t+1}^i = x \ , \tag{6.11}$$

$$\sum_{i=1}^{n_{t+1}^x} \alpha_i = 1 \tag{6.12}$$

$$0 \leqslant \alpha_i \leqslant 1 \ . \tag{6.13}$$

Using this trick the optimization problem (6.9b) can be solved (approximately) with convex programming techniques, hence removing the control and uncertainties curses of dimensionality. However the state curse of dimensionality remains.

State space forward backward exploration

Forward-backward algorithms explore the state spaces differently. Instead of an exhaustive search among the state spaces they explore only *relevant* state, that is states that are often explored by the controlled system dynamics.

Forward-backward Scheme Forward-backward algorithms improve iteratively the costs-to-go functions $\{\widetilde{V}_t\}_{t=0,\dots,T}$.

Initialization: At a first iteration an initial value is given to $\{\widetilde{V}_t^0\}_{t=0,\dots,T}$. For instance, we can choose a lower bound like 0 if all the costs are non negative.

Iteration k: Assume that a sequence of costs $\{\widetilde{V}_t^{k-1}\}_{t=0,\dots,T}$ have been computed at iteration $k-1$. We build a cost-go-go policy $\pi_0^{k-1}, \dots, \pi_{T-1}^{k-1}$ out of this sequence such that:

$$\forall x \in \mathbb{X}_t, \pi_t^{k-1}(x) \in \underset{u \in \mathbb{U}_t}{\arg\min} \sum_{i=1}^{n_{t+1}} p_{t+1}^i \Big[L_t(x, u, w_{t+1}^i) + \widetilde{V}_{t+1}^{k-1}\Big(f_t(x, u, w_{t+1}^i)\Big)\Big] . \tag{6.14}$$

An initial state x_0^k and a scenario $w_0^k, \dots, w_T^k$ are drawn. Then a trajectory of states $x_1^k, \dots, x_T^k$ is generated by plugging the policy $\pi_0^{k-1}, \dots, \pi_{T-1}^{k-1}$ in the state dynamics.

$$x_t^k = f_t(x_{t-1}^k, \pi_t^{k-1}(x_{t-1}^k), w_{t+1}^k) . \tag{6.15}$$

A new sequence of costs-to-go $\{\widetilde{V}_t^k\}_{t=0,\dots,T}$ is obtained by solving (6.6) along the new explored states.

DynOpt implements this forward-backward scheme in an abstract manner, defining an abstract type ForwardBackwardSolver. Each Julia structure that inherits from ForwardBackwardSolver has to define an approximation object for the costs to go, that is a class of functions, and specific parameters. Then the following Julia functions have to be specified for each kind of forward backward algorithms:

```
# gives a initial value to value functions
function initialize_value_functions end

# returns an approximation object out of a solved JuMP model
function get_approximation_object end

# eventually removes non relevent state trajectories
function select_trajectories! end

# updates the costs to go
function update_cost_to_go! end

# eventually simplifies value functions approximation
function prune! end
```

Figure 6.3: Abstraction layer of forward-backward algorithms

That kind of algorithm leverage mathematical programming techniques to solve (6.9b). Hence, a function $\widetilde{V}_{t+1}^k$ is defined as an approximation object and the corresponding policy π_t^k as a JuMP model. We detail here two ways to achieve this part in a convex case and in a monotonic case.

Stochastic Dual Dynamic Programming (SDDP) SDDP [158] can be applied in the same case as inner approximation presented in subsection 6.3.2. That is, the dynamics and constraints have to be linear and the costs convex. In this case we look for $\{\widetilde{V}_t^k\}$ in the class of convex polyhedral functions. That kind of functions can be evaluated on a point by solving a linear program:

$$x \in \mathbb{X}_t \mapsto \min_{\theta} \quad \theta \tag{6.16}$$

$$\text{s.t} \quad \theta \geqslant \lambda_t^i \cdot x + \beta_t^i, \forall i \in \{1, \dots, c_t^k\} \ , \tag{6.17}$$

where c_t^k is the number of cuts of $\widetilde{V}_t^k$, $\lambda_t^1, \dots, \lambda_t^{c_t^k}$ the slopes and $\beta_t^i, \dots, \beta_t^{c_t^k}$ the constant terms of the cuts.

The main specificity of SDDP is to update the function V_t^{k-1} at iteration k using dual multipliers. Using current cuts of V_{t+1}^k and explored state x_t^k it solves the following problem using a convex programming solver for any possible w_{t+1}^i:

$$l_t^k(w_{t+1}^i) = \min_{\theta, u \in \mathbb{U}_t, x \in \mathbb{X}_t} \Big[L_t(x, u, w_{t+1}^i) + \theta \Big] \ , \tag{6.18a}$$

$$\text{s.t } x = x_t^k \tag{6.18b}$$

$$\theta \geqslant \lambda_{t+1}^i \cdot f_t(x, u, w_{t+1}^i) + \beta_{t+1}^i, \forall i \in \{1, \dots, c_{t+1}^k\} \ . \tag{6.18c}$$

c_t^k is updated to $c_t^{k-1} + 1$. Then the dual multiplier $\lambda_t^{c_t^k}(w_{t+1}^i)$ is extracted by taking the dual variable of constraint (6.18b) for each realization w_{t+1}^i and taking the expectation we obtain the new slope $\lambda_t^{c_t^k} = \sum_{i=1}^{n_{t+1}} p_{t+1}^i \lambda_t^{c_t^k}(w_{t+1}^i)$. Finally the new constant term is $\beta_t^{c_t^k} = \sum_{i=1}^{n_{t+1}} p_{t+1}^i l_t^k(w_{t+1}^i) - \lambda_t^{c_t^k} \cdot x_t^k$. This cut $(\lambda_t^{c_t^k}, \beta_t^{c_t^k})$ is added to $\widetilde{V}_t^{k-1}$ to produce $\widetilde{V}_t^k$:

$$V_t^k(x) = \max\{V_t^{k-1}(x), \ \lambda_t^{c_t^k} \cdot x + \beta_t^{c_t^k}\} \ . \tag{6.19}$$

We might think that $c_t^k = k$ due to the update $c_t^k = c_t^{k-1} + 1$; however it might be efficient to remove inactive cuts regularly to speed up the resolution of (6.18).

Mixed Integer Dynamic Approximation Scheme (MIDAS) MIDAS [159] is a recent algorithm to theoretically solve all kinds of multistage stochastic optimization problems as long as the value functions are monotonic. MIDAS models value functions as *monotonic step functions*. Such functions have the following Mixed Integer Programming representation:

$$x \in \mathbb{X}_t \mapsto \min_{\theta, \omega, z} \quad \theta \tag{6.20}$$

$$\text{s.t} \quad \theta \geqslant q_t^i + (\underline{V}_t - q_t^i) \times (1 - \omega), \forall i \in \{1, \dots, s_t^k\} \ , \tag{6.21}$$

$$x \geqslant (x_t^i - \underline{x}_t) \cdot z + \delta + \underline{x}_t, \forall i \in \{1, \dots, s_t^k\} \ , \tag{6.22}$$

$$1 - \omega = \sum_{i=1}^{N_X} z_i \ , \tag{6.23}$$

$$\omega \in \{0, 1\}, \quad z \in \mathbb{R}^{N_x} \ , \tag{6.24}$$

where s_t^k is the number of steps of $\widetilde{V}_t^k$, $x_t^1, \dots, x_t^{s_t^k}$ the breakpoints and $q_t^i, \dots, q_t^{c_t^k}$ the levels of the steps. $\underline{x}_t$ is the lower bound of $\mathbb{X}_t$, $\underline{V}_t$ a lower bound of V_t^k and δ is an arbitrary positive real number that should be chosen small. The major specificity of MIDAS is to update the function V_t^{k-1} at iteration k by adding a step using the value of problem (6.25). Using current steps of V_{t+1}^k and explored state x_t^k it solves the following problem using a mixed integer programming solver for any possible w_{t+1}^i:

$$m_t^k(w_{t+1}^i) = \min_{u \in \mathbb{U}_t, \theta, \omega, z} \left[L_t(x_t^k, u, w_{t+1}^i) + \theta \right], \tag{6.25a}$$

$$\text{s.t} \quad \theta \geqslant q_{t+1}^i + (\underline{V}_{t+1} - q_{t+1}^i) \times (1 - \omega), \forall i \in \{1, \dots, s_{t+1}^k\}, \tag{6.25b}$$

$$f_t(x_t^k, u, w_{t+1}^i) \geqslant (x_{t+1}^i - \underline{x}_{t+1}) \cdot z + \delta + \underline{x}_{t+1}, \forall i \in \{1, \dots, s_{t+1}^k\}, \tag{6.25c}$$

$$1 - \omega = \sum_{i=1}^{N_X} z_i, \tag{6.25d}$$

$$\omega \in \{0, 1\}, \quad z \in \mathbb{R}^{N_x}. \tag{6.25e}$$

s_t^k is updated to $s_t^{k-1} + 1$. Finally the level is $q_t^{s_t^k} = \sum_{i=1}^{n_{t+1}} p_{t+1}^i m_t^k(w_{t+1}^i)$ is added to $\widetilde{V}_t^{k-1}$ at point x_t^k to produce $\widetilde{V}_t^k$.

As for SDDP we might think that $s_t^k = k$ due to the update $s_t^k = s_t^{k-1} + 1$ however if x_t^k is close enough (with a user tuned parameter) to a previous point in $x_t^0, \dots, x_t^{k-1}$, then the closest point level is updated and no new breakpoint is added.

MIDAS is an experimental algorithm in DynOpt. It is currently not efficient to solve problems with more than 10 time steps and does not alleviate the curse of dimensionality. We propose here-under a small trick to speed up MIDAS convergence for stochastic mixed integer convex programs.

Hotstarting MIDAS with SDDP When the continuous relaxation of stochastic optimization problem with monotonous value functions has linear dynamics and constraints and convex costs, it is possible to hotstart MIDAS with SDDP cuts. The idea is to apply SDDP to the continuous relaxation, then keep the cuts to initialize the value functions for MIDAS. As we are minimizing, the value functions of the continuous relaxation are guaranteed to remain below the true value functions of the problem, hence it gives a lower bound that MIDAS is able to improve for the original problem. Our numerical results on a toy example show promising results to speed up the convergence of MIDAS.

Stopping criterion Currently, a simple stopping criterion is implemented in DynOpt. It is a fixed number of forward-backward iterations. However there are multiple stopping criterion for SDDP that could be implemented and added to DynOpt [164, 156].

Solver types

We present the four types of cost-to-go solvers implemented in DynOpt. Comments are provided directly in the code.

```
struct GridDPSolver <: AbstractSolver

    X::DiscreteSpace # A discretization of the state space

    U::DiscreteSpace # A discretization of the control space

    W::WhiteNoise # Discrete random variables stagewise independent

    # Parallelism is set to True if the loop over states is parallelised
    # Requires to define the problem in an @everywhere environment
    parallelism::Bool
end
```

Figure 6.4: Exhaustive search solver type

```
struct LinearDP <: AbstractSolver

    X::DiscreteSpace # A discretization of the state space

    W::WhiteNoise # Discrete random variables stagewise independent

    # A linear programming solver e.g CplexSolver() or ClpSolver()
    lpsolver::MathProgBase.AbstractMathProgSolver
end
```

Figure 6.5: Inner approximation solver type

```
mutable struct SDDPSolver <: ForwardBackwardSolver


    max_iter::Int # A maximum number of iterations

    max_cuts::Int # A maximum number of cuts, at least max_iter+1

    n_pass::Int # A number of forward passes at each iteration

    # A linear programming solver e.g CplexSolver() or ClpSolver()
    lp_solver::MathProgBase.AbstractMathProgSolver

    W::WhiteNoise # Discrete random variables stagewise independent

    # A number of epsilon greedy passes,
    # if above 0 the first n_greedy_passes forward passes
    # select random controls with probability epsilon
    # instead of optimal controls
    n_greedy_passes::Int
    epsilon::Float64

    # Every n_pruning passes cut pruning is performed, if set to 0 no pruning
    n_pruning::Int

    # A polyhedral representation of K
    polyhedral_final_cost::Union{PolyhedralFunction, Void}

end
```

Figure 6.6: SDDP solver type

```
struct MIDASSolver <: ForwardBackwardSolver

    max_iter::Int # A maximum number of iterations

    max_steps::Int # A maximum number of steps, at least max_iter+1

    n_pass::Int # A number of forward passes at each iteration

    delta::Array{Float64,1} # The minimum distance to add a new point

    milp_solver # A MILP solver

    W::WhiteNoise # Discrete random variables stagewise independent

    # A number of greedy passes, if > 0 the first n_greedy_passes forward passes
    # select random controls with probability epsilon instead of optimal controls
    greedy_passes::Int
    epsilon::Float64

    # Every n_pruning passes cut pruning is performed, if set to 0 no pruning
    n_pruning::Int

    # If true value functions hostarted with SDDP cuts of continuous relaxation
    sddp_hotstart::Bool

    # An SDDP solver to compute the value functions of continuous relaxation
    sddp_solver::AbstractSolver

end
```

Figure 6.7: MIDAS solver type

6.3.3 Computing a Model Predictive Control (MPC) policy

DynOpt provides functions to build automatically a Model Predictive Control [145] policy by building a JuMP model out of a DynamicOptimizationModel and MPC specific parameters. It requires, notably, to define the MPC rolling horizon, the step of re-optimization as well as a function to build scenarios of future uncertainties based on current state and previous uncertainties realization.

It could be more efficient to let the user define himself the deterministic optimization problem MPC requires to solve. The JuMP model is generated automatically from the cost, dynamics and constraints functions. A user defined deterministic problem could use more cleverly Mathematical Programming techniques such as disciplined convex programming or linear programming reformulations. For instance a problem to be solved by dynamic programming can be written with non-linear dynamics while an equivalent linear programming formulation is available introducing new auxiliary decision variables. We present such a case in subsection 6.4.1.

6.3.4 Assessment of a policy

To assess a policy it must be simulated along multiple scenarios of uncertainties realizations over the whole time horizon T of the considered problem.

The generic simulation algorithm of a given policy $\pi = \{\pi_0, \ldots, \pi_{T-1}\}$ with $\pi_t : \mathbb{W}_0 \times \ldots \times \mathbb{W}_t \to \mathbb{U}_t$ requires to draw $S \in \mathbb{N}^*$ scenarios $\{w_t^s\}_{t=0;\ldots,T, \quad s=1,\ldots,S}$. Ideally these

scenarios should be different to the ones used to build the policy π so as to realize an *out-of-sample* assessment of the policy. Otherwise the results could be biased.

Algorithm 5: Simulation algorithm

Data: A DynamicOptimizationProblem m, a policy π, scenarios $\{w_t^s\}_{t=0;\dots,T}^{s=1,\dots,S}$, an initial state $x_0 \in \mathbb{X}_0$

Result: A vector of costs L, an array of states X, an array of controls U

for $s = 1, \dots, S$ **do**

  $L^s = 0, \quad x_0^s = x_0$. **for** $t = 0, \dots, T-1$ **do**

    $u_t^s = \pi_(w_0^s, \dots, w_t^s)$,

    $L^s += L_t(x_t^s, u_t^s, w_{t+1}^s)$,

    $x_{t+1}^s = f_t(x_t^s, u_t^s, w_{t+1}^s)$.

  end

  $L^s += K(x_T^s)$.

end

return $L = \{L^s\}_{s=1,\dots,S}, X = \{x_t^s\}_{t=0,\dots,T}^{s=1,\dots,S}, U = \{u_t^s\}_{t=0,\dots,T-1}^{s=1,\dots,S}$.

Then, taking the mean value of the vector L we obtain a Monte Carlo estimation of the expectation of rewards using the policy π. To compare policies for stochastic optimal control problems it is mandatory to compare distributions of the costs L so as to state that a policy is better than another one $x\%$ of the time. Finally it can be useful to plot the states and controls along different scenarios to understand the policy, track mistakes and compare policies on criteria other than expected reward.

Simulation function

DynOpt provides functions to simulate different kinds of policies. It uses Julia type disptach design to select the right simulation function for a given policy. A simulation function prototype is as follows.

```
function simulation(m::DynamicOptimizationModel, Π::T,
                    scenario::Array{Float64,3}, x0::Vector{Float64})
                    where T <: Policy
                    ...
end
```

Figure 6.8: Abstract prototype of a simulation function for a policy

It takes as input a DynamicOptimizationModel, a policy, a set of scenarios for the uncertainties and an initial state. It returns the costs obtained by simulating the policy along each scenarios as well as the corresponding sequence of states and controls.

Policies type

We present here the different types of policies implemented in DynOpt.

```
abstract type Policy end

# π is function whose arguments are t and x_t
# π returns u_t
struct StateFeedback <: Policy
    π::Function
end

# π is a function whose arguments are t, x_t and w_{t+1}
# π returns u_t
struct AnticipativeStateFeedback <: Policy
    π::Function
end

# π is a function whose arguments are t, x_t and w_{t-past_size},..., w_{t}
# π returns u_t
struct HistoryDependentPolicy <: Policy
    π::Function
    past_size::Int
end

# π is a function whose arguments are t, x_t and w_{t-past_size},..., w_{t}
# π solves a determinstic optimization problem up to time t+Δh
# π returns u_t, ..., u_{t+step}
struct RollingHorizonPolicy <: Policy
    π::Function
    past_size::Int
    Δh::Int
    step::Int
end

# π is a function whose arguments are t, x_t
# π solves a deterministic optimization problem up to time T
# π computes a forecast of uncertainties between t and T
# π returns u_t
struct RollingForecastPolicy <: Policy
    π::Function
    forecast::Function
end
```

Figure 6.9: Policies types

We observe that the two first policies contains exactly the same Julia type, a Function. However we make two different definitions as their simulation is not the same, then we leverage Julia type dispatch design to efficiently apply the dedicated simulation function.

6.4 Energy management applications

DynOpt has already been applied to energy storage management issues [162] and chapter 4 and 5 of this thesis.

6.4.1 An energy storage management toy example

We explain now how to formulate and solve a simple energy storage management problem over a one day horizon with DynOpt. We consider a microgrid as displayed Figure 6.10. We control a 13.5 kWh battery in a house, connected to the national grid, with 6 kW solar

panels and uncertain electrical demand of around 20 kWh a day. We want to minimize the electricity bill over a day with a 15 min time step.

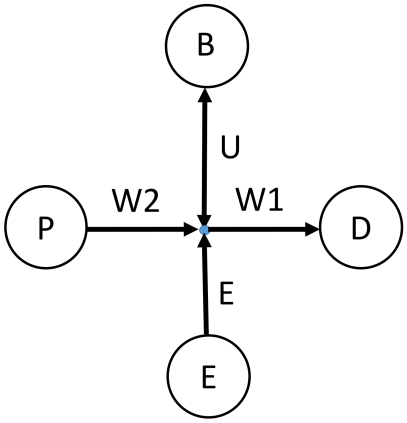

Figure 6.10: A simple microgrid

First, we formulate the mathematical problem. At each time step t, we buy a quantity $\boldsymbol{E}_t = (\boldsymbol{W}_t^1 - \boldsymbol{W}_t^2 + \boldsymbol{U}_t)^+$ of energy in kWh to the national grid at a price c_t in $\$/kWh$. We take the positive part as we assume that it is not possible to sell electricity to the grid. The cost of electricity at each hour is assumed to be deterministic and the same everyday. It is a peak, off-peak hours tariff as displayed Figure 6.14. We present in Figures 6.12 and 6.13 multiple scenarios of electrical demand and solar production as well as histograms of total energy consumed and produced every day. With a large time step of 15 minutes we assume that the controls are in Hazard-Decision (or WaitAndSee) setting. It means that an admissible policy at time t is assumed to be a function of (x_t, w_t).

$$\min_{\boldsymbol{X},\boldsymbol{U}} \mathbb{E}\left[\sum_{t=0}^{T-1} c_t \times (\boldsymbol{W}_t^1 - \boldsymbol{W}_t^2 + \boldsymbol{U}_t)^+\right], \tag{6.26a}$$

$$\text{s.t } \boldsymbol{B}_{t+1} = \boldsymbol{B}_t + \rho_c \boldsymbol{U}_t^+ - \rho_d^{-1} \boldsymbol{U}_t^- \ , \tag{6.26b}$$

$$\underline{B} \leqslant \boldsymbol{B}_t \leqslant \overline{B} \ , \tag{6.26c}$$

$$\sigma(\boldsymbol{U}_t) \subset \sigma(\boldsymbol{W}_0, \ldots, \boldsymbol{W}_t) \ . \tag{6.26d}$$

To solve this problem with DynOpt using stochastic dynamic programming, 60 lines of Julia code (with comments) are used as presented in Figure 6.11. We comment further the code by lines ranges:

1–8 These lines are used to load data, namely 365 past electrical demand scenarios, 365 solar panels past production scenarios and a single cost of electricity. These have a 15 min time step and 24 hours horizon hence every scenario has 96 stages.

9–13 define some constants of the problem. The time step in hours useful to calculate energies, the number of stages and battery efficiency parameters.

14–24 format the demand and solar scenarios in a $T \times N_s \times N_w = 96 \times 365 \times 2$ array. Then we split it in two stacks: the 182 scenarios that will be used to tune optimization algorithm and the 183 scenarios to assess the policy. We choose odd and even days to ensure that both stacks contains scenarios for each months of the year.

25–41 define all the features of a stochastic optimal control problem and build a DynamicOptimizationModel.

42–46 build discrete state and control spaces for exhaustive search algorithm.

47–50 build a white noise for the exhaustive search algorithm. It takes as argument the 182 optimization scenarios with $T = 96$ stages and a integer N_{bins}. At each stages in $\{1, \dots, 96\}$ the 182 equiprobable realizations (with probability 1/182) model the support of a discrete random variable. This variable is quantized using k-means algorithm. The resulting discrete random variable has a support of size lower than N_{bins} and corresponding probabilities.

51–56 build an exhaustive search solver out of the discrete spaces and white noise, then compute costs-to-go using this solver and finally we build a policy. This policiy is anticipative and does not require a model of the noises.

57–60 simulate the policy along the assessment scenarios beginning with a state of charge x_0.

```julia
using DynOpt # Importing DynOpt

# Loading data
using JLD # Loading a package to manipulate hdf5 data in Julia
const demand_scenarios = load("demands.jld")["D"] # 365 demand scenarios
const solar_scenarios = load("solar.jld")["P"] # 365 solar scenarios
const c = load("cost.jld")["c"] # Cost of electricity scenario

const Δt = 15/60 # Time step of the problem in hours
const T = floor(Int, 24/Δt) # Number of time stages
const ρd = 1/0.97 # Battery discharge efficiency
const ρc = 0.98 # Battery charge efficiency

# We format the scenarios for DynOpt: stages x Nscenarios x dimension of w
noise_scenarios = reshape([demand_scenarios solar_scenarios], T, 365, 2);

# We split the scenarios in two stacks

# Odd days: scenarios to tune optimization algorithm
optim_scenarios = noise_scenarios[:,1:2:365,:];

# Even days: scenarios to assess the policy
assess_scenarios = noise_scenarios[:,2:2:365,:];

const xmin = [0.] # Lower bounds for battery soc
const xmax = [13.5] # Upper bound for battery soc

const umin = [-7.].*Δt # Lower bound for battery charge
const umax = [5.].*Δt # Upper bounds for battery charge
const uanticipativity = [WaitAndSee] # Anticipativity of controls

# Problem functions, dynamics, cost, constraints, final cost
f(new_x, t, x, u, w) = new_x[1] = x[1] + ρc*max(u[1], 0.) + ρd*min(u[1],0.)
L(t, x, u, w) =  c[t] * max(0, w[1] - w[2] + u[1] )
B(t, x, u, w) = true # No constraints
K(x) = 0 # No final cost

# Building a DynamicOptimizationModel
sp = DynOpt.DynamicOptimizationModel(T, L, f, B, K, xmin, xmax,
                                     umin, umax, uanticipativity)

const Δx = [0.5] # Step of the states grid
const Δu = [0.1].*Δt # Step of the controls grid
const X = DiscreteSpace(xmin, xmax, Δx); # Building states grid
const U = DiscreteSpace(umin, umax, Δu); # Building controls grid

# We build a white noise by quantizing the 182 optim_scenarios realizations
# at each hour of the day with a maximum of 5 bins
const W = WhiteNoise(optim_scenarios, 5);

solver = GridDPSolver(X, U, W); # Setting an exhaustive search solver

@time V = solve(sp, solver, risk = Expectation()); # Computing costs-to-go

Π = Policy(sp, X, U, V) # Computing a cos-to-go policy

x0 = [0.5] # Choosing a initial battery state of charge

# Simulating the policy along the assessment scenarios
@time L, xres, ures = simulation(sp, Π, assess_scenarios, x0)
```

Figure 6.11: Stochastic Dynamic Programming resolution of a battery problem

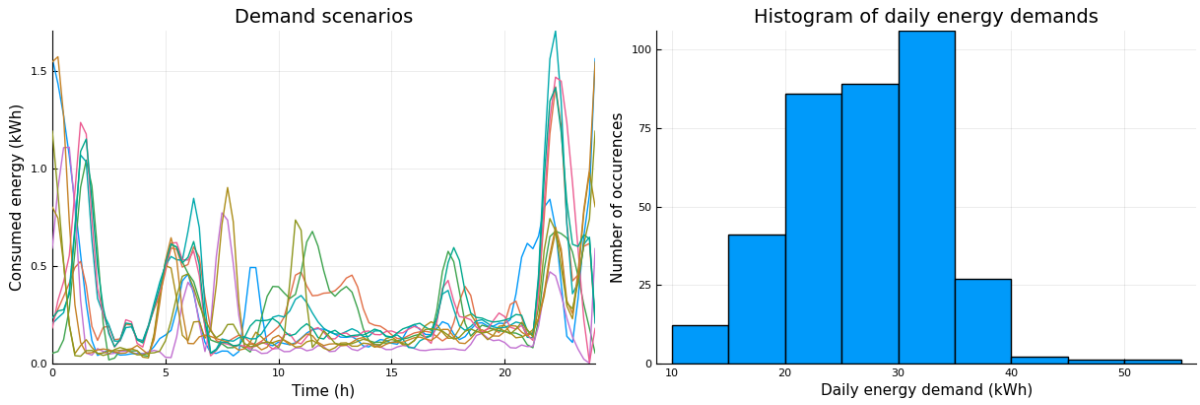

Figure 6.12: Demand scenarios

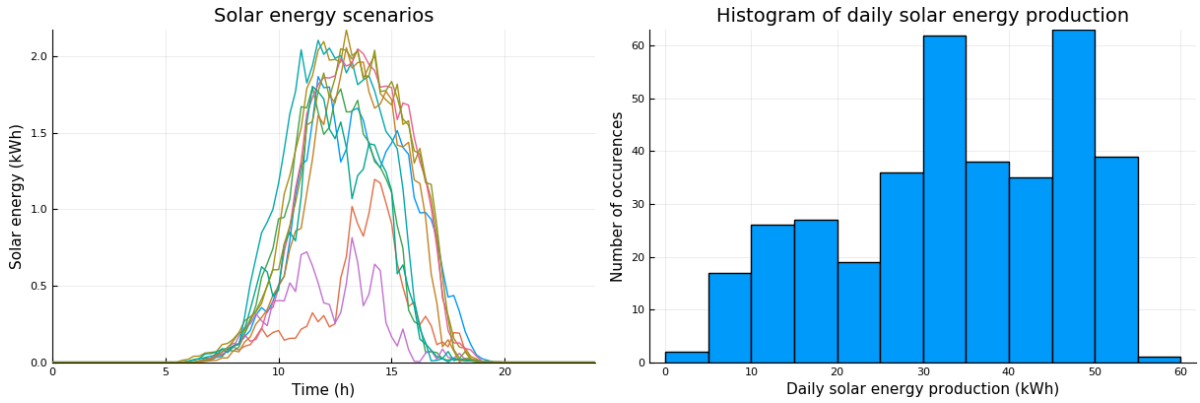

Figure 6.13: Solar scenarios

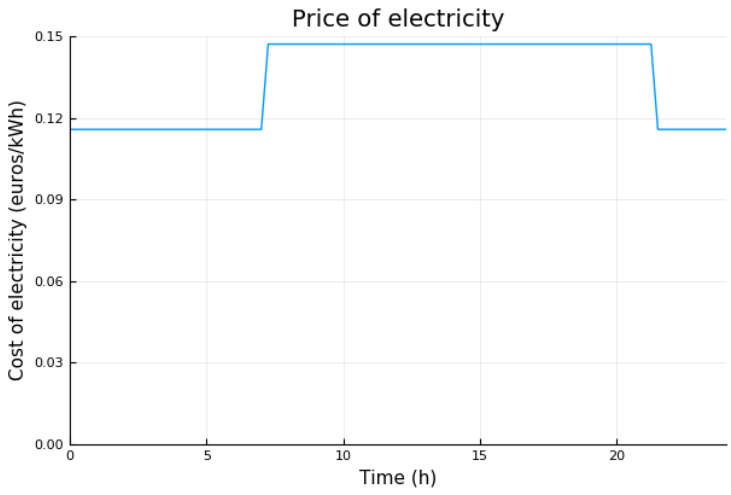

Figure 6.14: Cost of electricity scenario

We can compare the policy obtained by dynamic programming to a heuristic policy. We choose the following intuitive policy

- If solar exceeds demand, store solar up to battery state of charge and battery maximum charge.
- Otherwise, during peak hours, discharge the battery to fulfill the demand up to maximum discharge rate up to minimum stage of charge.
- Otherwise, during off-peak hours, charge the battery at maximum rate up to maximum state of charge.

It gives the following anticipative policy:

$$\begin{aligned}(t, b_t, w_t) \mapsto &\text{if } w_t^2 > w_t^1 \text{ then } \min(w_t^2 - w_t^1, \rho_c^{-1}(\overline{B} - b_t), \overline{U}) \ , \\ &\text{elseif } t \in \text{ peakhours then } \max(w_t^2 - w_t^1, \underline{U}, \rho_d(\underline{B} - b_t)) \ , \\ &\text{else } \min(\overline{U}, \rho_c^{-1}(\overline{B} - b_t)) \ .\end{aligned}$$

It is implemented in Julia code as follows:

```
# Stages of the peak hours, we assume that all evening hours are peak hours
# so as not to charge the battery at the end of the day
const peak_hours = 30:96

# We define a function for the strategy based on hours of the day
π_hours(t,x,w) = (t in peak_hours? max(umin[1], (xmin[1]-x[1])/ρd, w[2]-w[1]) :
 min(umax[1], (xmax[1]-x[1])/ρc)

# We define the heuristic strategy that do not wastes solar
# then apply the hours based strategy
Πheuristic = DynOpt.AnticipativeStateFeedback((t,x,w) -> (w[2]>w[1])?
min(w[2]-w[1], (xmax[1]-x[1])/ρc, umax[1]) : π_hours(t,x,w))

# We simulate the policy along the assessment scenarios
Lheuristic, xheuristic, uheuristic = simulation(sp, Πheuristic,
                                                assess_scenarios,
                                                x0);
```

Figure 6.15: Heuristic policy definition and simulation

We compare the simulation of both policies. It appears on Figure 6.16 and 6.17 that the policies give pretty different state of charge and charge/discharge trajectories for every scenario. In particular the SDP strategy does not charge the battery during off-peak hours which is counter intuitive. The main benefit of the SDP strategy is to charge the battery when there is solar power at a more appropriate rate than the heuristic strategy. Finally, Figure 6.18 displays the distribution of relative gaps that is $2 \times \frac{L - L_{heuristic}}{L + L_{heuristic}}$. As we are minimizing, when the gap is below zero SDP outperforms the heuristic. We observe the SDP outperforms the heuristic over almost all scenarios by around 1%, only 2 scenarios out of 182 are in favor of the heuristic. This toy example shows the interest of DynOpt: it is as simple to build a SDP policy as to build a heuristic policy. SDP policy outperforms heuristic policy even with a pretty poor model of uncertainties and coarse discretizations

of state, control spaces and quantification of random variables. Finally SDP requires here 0.27 seconds to compute value functions offline and then computes a decision in around 2×10^{-5} seconds. The heuristic policy does not require offline computation time and computes a decision online in approximately 10^{-6} seconds which is faster but with a time step of 15 minutes both policies are fast enough.

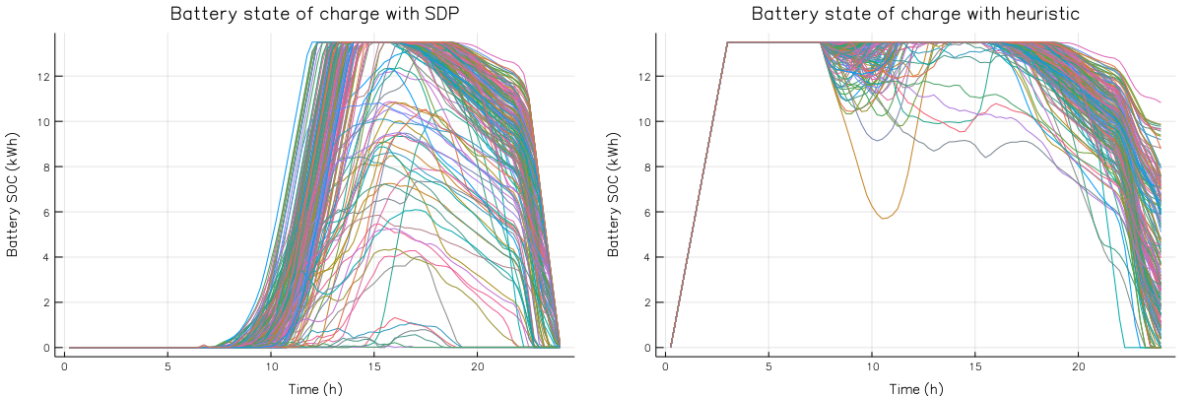

Figure 6.16: Battery state of charge trajectories

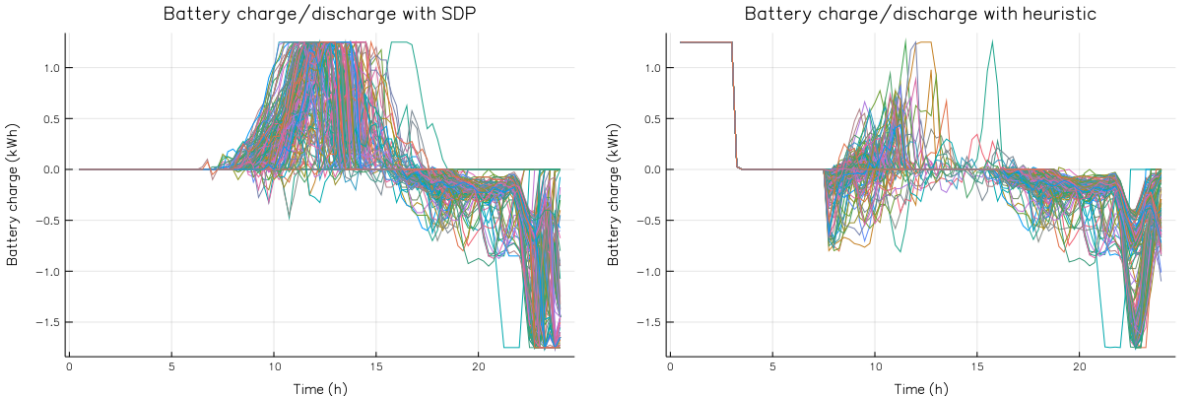

Figure 6.17: Battery charge/discharge trajectories

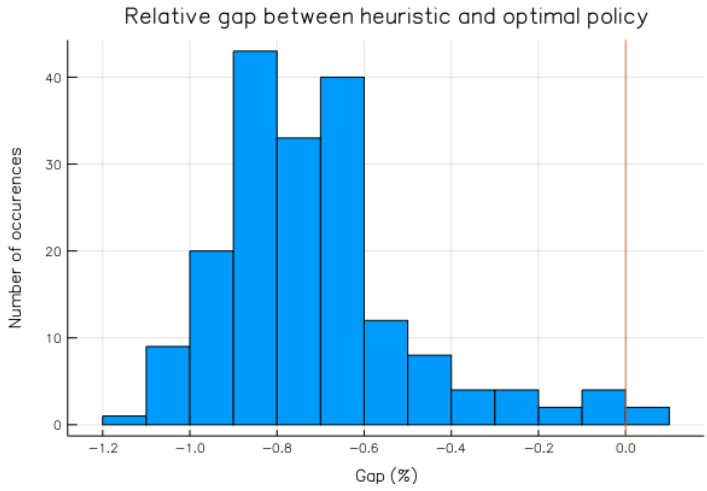

Figure 6.18: Histogram of relative gap between policies simulation costs

6.4.2 Two-Time Scales problems

One drawback of this toy example for a real energy management problem is the lack of a final cost to go K leading to an empty battery at the end of each day, which might be inefficient for the next day. In general, we need to compute a *final value of energy*, that is the economic value of a state of charge at the end of each day. There are different methods to compute a relevant final cost for such problems. We distinguish methods based on stationary assumptions, implying that the final cost function is the same for everyday of an infinite horizon, and non stationary assumptions, implying that the final cost changing everyday of a finite horizon. In the subsection every variables are indexed by a day index $d \in \{0, \dots D\}$, where D in a number of days and the previous intraday index $t \in \{0, \dots T\}$. D an be equal to $+\infty$.

Stationary case: infinite horizon value iteration

In this part we briefly present infinite horizon two time scales problems and resolution algorithms. We make the following stationary assumptions:

- the state spaces $\{\mathbb{X}_t\}_{t\in\{0,\dots,T\}}$ are the same everyday and $\mathbb{X}_0 = \mathbb{X}_T$
- the control spaces $\{\mathbb{U}_t\}_{t\in\{0,\dots,T-1\}}$ are the same everyday
- the uncertainty spaces $\{\mathbb{W}_t\}_{t\in\{0,\dots,T\}}$ are the same everyday
- the costs $\{L_t\}_{t\in\{0,\dots,T\}}$ are the same everyday
- the dynamics $\{f_t\}_{t\in\{0,\dots,T\}}$ are the same everyday
- the constaints $\{B_t\}_{t\in\{0,\dots,T\}}$ are the same everyday
- the random variables $\{\boldsymbol{W}_{d,0,\dots,T}\}_{d\in\{0,\dots D\}}$ are independent and indentically distributed, that is the law of $\boldsymbol{W}_{d,0,\dots,T} = \{\boldsymbol{W}_{d,0}, \dots, \boldsymbol{W}_{d,T}\}$ does not depend on d.

The whole idea is to view the daily final cost as the value of an infinite horizon discounted problem. We call $\gamma \in [0, 1[$ the daily discount factor.

$$\forall x \in \mathbb{X}_0 = \mathbb{X}_T,\ K(x) = \min_{\boldsymbol{X},\boldsymbol{U}} \mathbb{E}\left[\sum_{d=0}^{+\infty} \gamma \sum_{t=0}^{T-1} L_t(\boldsymbol{X}_{d,t}, \boldsymbol{U}_{d,t}, \boldsymbol{W}_{d,t+1}) \ , \right] \tag{6.27a}$$

$$\text{s.t } \boldsymbol{X}_{d,t+1} = f_t(\boldsymbol{X}_{d,t}, \boldsymbol{U}_{d,t}, \boldsymbol{W}_{d,t+1}) \ , \tag{6.27b}$$

$$B_t(\boldsymbol{X}_{d,t}, \boldsymbol{U}_{d,t}, \boldsymbol{W}_{d,t+1}) \leqslant 0 \ , \tag{6.27c}$$

$$\boldsymbol{X}_{d,0} = \boldsymbol{X}_{d-1,T} \ , \tag{6.27d}$$

$$\sigma(\boldsymbol{U}_{d,t}) \subset \sigma(\boldsymbol{X}_{d,0}, \boldsymbol{W}_{d,0}, \dots, \boldsymbol{W}_{d,t}) \ . \tag{6.27e}$$

With the stationary assumptions K verifies the following fixed point equation:

$$\forall x \in \mathbb{X}_0 = \mathbb{X}_T,\ K(x) = \min_{\boldsymbol{X},\boldsymbol{U}} \mathbb{E}\left[\sum_{t=0}^{T-1} L_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) + \gamma K(\boldsymbol{X}_T) \ , \right] \tag{6.28a}$$

$$\text{s.t } \boldsymbol{X}_{t+1} = f_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) \ , \tag{6.28b}$$

$$B_t(\boldsymbol{X}_{d,t}, \boldsymbol{U}_{d,t}, \boldsymbol{W}_{t+1}) \leqslant 0 \ , \tag{6.28c}$$

$$\boldsymbol{X}_{d,0} = \boldsymbol{X}_{d-1,T} \ , \tag{6.28d}$$

$$\sigma(\boldsymbol{U}_{d,t}) \subset \sigma(\boldsymbol{W}_0, \dots, \boldsymbol{W}_t) \ , \tag{6.28e}$$

and we can compute K using a fixed point algorithm, this is a discounted value iteration. When the discount factor *gamma* is equal to 1 we can apply the same algorithm but without a guarantee of convergence. Convergence fully depends on the intraday problem. This is a total cost value iteration.

It is also possible to view K as an average cost that is:

$$\forall x \in \mathbb{X}_0 = \mathbb{X}_T,\ K(x) = \min_{\boldsymbol{X},\boldsymbol{U}} \mathbb{E}\left[\lim_{D\to\infty} \frac{1}{D} \sum_{d=0}^{D} \sum_{t=0}^{T-1} L_t(\boldsymbol{X}_{d,t}, \boldsymbol{U}_{d,t}, \boldsymbol{W}_{d,t+1}) \ , \right] \tag{6.29a}$$

$$\text{s.t } \boldsymbol{X}_{d,t+1} = f_t(\boldsymbol{X}_{d,t}, \boldsymbol{U}_{d,t}, \boldsymbol{W}_{d,t+1}) \ , \tag{6.29b}$$

$$B_t(\boldsymbol{X}_{d,t}, \boldsymbol{U}_{d,t}, \boldsymbol{W}_{d,t+1}) \leqslant 0 \ , \tag{6.29c}$$

$$\boldsymbol{X}_{d,0} = \boldsymbol{X}_{d-1,T} \ , \tag{6.29d}$$

$$\sigma(\boldsymbol{U}_{d,t}) \subset \sigma(\boldsymbol{X}_{d,0}, \boldsymbol{W}_{d,0}, \dots, \boldsymbol{W}_{d,t}) \ . \tag{6.29e}$$

In this case K does not verify a fixed point equation. However the so-called *relative value* $K - K(\hat{x})$ with $\hat{x} \in \mathbb{X}_0$ verifies one:

$$\forall x \in \mathbb{X}_0 = \mathbb{X}_T,\ K(x) - K(\hat{x}) = \min_{\boldsymbol{X},\boldsymbol{U}} \mathbb{E}\left[\sum_{t=0}^{T-1} L_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) + K(\boldsymbol{X}_T) - K(\hat{x}) \ , \right] \tag{6.30a}$$

$$\text{s.t } \boldsymbol{X}_{t+1} = f_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) \ , \tag{6.30b}$$

$$B_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) \leqslant 0 \ , \tag{6.30c}$$

$$\sigma(\boldsymbol{U}_t) \subset \sigma(\boldsymbol{W}_0, \dots, \boldsymbol{W}_t) \ . \tag{6.30d}$$

We implement three algorithms to compute a final cost to go based on value iteration as presented in [145] and [155] and applied in [154] for such a kind of problems. DynOpt provides functions to compute a final cost K iteratively.

```
#  A abstract final cost solver
abstract type FinalCostSolver end

struct InfiniteHorizonDiscountedCost{T <: AbstractSolver} <: FinalCostSolver

    subproblem_solver::T # Solver of the intraday problem

    γ::Float64 # Discount factor

    max_iter::Int # Maximum number of iterations

    ϵ::Float64 # Convergence tolerance
end

struct InfiniteHorizonAverageCost{T <: AbstractSolver} <: FinalCostSolver

    subproblem_solver::T # Solver of the intraday problem

    max_iter::Int # Maximum number of iterations

    ϵ::Float64 # Convergence tolerance
end

struct InfiniteHorizonTotalCost{T <: AbstractSolver} <: FinalCostSolver

    subproblem_solver::T # Solver of the intraday problem

    max_iter::Int # Maximum number of iterations

    ϵ::Float64 # Convergence tolerance
end
```

Figure 6.19: Final cost solvers

The idea of these solvers is to solve the intraday problem multiple times with an updated final cost each time. The algorithms stop when a maximum number of iteration is reached or when the final cost has converged with a given tolerance. The three solvers are used for type dispatch of a fix point algorithm function. At iteration k with a given final cost $K^{(k)}$ the value of the intraday problem V_0^{k+1} is computed using a DynOpt algorithm (SDP, SDDP, MIDAS,...). Depending on the type of solver the update rule of the final cost changes:

- Discounted problems: $K^{(k+1)}(x) = \gamma V_0^{(k+1)}(x)$,
- Total cost problems: $K^{(k+1)}(x) = V_0^{(k+1)}(x)$,
- Average cost problems: $K^{(k+1)}(x) = V_0^{(k+1)}(x) - V_0^{(k+1)}(x_0)$.

Non stationary case: two time scales stochastic dynamic optimization

Without the stationary assumptions the final cost K should depend on the day. We can view it as a daily value function of a two time scales problem:

$$\forall x \in \mathbb{X}_0 = \mathbb{X}_T,\ \min_{\boldsymbol{X},\boldsymbol{U}} \mathbb{E}\left[\sum_{d=0}^{D}\sum_{t=0}^{T-1} L_{d,t}(\boldsymbol{X}_{d,t}, \boldsymbol{U}_{d,t}, \boldsymbol{W}_{d,t+1})\ ,\right] \tag{6.31a}$$

$$\text{s.t } \boldsymbol{X}_{d,t+1} = f_{d,t}(\boldsymbol{X}_{d,t}, \boldsymbol{U}_{d,t}, \boldsymbol{W}_{d,t+1})\ , \tag{6.31b}$$

$$B_{d,t}(\boldsymbol{X}_{d,t}, \boldsymbol{U}_{d,t}, \boldsymbol{W}_{d,t+1}) \leqslant 0\ , \tag{6.31c}$$

$$\boldsymbol{X}_{d,0} = \boldsymbol{X}_{d-1,T}\ , \tag{6.31d}$$

$$\sigma(\boldsymbol{U}_{d,t}) \subset \sigma(\boldsymbol{W}_{0,0}, \ldots, \boldsymbol{W}_{d,t})\ . \tag{6.31e}$$

when the random variables $\{\boldsymbol{W}_{d,0,\ldots,T}\}_{d\in\{0,\ldots D\}}$ are independent we can define a daily value functions K_d verifying the following backward recursion:

$$\forall x \in \mathbb{X}_{d,0},\ K_d(x) = \min_{\boldsymbol{X},\boldsymbol{U}} \mathbb{E}\left[\sum_{t=0}^{T-1} L_{d,t}(\boldsymbol{X}_{d,t}, \boldsymbol{U}_{d,t}, \boldsymbol{W}_{d,t+1}) + K_{d+1}(\boldsymbol{X}_{d,T})\ ,\right] \tag{6.32a}$$

$$\text{s.t } \boldsymbol{X}_{d,t+1} = f_{d,t}(\boldsymbol{X}_{d,t}, \boldsymbol{U}_{d,t}, \boldsymbol{W}_{d,t+1})\ , \tag{6.32b}$$

$$B_{d,t}(\boldsymbol{X}_{d,t}, \boldsymbol{U}_{d,t}, \boldsymbol{W}_{d,t+1}) \leqslant 0\ , \tag{6.32c}$$

$$\sigma(\boldsymbol{U}_{d,t}) \subset \sigma(\boldsymbol{W}_{d,0}, \ldots, \boldsymbol{W}_{d,t})\ . \tag{6.32d}$$

We present in chapter 5 of this thesis different ways to compute these daily value functions. They consist in decomposition the two time scales problem in two mono-scale problem, hence we can apply DynOpt core functionality to solve them. We do not provide currently generic DynOpt objects to define this kind of problems and automatically decompose and solve them using DynOpt. This is one of our perspectives.

6.4.3 $M\mu GO$: Modular Microgrids Optimization

DynOpt can be used as a backend for energy management applications. Using Julia packages Mux.jl [14] and HTTP.jl [15] we are able to build REST APIs to call policies produced with DynOpt or directly resolution algorithms for a wide variety of energy management problems under uncertainty. The REST API's can be deployed in any Linux machine using Docker[16] images. We used such a scheme in the following applications:

- Model Predictive Control of a DC micro-grid simulation: an isolated DC micro-grid voltage stability problem has been modeled in DynOpt to produce a MPC policy. A REST API has been built to deploy the strategy in a Raspberry Pi able to control a Simulink simulation in another computer. This is presented in chapter 4 of this thesis.

[14] https://github.com/JuliaWeb/Mux.jl
[15] https://github.com/JuliaWeb/HTTP.jl
[16] https://www.docker.com/

- Sizing of a battery: we modeled a battery sizing problem for a stationary storage in a micro-grid using DynOpt. Using two time scales algorithms we are able to compute an optimal sizing of the battery as well as a maintenance strategy. We built a REST API and deployed it in the cloud to serve as a back-end for a battery sizing web-page. This is presented in chapter 5 of this thesis.

- Thermal control of a real building : we modeled a building thermal control problem using DynOpt and built an API so that a smart plug can receive On-Off instructions in real time from the produced policy. This is presented in chapter 7 of this thesis.

6.5 Conclusion and perspectives

This paper introduced DynOpt, a Julia package, to model and solve stochastic optimization problems and produce online control policies used for simulations, DynOpt can be implemented for real time control of systems. We presented the interface as well as different underlying structure and algorithms.

Thanks to its modular architecture, DynOpt can be easily extended. In particular we believe it is developer friendly to add new kind of policies or resolution algorithms. Stochastic Programming methods can therefore easily extend DynOpt. The forward-backward algorithms abstraction layer allows to easily experiment new ways to approximate value functions, for example it might be interesting to model value functions as MIPs using MIP representation of multivariate functions introduced in [165]. Adding new classes of value functions should also easily allow to implement Reinforcement Learning methods. We think the user interface of DynOpt already makes possible concise modeling of stochastic optimization problems. We began to implement a higher level user interface à la JuMP using Julia metaprogramming capabilities. A prototype version is already operational but should be tested for many kind of problems and improved. Risk modeling contains currently only the expectation. An implementation of the Nested Conditional Value at Risk has been added but only integrated in the exhaustive search solver. The integration of risk models in the mathematical programming based policies of DynOpt (SDDP, MIDAS, Inner Approximation, MPC...) is challenging but should be addressed shortly.

Finally, DynOpt, is currently being used as a back-end for energy management tools at Efficacity. It aims at providing a versatile stochastic optimization solver for different energy management software, even existing commercial ones that could improve their optimization features. It remains to rigorously test it on different test-beds and demonstrators so as to improve it and envisage commercial applications.

Chapter 6. Bibliography

[144] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, third edition, 2005.

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

[146] D. P. Bertsekas and H. Yu. A unifying polyhedral approximation framework for convex optimization. *SIAM Journal on Optimization*, 21(1):333–360, 2011.

[147] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *CoRR*, abs/1411.1607, 2014.

[148] J. Bonnans, Frederic, D. Giorgi, V. Grelard, B. Heymann, S. Maindrault, P. Martinon, O. Tissot, and J. Liu. Bocop – A collection of examples. Technical report, INRIA, 2017.

[149] P. Carpentier, J.-P. Chancelier, G. Cohen, and M. De Lara. *Stochastic Multi-Stage Optimization. At the Crossroads between Discrete Time Stochastic Control and Stochastic Programming*. Springer-Verlag, Berlin, 2015.

[150] O. Dowson. The policy graph decomposition of multistage stochastic optimization problems. *Optimization Online*, 2018.

[151] O. Dowson and L. Kapelevich. Sddp. jl: a julia package for stochastic dual dynamic programming. *Optimization Online. URL http://www. optimization-online. org/DB_HTML/2017/12/6388. html*, 2017.

[152] I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[153] H. Gevret, J. Lelong, and X. Warin. STochastic OPTimization library in C++. Research report, EDF Lab, Sept. 2016.

[154] P. Haessig, H. B. Ahmed, and B. Multon. Energy storage control with aging limitation. In *PowerTech, 2015 IEEE Eindhoven*, pages 1–6. IEEE, 2015.

[155] O. Hernández-Lerma and J. B. Lasserre. *Discrete-time Markov control processes: basic optimality criteria*, volume 30. Springer Science & Business Media, 2012.

[156] V. Leclère, P. Carpentier, J.-P. Chancelier, A. Lenoir, and F. Pacaud. Exact converging bounds for stochastic dual dynamic programming via fenchel duality. *Optimization Online*, 2018.

[157] V. Leclère, H. Gérard, F. Pacaud, and T. Rigaut. Stochdynamicprogramming. jl a julia library for multistage stochastic optimization. `https://github.com/JuliaStochOpt/StochDynamicProgramming.jl`, 2017.

[158] M. V. Pereira and L. M. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1-3):359–375, 1991.

[159] A. Philpott, F. Wahid, and F. Bonnans. MIDAS: A Mixed Integer Dynamic Approximation Scheme. Research report, Inria Saclay Ile de France, June 2016.

[160] W. B. Powell. Clearing the jungle of stochastic optimization. *Informs*, pages 109–137, 2014.

[161] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

[162] T. Rigaut, P. Carpentier, J. Chancelier, M. D. Lara, and J. Waeytens. Stochastic optimization of braking energy storage and ventilation in a subway station. *IEEE Transactions on Power Systems*, pages 1–1, 2018.

[163] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[164] A. Shapiro, W. Tekaya, J. P. da Costa, and M. P. Soares. Final report for technical cooperation between georgia institute of technology and ons–operador nacional do sistema elétrico. *Georgia Tech ISyE Report*, 2012.

[165] J. P. Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, 57:3–57, 2015.

Chapter 7

Energy aware temperature control of a house using Stochastic Dual Dynamic Programming: a first testbed implementation

This is a joint work with Frédéric Bourquin and Julien Waeytens.

Chapter Abstract

We present the implementation of a control strategy for the temperature of a real house, minimizing the energy energy consumption with stochastic thermal gains. We present a calibration method of an RC model, which models a building thermal behavior. This calibration methods allows to determine some house thermal parameters but also to build a stochastic model of the thermal gains that are not generated by electrical heaters. These calibrated RC model and stochastic model of the thermal gains are used to build a control policy to minimize the energy consumption of the house while maintaining a comfortable temperature. This policy is computed using the SDDP algorithm using DynOpt, the toolbox introduced Chapter 6. We build a software architecture based on containerized APIs to calibrate the model, compute the control policy and apply the control policy online from a server or in the cloud without requiring a human input. We implement this architecture to a real house, controlling two electrical heaters in two rooms using smart plugs.

Contents

7.1 Introduction

This chapter describes a test-bed implementation of an online control using stochastic optimization and in particular the Stochastic Dual Dynamic Programming (SDDP) algorithm implemented in DynOpt, presented in Chapter 6 of this manuscript. The test bed is a chalet in the mini city demonstrator Sense City [172]. The thermal dynamic of the chalet is modeled with an RC electrical equivalent model. We introduce a calibration method that does not require a human input of the model. This calibrated model is used to apply the SDDP algorithm to minimize the energy consumption of the building while satisfying thermal comfort constraints with uncertain indoor thermal gains. Finally we present how to build REST APIs based on DynOpt that can be embedded in any Linux machine and how we build data pipelines for model calibration and control of a real stochastic dynamical system. We apply our implementation strategy to control the temperature of the test bed building and present results.

7.1.1 Literature on temperature control in houses and experimentation

There exists different modeling strategies to simulate the thermal behavior of buildings. We refer to [168] for an exhaustive comparison and assessment of existing simulation methods. In this paper we use a RC model relying on an electrical interpretation of a building thermal dynamic. A comparison of calibration methods for such models are also presented in [168], the comparison shows that Interior Point method outperforms meta heuristics. In [166, 171] they present a more detailed physical model of a building and an associated calibration strategy based on the Levenberg-Marquardt algorithm where the gradient is computed efficiently using the adjoint of the model. A similar method is

applied in [166]. In [173] they present a calibration strategy in two phases combining a meta heuristic and a physical model based calibration.

In the sequel we combine a meta heuristic to find a first guess of the RC model unknown parameters and a gradient based algorithm to improve the parameters estimation.

Multiple uncertain phenomena impact buildings thermal behavior namely, outdoor temperature, door and window openings, indoor thermal gains, solar gains... Control algorithms such as Model Predictive Control (MPC) and stochastic MPC have been widely applied to handle uncertainty in thermal control of buildings [177, 182, 175]. In [178] they develop a MPC strategy for a four-storey building and the control strategy is applied to detailed simulation but not a real test bed. Some MPC applications exist on real buildings such as [166, 179]. Stochastic Dynamic Programming (SDP) [167, 169] and Approximate Dynamic Programming [170, 180] have been applied on building simulations, we refer to [174] for an overview of the literature, and sometimes on real test beds as in [181]. SDP computes value functions offline and then at time t only needs to solve a small optimization problem using a value function while MPC requires to solve a possibly large mathematical program with a dedicated solver. SDP suffers the well known curse of dimensionality. There are two state variables in a monozone RC model, SDP is limited to at most four state variables. Hence applying SDP to large buildings or multiple houses is out of reach. When the dynamic is linear in state and control variables the Stochastic Dual Dynamic Programming algorithm makes it possible to compute value functions approximations for problems with up to 40 state variables. A thermal control of a house using SDDP is presented in [176] and compared to a MPC approach. Their results show that SDDP outperforms MPC on several simulations but no application on a real test bed is performed.

We present in the sequel a thermal control strategy of a real house with uncertain thermal gains using SDDP.

7.1.2 Structure of the chapter

In this chapter we first present the building thermal model that we use to optimize its heating. We emphasize on a model calibration techniques that use only measurements and does not require a human input. Then we formulate an optimization problem that aims at minimizing the energy consumption cost of the house while ensuring a decent thermal comfort. This problem displays stochasticity as some indoor and outdoor thermal gains are not measured and could not be modeled with physical equations. We also present how this optimization problem is solved to produce an online control policy. Finally we present how these calibration and optimization techniques are implemented to control the electrical heaters of a house, using stochastic optimization, without a human input.

7.2 Building energy model and parameters calibration

We depict in this section the physical equations we use to model the temperature dynamics in the house. We present as well the calibration strategy we implement to infer a generic building thermal characteristics without prior physical knowledge. Then we model some uncontrollable and unforeseeable thermal phenomena occurring inside a generic building using optimization and a simple statistical model.

7.2.1 Building thermal model

We model the thermal behavior of the house so as to control its temperature. We use a simple model in order to preserve tractability of the eventual optimization algorithms we will use.

The thermal behavior of the house is modeled using an electrical analogy, a R6C2 model. as introduced in [176, 168]. It consists roughly in representing the walls as an equivalent homogeneous material exchanging heat with the air represented as a single zone as well. Let $\theta(t)$ and $\theta^w(t))$ (K) be respectively the temperature of the air and of the walls at time t. We model the dynamics of these temperature with the following parametric differential equations:

$$c_i \frac{d\theta}{dt} = \frac{\theta^w(t) - \theta(t)}{R_i + R_s} + \frac{R_v + R_f}{R_v R_f}\Big(\theta^o(t) - \theta(t)\Big) + \gamma q(t) + \phi^i(t) \tag{7.1a}$$

$$c_w \frac{d\theta^w}{dt} = \frac{\theta(t) - \theta^w(t)}{R_i + R_s} + \frac{\theta^o(t) - \theta^w(t)}{R_m + R_o} + (1-\gamma) q(t) + \phi^w(t) \tag{7.1b}$$

where c_i is the thermal capacity of the zone (air + furniture), c_w the thermal capacity of the walls, γ (%) is the proportion of heat from electrical heater that is transmitted to the air,$(1-\gamma)$ is the proportion transmitted to walls and $\{R_i, R_s, R_v, R_f, R_m, R_o\}$ are resistances modeling the building thermal behavior, we refer the reader to [176, 168] for a more precise definition.

Four time varying variables remain $\theta^o(t)$ (K) the outside air temperature, $q(t)$ the heat generated (W) by the electric heaters, $\phi^i(t)$ (W) and $\phi^w(t)$ respectively the exogenous indoor thermal gain and the walls thermal gain (solar radiation, people heat, computers...).

Measured variables, known and unknown variables

In model (7.1) the time varying variables $\theta, \theta^w, \theta^o, q$ are assumed measured. The first three temperatures are measured using 4-wire PT100 probes while q can be measured using smart plugs for instance. The thermal capacity of the zone can be computed as $c_i = c_p \times \mathcal{V}$ where c_p is the thermal capacity of the air at $300K$, and $\mathcal{V}$ is the volume of the zone. This approximation falls down to consider that the zone is empty. Here we assume that c_i is unknown so that the calibration strategy does not require a human input to define the volume of the building.

All resistances, c_w and γ are unknown and require calibration. The fluxes ϕ and ϕ^w are unknown as well, we present here under how we dealt with them.

7.2.2 Model calibration

As stated previously $\theta, \theta^w, \theta^o, q$ are measured, hence we have historical data to calibrate model (7.1), which implies finding a value for the unknown parameters in order to simulate the house thermal behavior properly. We call Δt the sample rate of the measures, that are assumed synchronous. T_h is the number of past historical data available for calibration. $\{(\overline{\theta}_t, \overline{\theta}^w_t, \overline{\theta}^o_t, \overline{q}_t)\}_{t=0,\dots,T_h}$ is the sequence of historical measures.

As we do not measure ϕ and ϕ^w we perform a calibration in two phases. For every sequence of the building constant parameters, resistances, walls capacity c_w and γ we infer the time varying fluxes ϕ, ϕ^w to produce a stochastic model of their evolution.

House parameters calibration

First we group and replace the unknown constant parameters in (7.1) by optimization variables $(p_1, \ldots, p_7)$:

$$p_1 \frac{d\theta}{dt} = p_2\Big(\theta^w(t) - \theta(t)\Big) + p_3\Big(\theta^o(t) - \theta(t)\Big) + p_4 q(t) + \phi_t \tag{7.2a}$$

$$p_5 \frac{d\theta^w}{dt} = p_6\Big(\theta(t) - \theta^w(t)\Big) + p_7\Big(\theta^o(t) - \theta^w(t)\Big) + (1 - p_4) q(t) + \phi_t^w \tag{7.2b}$$

We discretize these two differential equations using regular forward differentiation with time step Δt producing the following fitting problem:

$$\min_{p_1,\ldots,p_7,\theta_t,\theta_t^w,\phi_t,\phi_t^w} \quad \sum_{t=0}^{T_h} |\theta_t - \overline{\theta}_t| + |\theta_t^w - \overline{\theta}_t^w| \,, \tag{7.3a}$$

$$\text{s.t } p_1\theta_{t+1} = p_1\theta_t + \Delta t\Big(p_2\big(\theta_t^w - \theta_t\big) + p_3\big(\theta_t^o - \theta_t\big) + p_4 q_t + \phi_t\Big) \,, \tag{7.3b}$$

$$p_5\theta_{t+1}^w = p_5\theta_t^w + \Delta t\Big(p_6\big(\theta_t - \theta_t^w\big) + p_7\big(\theta_t^o - \theta_t^w\big) + (1 - p_4) q_t + \phi_t^w\Big) \,, \tag{7.3c}$$

$$0 \leqslant p_i \leqslant p_i^{max},\ \phi_t \geqslant 0\ \phi_t^w \geqslant 0 \,. \tag{7.3d}$$

Problem (7.3) has a convex objective however both constraints (7.3b) and (7.3c) contain bi-linear terms in decision variables. This problem may have multiple local minima and we would need Non-Linear Programming techniques to compute admissible solutions. We notice that the exogenous thermal gains ϕ_t and ϕ_t^w for $t = 0, \ldots, T_h$ are decision variables of problem (7.3). It may adds a significant number of decision variable to the problem is the horizon of calibration T_h is large, which is suitable for a good calibration. We introduce the following optimization problem whose value V_m depends on the model parameters $p_1, \ldots, p_7$.

$$V_m(p_1, \ldots, p_7) = \min_{\theta_t,\theta_t^w,\phi_t,\phi_t^w} \quad \sum_{t=0}^{T_h} |\theta_t - \overline{\theta}_t| + |\theta_t^w - \overline{\theta}_t^w| \,, \tag{7.4a}$$

$$\text{s.t } p_1\theta_{t+1} = p_1\theta_t + \Delta t\Big(p_2\big(\theta_t^w - \theta_t\big) + p_3\big(\theta_t^o - \theta_t\big) + p_4 q_t + \phi_t\Big) \,, \tag{7.4b}$$

$$p_5\theta_{t+1}^w = p_5\theta_t^w + \Delta t\Big(p_6\big(\theta_t - \theta_t^w\big) + p_7\big(\theta_t^o - \theta_t^w\big) + (1 - p_4) q_t + \phi_t^w\Big) \,, \tag{7.4c}$$

$$\phi_t \geqslant 0\ \phi_t^w \geqslant 0 \,. \tag{7.4d}$$

This problem is linear so we can perform linear programming techniques to solve it

efficiently. Moreover problem (7.3) is equivalent to the following problem

$$\min_{p_1,\dots,p_7} V_m(p_1,\dots,p_7)\ , \tag{7.5a}$$

$$\text{s.t } 0 \leqslant p_i \leqslant p_i^{max}\ . \tag{7.5b}$$

on which we can apply non linear programming techniques where the value of the oracle V_m is computed using linear programming. We present here under our optimization strategy of problem (7.5) that produces a set of parameters $p_1^\sharp,\dots,p_7^\sharp$.

Producing an initial guess using Simulated Annealing The classical method to calibrate RC models is to provide an initial guess of the parameters based on physical study of the building. However we prefer a generic method that does not require a human input. From a random initial guess we perform a meta heuristic, the so called simulated annealing algorithm, to produce a decent initial sequence of parameters. By decent we mean that the indoor temperature does not reach unrealistic values (e.g more than 50 or less than -50 celsius degrees).

Improving parameters using a gradient based method Now that we obtained a decent initial guess for the parameters we perform a gradient descent based algorithm, here we chose BFGS, to improve the objective value and obtain the best parameters possible.

Stochastic model of the exogenous thermal gains

We obtain two sequences of historical exogenous thermal gains $\{\overline{\phi}_0,\dots,\overline{\phi}_{T_h}\}$ and $\{\overline{\phi}_0^w,\dots,\overline{\phi}_{T_h}^w\}$ by solving problem (7.4) with the sequence of parameters previously computed $p_1^\sharp,\dots,p_7^\sharp$. These sequences of historical thermal gains are produced to replace data that we are physically unable to measure and that is cumbersome to model. Now we can perform a regular statistical analysis on these two sequences of inferred historical data to produce a stochastic model of the exogenous thermal gains. In the sequel bold uppercase variables $\boldsymbol{X}$ are random variables while lowercase letters x are used for deterministic variables.

In this Chapter we use an order 1 auto-regressive model ($AR(1)$) for the exogenous thermal gains. Φ_t and Φ_t^w follow the $AR(1)$ models:

$$\boldsymbol{\Phi}_{t+1} = a\boldsymbol{\Phi}_t + b + \boldsymbol{R}\ , \tag{7.6a}$$

$$\boldsymbol{\Phi}_{t+1}^w = a^w\boldsymbol{\Phi}_t^w + b^w + \boldsymbol{R}^w\ . \tag{7.6b}$$

Coefficients a, b and a^w, b^w are computed by performing a linear least square regression using the sequences of historical thermal gains $\{\overline{\phi}_0,\dots,\overline{\phi}_{T_h}\}$ and $\{\overline{\phi}_0^w,\dots,\overline{\phi}_{T_h}^w\}$.

The law of the residuals $\boldsymbol{R}$ and $\boldsymbol{R}^w$ is obtained by quantization. For instance, we consider an uniform discrete random variable with support $\{\overline{\phi}_{t+1} - a\overline{\phi}_t - b\}_{t=0\dots T_h}$, that is each realization has probability $\frac{1}{T_h}$. We apply the *k-means* algorithm to quantize this random variable to obtain the law of $\boldsymbol{R}$ with a smaller support.

7.3 Optimization problem statement

We design an Energy Management System for the electrical heaters of the house to minimize the heating electrical consumption while ensuring a good thermal comfort. This EMS implementation assumes that good outside temperature forecasts are available but considers that the exogenous thermal gains are unpredictable.

We adopt a discrete time frame to model the optimization problem as heating decisions will be made at discrete times (e.g every 15 minutes). As previously we call Δt the time step, T is the number of time stages during which we make heating decisions. We formulate a discrete time stochastic optimization problem based on the physical model we presented.

7.3.1 We control heating energy

The electrical heaters can produce heat up to a maximum power Q^{max} in Watts (W). We call $\boldsymbol{U}_t$ (Wh) the heat produced during the time interval $[t, t+\Delta t[$. This energy should be maintained below the threshold $Q^{max} \times \Delta t$ to remain achievable by the electrical heaters. $\boldsymbol{U}_t$ is the decision variable of the problem and $\mathbb{U}_t$ the admissible set.

The sequence of decisions $\{\boldsymbol{U}_0, \dots, \boldsymbol{U}_{T-1}\}$ is a stochastic process. As time goes the random exogenous thermal gains $\boldsymbol{\Phi}_t$ and $\boldsymbol{\Phi}_t^w$ materializes. We call $\boldsymbol{I}_t = (\boldsymbol{\Phi}_t, \boldsymbol{\Phi}_t^w)^T$. $\boldsymbol{\Phi}_{t+1}$ the thermal gain vector. $\boldsymbol{I}_{t+1}$ is the exogenous thermal gain during the interval $[t, t+\Delta[$. As a decision $\boldsymbol{U}_t$ is taken knowing all the previous thermal gains realizations $(w_0, \dots, w_t)$, it is a random variable.

7.3.2 State and dynamics

The system is described by two physical state variables, the indoor air temperature $\boldsymbol{\Theta}_t$ and the walls temperature $\boldsymbol{\Theta}_t^w$, both assumed constant during $[t, t+\Delta t[$. As introduced in equations (7.6) the thermal gains display a statistical dynamic equation. They constitute an *informational state*. We call

$$\boldsymbol{W}_t = (\boldsymbol{R}_t, \boldsymbol{R}_t^w)^T \ , \tag{7.7}$$

the vector of residuals, also called exogenous noise, which has the same law as $(\boldsymbol{R}, \boldsymbol{R}^w)$ with support $\mathbb{W}_t$.

And we introduce the state of the system at time stage t,

$$\boldsymbol{X}_t = (\boldsymbol{\Theta}_t, \boldsymbol{\Theta}_t^w, \boldsymbol{\Phi}_t, \boldsymbol{\Phi}_t^w)^T \ , \tag{7.8}$$

that belongs to the set $\mathbb{X}_t$. Finally we define the dynamic $f_t : \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1} \to \mathbb{X}_{t+1}$ such that for all $(x, u, w) \in \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1}$,

$$f_t(x,u,w) = \begin{pmatrix} \theta_t + \frac{\Delta t}{c_i}\Big(p_1\big(\theta_t^w - \theta_t\big) + p_2\big(\theta_t^o - \theta_t\big) + \frac{p_3}{\Delta t}u_t + a\phi_t + b + r_t\Big) \\ \theta_t^w + \Delta t\Big(p_4\big(\theta_t - \theta_t^w\big) + p_5\big(\theta_t^o - \theta_t^w\big) + \frac{p_6}{\Delta t}u_t + a^w\phi_t^w + b^w + r_t^w\Big) \\ a\phi_t + b + r_t \\ a^w\phi_t^w + b^w + r_t^w \end{pmatrix} . \tag{7.9}$$

7.3.3 Optimal controls are searched as state feedbacks

We introduce $\mathcal{F}_t$ the sigma algebra generated by all the past exogenous noises at time step t:

$$\mathcal{F}_t = \sigma(\boldsymbol{W}_0, \dots, \boldsymbol{W}_t) \; . \tag{7.10}$$

To model the fact that the decision maker takes decisions knowing only past uncertainties realization we introduce the so called non anticipativity constraint

$$\sigma(\boldsymbol{U}_t) \subset \sigma(\boldsymbol{W}_0, \dots, \boldsymbol{W}_t) \; , \tag{7.11a}$$

or its equivalent functional form

$$\exists \pi_t : \mathbb{W}_0, \dots, \mathbb{W}_t, \quad \boldsymbol{U}_t = \pi_t(\boldsymbol{W}_0, \dots, \boldsymbol{W}_t) \; . \tag{7.11b}$$

When the exogenous random variables are stage-wise independent it is enough to restrict the search to state feedback that is function of the state of the system $\boldsymbol{X}_t$:

$$\sigma(\boldsymbol{U}_t) \subset \sigma(\boldsymbol{X}_t) \; , \tag{7.12a}$$

or equivalently

$$\exists \pi_t : \mathbb{X}_t \to \mathbb{U}_t, \quad \boldsymbol{U}_t = \pi_t(\boldsymbol{X}_t) \; . \tag{7.12b}$$

This is the so called Markov condition. In this instance we modeled the exogenous random variables $\boldsymbol{R}$ and $\boldsymbol{R}^w$ in (7.6) making a stationary assumption as we quantified all the occurrences observed in the available measures.

7.3.4 The objective is an expected daily cost

The objective is to minimize the cost of electricity consumed by the heater over a day while maintaining the temperature within a comfort zone (around $\theta_{ref} = 295K$). We consider an electricity tariff known in advance c_t to be in $euros/kWh$. The objective is the following

$$\min_{\boldsymbol{U}_0, \dots, \boldsymbol{U}_{T-1}} \mathbb{E}\Big[\sum_{t=0}^{T} c_t \times \boldsymbol{U}_t \Big] \; . \tag{7.13}$$

To simplify the computation we consider that the temperature constraint is soft, mathematically it materializes as a constraint in expectation as follows

$$\mathbb{E}\Big(\sum_{t=0}^{T} |\boldsymbol{\Theta}_t - \theta_{ref}| - M \Big) \leqslant 0 \; , \tag{7.14}$$

with an arbitrary constant M. This constraint states that during the day we don't want to be too far from the reference temperature θ_{ref}. We dualize this constraint to solve the problem. A better way would be to find the optimal multiplier associated to the constraint. However to simplify the instance, we fix a marginal price λ of temperature by trial and errors. It finally leads to the following objective

$$\min_{\boldsymbol{U}_0, \dots, \boldsymbol{U}_{T-1}} \mathbb{E}\Big[\sum_{t=0}^{T-1} c_t \times \boldsymbol{U}_t + \lambda |\boldsymbol{\Theta}_t - \theta_{ref}| + \lambda |\boldsymbol{\Theta}_T - \theta_{ref}| \Big] \; . \tag{7.15}$$

We call $L_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) = c_t \times \boldsymbol{U}_t + \lambda |\boldsymbol{\Theta}_t - \theta_{ref}|$ and $K(\boldsymbol{X}_T) = \lambda |\boldsymbol{\Theta}_T - \theta_{ref}|$. Now we have all the ingredients to state a stochastic optimization problem.

7.3.5 Stochastic optimization problem statement

Gathering all the above equations we state a stochastic optimal control problem:

$$\min_{\boldsymbol{U}\in\mathbb{U}} \quad \mathbb{E}\Big[\sum_{t=0}^{T} L_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) + K(\boldsymbol{X}_T)\Big] , \tag{7.16a}$$

$$\text{s.t } \boldsymbol{X}_{t+1} = f_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) , \tag{7.16b}$$

$$\sigma(\boldsymbol{U}_t) \subset \sigma(\boldsymbol{X}_t) . \tag{7.16c}$$

7.3.6 Producing a control policy: Stochastic Dual Dynamic Programming

Problem (7.16) has a linear objective and linear dynamics, therefore we can apply the SDDP algorithm to compute a lower convex polyhedral approximation of the Bellman value functions of problem (7.16), we refer the reader to [176] or Chapter 6 of this manuscript for a broader description of SDDP. We call $\{\widetilde{V}_t\}_{t=0,\dots,T}$ these approximated value functions.

At time t we observe the state of the system x_t. We compute a control solving the small optimization problem (7.17) with $\widetilde{V}_{t+1}$ precomputed and x observed

$$\pi_t(x) \in \underset{u\in\mathbb{U}_t}{\arg\min} \ \mathbb{E}\Big[L_t(\boldsymbol{X}_t, u, \boldsymbol{W}_{t+1}) + \widetilde{V}_{t+1}\Big(f_t(\boldsymbol{X}_t, u, \boldsymbol{W}_{t+1})\Big)\Big] . \tag{7.17}$$

As $\widetilde{V}_{t+1}$ is convex polyhedral this problem can be solved using linear programming. $(\pi_0, \dots, \pi_{T-1})$ is the control policy of the house. To implement this policy on a real system we need essentially four features:

1. a storage of the cost functions $\{L_t\}_{t=0,\dots,T-1}$,
2. a storage of the control spaces $\{\mathbb{U}_t\}_{t=0,\dots,T-1}$,
3. a storage of the value functions $\{\widetilde{V}_t\}_{t=0,\dots,T}$,
4. a linear programming solver to solve (7.17) online.

We now describe how to implement this control policy as well as our calibration strategy on a real house.

7.4 Application to a real house

We apply the calibration and optimal control strategy to a real house. This house is located inside the Facility of Excellence, Sense City, located near Paris in France. Sense City is a mini city that aims at testing novel instrumentation strategies for future cities. It provides a test bed for experiments such as monitoring of urban dynamics and control of urban environment. Sense City contains a moving climate chamber that covers a surface of $400m^2$. It can cover an empty space of a "mini city" area containing a 2 floors building and a house (see Figure 7.1). The climate chamber can produce any requested weather in therms of solar irradiance, outside temperature, humidity and so on.

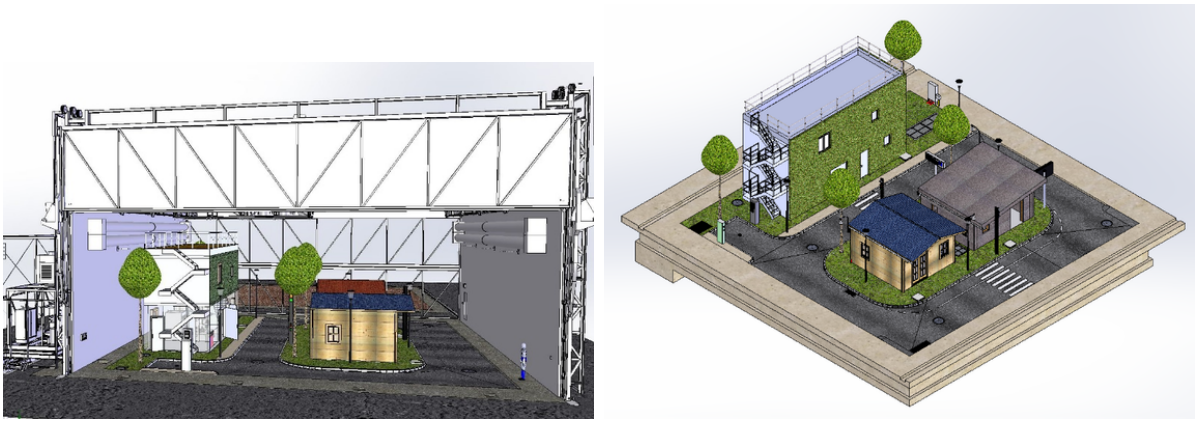

Figure 7.1: Equipment of Excellence “Sense City”

7.4.1 Description of the chalet in the equipment Sense City

As a first step we control the temperature inside the chalet. We display here under a photo of the chalet as well as a schematic representation of the relevant devices inside and outside.

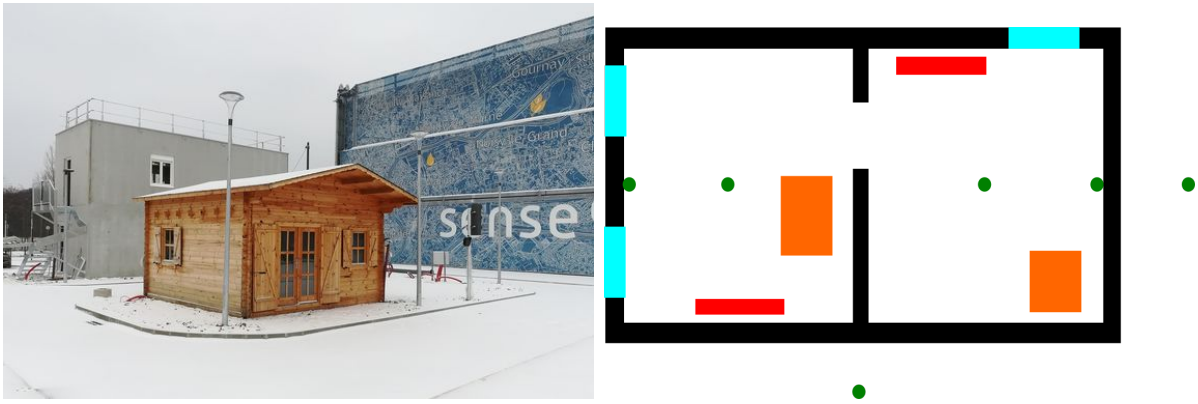

Figure 7.2: The controlled house

On right picture of Figure 7.2 we display a schematic representation of the inside of the house from above. We detail here the signification of colors:

1. Black for the walls,
2. Blue for the windows and door,
3. Red for the electrical heaters,
4. Orange for the computers,
5. Green for temperature sensors.

There are two rooms, each one is equipped with a heater. Each heater is connected to a smart plug that we can switch on or off from the cloud with a GET HTTP request. Each heater produces $800W$ when it is switched on. The smart plugs are also power sensors, hence when a heater is switched on it indicates $800W$

There is one air temperature sensor in each room, four surface temperature sensors on inside and outside faces of the walls and two outside temperature sensors.

In our experiment we provided a week scenario for the temperature inside the climate chamber. The temperature is therefore known in advance. Herein, we reproduce from RT2012 weather file a simplified climate of the region of Carpentras in France during one week in December. However in the house we did not know when people would open the door, stay in the house, use the computers (from the house or from the cloud) or when the solar lights of the chamber would be turned on. That makes the stochastic modeling of exogenous thermal gains relevant in this experiment.

We display on Figure 7.3 the general implementation of our experiment. We detail in the sequel all the components of this implementation.

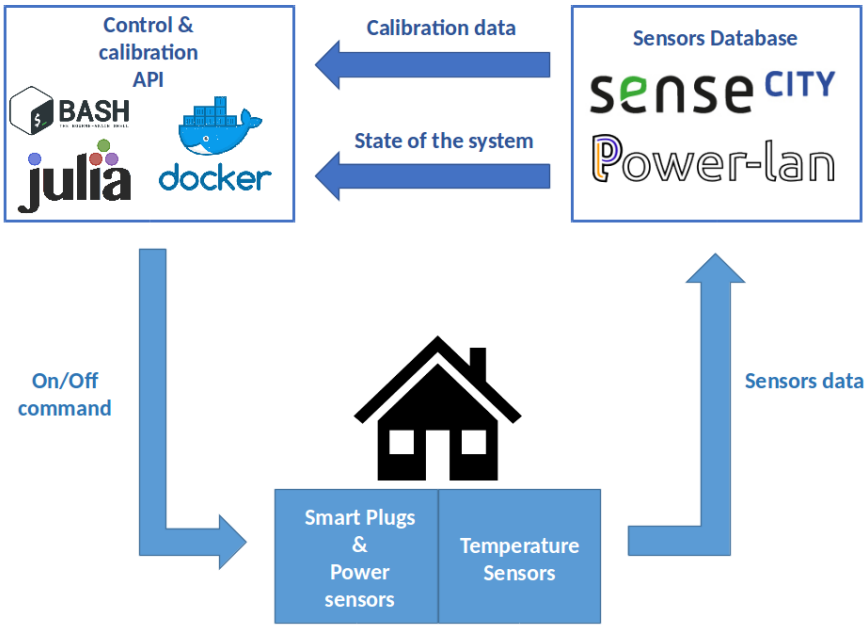

Figure 7.3: Schematic representation of the implementation

7.4.2 Sensors, database and APIs

The temperatures sensors are 4-wire PT100 probes with a precision of about 0.5 Celsius degree. The temperatures inside and outside the chalet is measured every 15 minutes. Then, we consider the smart plugs SONOFF Pow R2 with a Tasmota firmware developed by Powerlan. The smart plugs measure the heating power every minutes. These data are acquired using a PEGASE cardboard, which is a micro computer similar to a Rasberry Pi for industrial applications. These data are sent to a sensors database located in a

Sense City server, we refer to it as the cloud as it is possible to reach it from outside the Sense City network.

The smart plugs use a wifi to send data and receive On of Off commands.

7.4.3 Numerical results of the calibration

We perform the calibration strategy introduced in Section 7.2 with $\Delta t = 900s$ and $T_h = 7$ days. The calibration problem (7.5) is modeled in Julia using the Non Linear Programming package Optim.jl. It requires to solve the linear program (7.4) at each call of the oracle. This linear program is modeled using the Mathematical Programming modeler JuMP and solved with the IBM solver Cplex. The simulated annealing first guess is produced in less than 1 second. The BFGS phase then takes approximately 10 minutes with a Core i7 $-$ 7700k CPU @ 4.20 GHz $\times 8$. Both algorithms are implemented in Optim.jl. We call $(p_1^\sharp, \ldots, p_7^\sharp)$ the parameters obtained after the calibration whose values are $\{100W.K^{-1}.s, 100W.K^{-1}, 7.28W.K^{-1}, 69.8\%, 50.2W.K.s, 89.2W.K^{-1}, 10.0W.K^{-1}\}$. We present in Figure 7.4 the indoor air temperature measurements used for calibration and compare with a simulation over $10,080$ minutes, that is 7 days, of the indoor air temperature using the calibrated parameters $(p_1^\sharp, \ldots, p_7^\sharp)$ and the associated thermal gains computed solving (7.4) with these parameters.

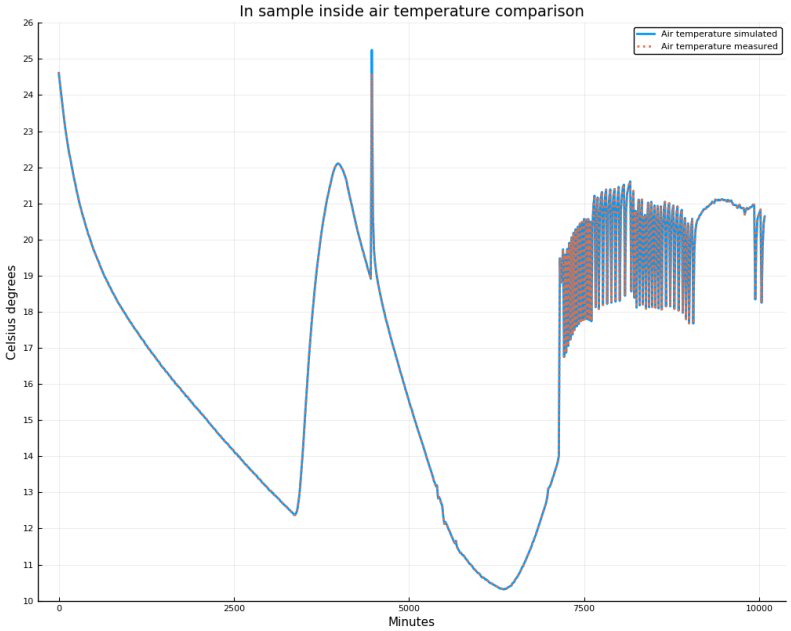

Figure 7.4: In sample assessment of model calibration

We perform the same comparison but this time the reference is measured temperature data that was not used during calibration. Doing so we ensure that the comparison is not biased by an assessment over a training data-set.

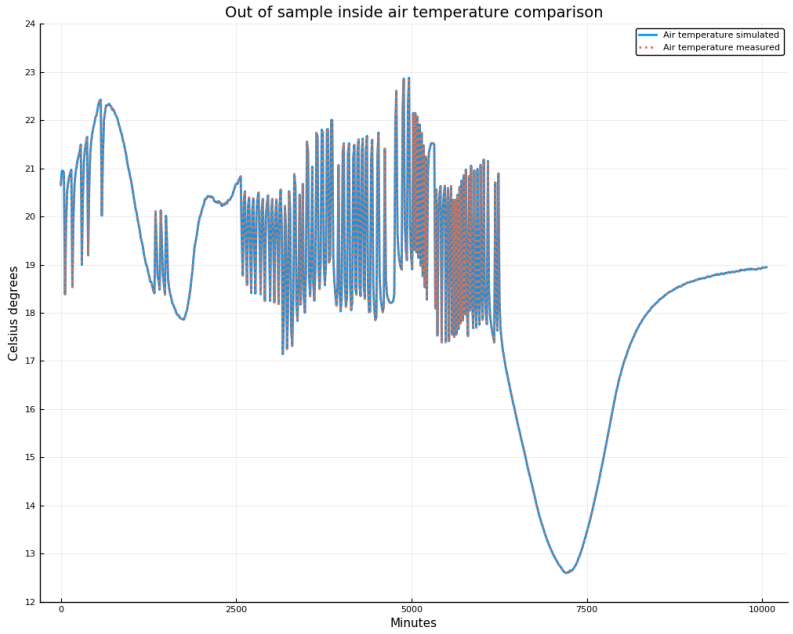

Figure 7.5: Out of sample assessment of model calibration

In both cases the two curves seem indistinguishable, the biggest gap between simulated and measured temperature at the same time step is under 0.5K which we consider satisfactory for our experiment as this is the order of the measurement error. We display on Figure 7.6 the computed exogenous thermal gains as well as the heating power within the out-of-sample data-set.

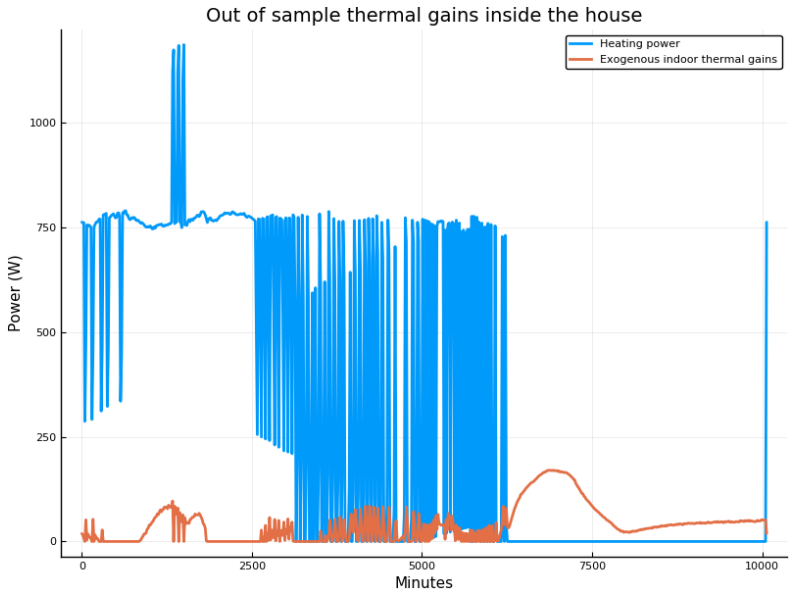

Figure 7.6: Out of sample exogenous thermal gains

We observe that the thermal gains reach a maximum of 150W which could be the

thermal gain of humans or computers in the room. Sometimes the variation of the thermal gains is significantly correlated with the one of heating power, even if exogenous thermal gains power remains under 7% of electrical heating power. This is due to discrepancies between the calibrated model (7.2) and real data. Hence modeling exogenous thermal gains as stochastic variables is also relevant to take into account stochastic model discrepancies.

7.4.4 Energy Management System implementation

As depicted in Section 7.3 the control policy is produced using the SDDP algorithm. Problem (7.16) is modeled using the package DynOpt presented in Chapter 6 of this manuscript. We distinguish two phases for the software implementation of the control of the house:

1. **the offline phase** is dedicated to the calibration of model (7.1) and the computation of the exogenous thermal gains model (7.6). These parameters and this model is then used to compute the value functions of Problem (7.16) using the SDDP algorithm. For instance this phase can be repeated every weeks using any computer that can access the sensors database.

2. **the online phase** at time step t uses the temperatures and power measurements at $t-1$ and t, namely $(\theta_t, \theta_t^w, q_t)$. Doing so it is possible to estimate the last internal gains (ϕ_t, ϕ_t^w) by solving Problem (7.4) over two time steps with the last calibrated parameters $(p_1^\sharp, \ldots, p_7^\sharp)$. Then the obtained state $x_t = (\theta_t, \theta_t^w, \phi_t, \phi_t^w)$ is used to solve the online Problem (7.17) and compute an heating energy for the next 15 minutes.

We present on Figure 7.7 the data pipelines within and between these two phases.

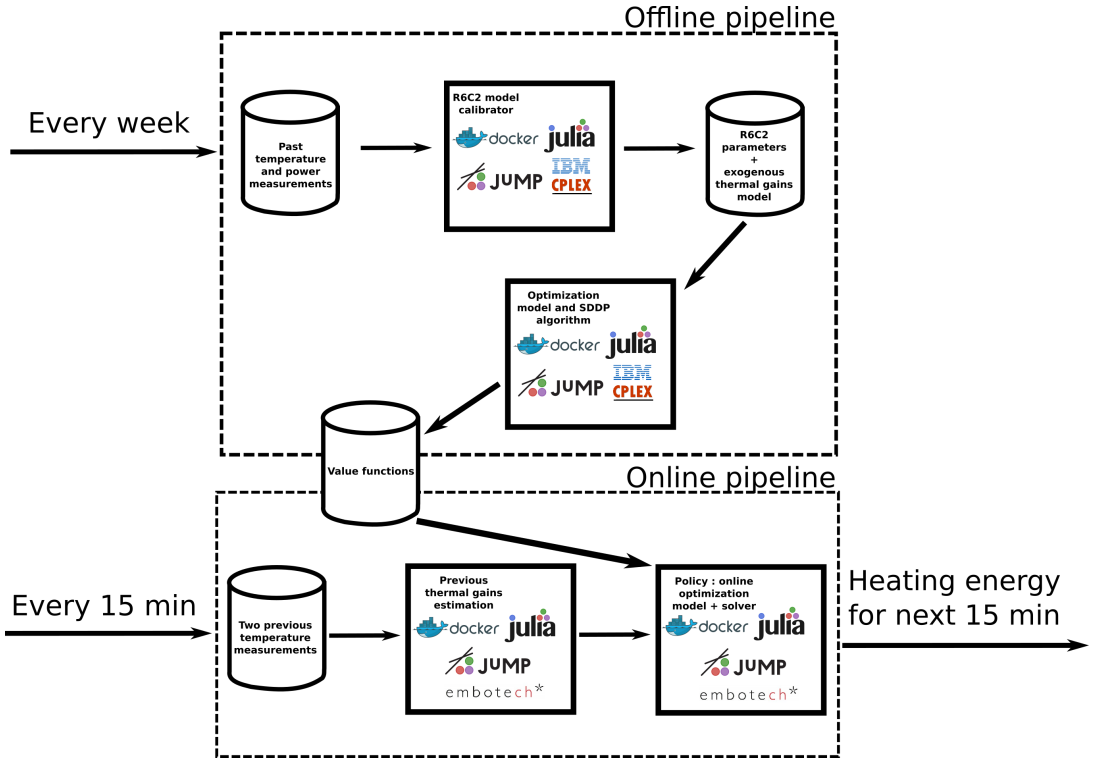

Figure 7.7: Offline and online pipelines

We emphasize that these two phases can be implemented in different computers as long as both machines access a common value functions database. The offline phase requires significant computing performance to calibrate model (7.1) and produce value functions. We used the IBM CPLEX [1] solver to solve problems (7.4) and compute value functions of (7.16) using SDDP. The online phase requires to solve two really small linear programs every 15 minutes, one for the exogenous thermal gains estimation solving (7.4) over two steps and one to compute a control solving (7.17). We use the embeddable conic solver ECOS [2] to solve these problems. With this architecture the offline phase is implemented in a desktop computer while the online phase can be implemented in any Linux based computer even with poor performances.

All our programs are implemented using the Julia language. Every square on Figure 7.7 is a REST API built with the Joseki [3] package and containerized in a Docker [4] image to be deployed in any Linux based computer.

Currently the orchestration of the online phase is performed using a simple Bash script but we are working on a scalable production version orchestrated by Pachyderm [5] and Kubernetes [6].

[1] https://www.ibm.com/analytics/cplex-optimizer
[2] https://www.embotech.com/ecos
[3] https://github.com/amellnik/Joseki.jl
[4] https://www.docker.com/
[5] https://www.pachyderm.io/
[6] https://kubernetes.io/

7.4.5 Low level controller implementation

The online pipeline at time step t provides a heating energy u_t that should be produced by the two 800W electrical heaters in the next 15 minutes. To avoid high variations of temperature during the 15 minutes we control the On/Off switch of the heaters every minutes. We compute $\Delta_t^{on} = \frac{u_t}{2\times 800}$ the amount of time during which the two heaters should be turned on to release the required amount of energy u_t within the 15 minutes, hence $\Delta_t^{on} \leqslant 15 \times 60 = 900$s. Then every minutes we switch on the two heaters during $\frac{\Delta_t^{on}}{15} \leqslant 60$s to dispatch the heating energy evenly across the 15 minutes. We present in Figure 7.4.5 a simplified version of this controller implemented as a bash script.

```
#!/bin/sh
while true
do
    CURRENT_DATE=$(date +'%s')
    NEXT_DATE=$((CURRENT_DATE / 900 * 900 + 900))

    PLUG_TIME=$(curl -s $IP_API/control_sddp/?ti=$START_DATE |
    jq ".result")

    ON_MINUTE_TIME=$((PLUG_TIME / 15))
    CURRENT_DATE=$(date +'%s')

    while [  $CURRENT_DATE -le $NEXT_DATE ]
    do

        curl "$PLUG_IP/cm?cmnd=Power%20on"
        sleep $ON_MINUTE_TIME
        curl "$PLUG_IP/cm?cmnd=Power%20off"

        CURRENT_DATE=$(date +'%s')
        sleep $(( CURRENT_DATE / 60 * 60 + 60 - CURRENT_DATE))
    done
done
```

Figure 7.8: Low level controller bash script

We now comment each line:

1–3 The first two lines ensures that the script is perpetually executed by Bash.

4–6 The variable `CURRENT_DATE` contains the date in seconds in Unix style. `NEXT_DATE` contains the next date which has a number of minutes multiple of 15 minutes. For instance if `CURRENT_DATE` is the Unix style date 12 february 2019 10h37 then `NEXT_DATE` is in Unix style 12 february 2019 10h45.

7–9 The cURL [7] command sends a GET request to the EMS API (see §7.4.4) deployed in a machine whose IP, and exposed port, is stored in the variable `IP_API`. The `control_sddp` endpoint solves Problem (7.17) to compute the amount of energy required for the next 15 minutes and returns the number of second the 1600 W heaters should be turned ON within the next 15 minutes to produce this energy. This value is stored in the `PLUG_TIME` variable. The endpoint requires the value `ti=START_DATE` which is

[7] https://github.com/curl/curl

the initial date from which we computed value functions with the Offline pipeline depicted in § 7.4.4. The EMS API returns data as JSON, hence we use jq [8], a command line JSON parser.

10–12 The variable `ON_MINUTE_TIME` contains a number of seconds between 0 and 60 seconds during which the heaters should be turned ON within a minute. `CURRENT_DATE` is updated to take into account the EMS computation time.

13–15 Until the `NEXT_DATE` where we will compute a new energy for 15 minutes executes the following code.

16–18 Turns ON the smart plug (the heaters) with the first cURL command. Waits `ON_MINUTE_TIME` seconds then turns OFF the smart plug with the second cURL command.

19–21 Waits until the end of the current minute.

We present in the next paragraph, real results showing that this low level controller and EMS combination is able to maintain a prescribed temperature efficiently.

7.4.6 Control results

We display in Figure 7.9 a 24 hour scenario where the house is controlled by our software presented above. In this experiment the temperature should be maintained around 23 Celsius degrees while minimizing the energy consumption to ensure this constraint.

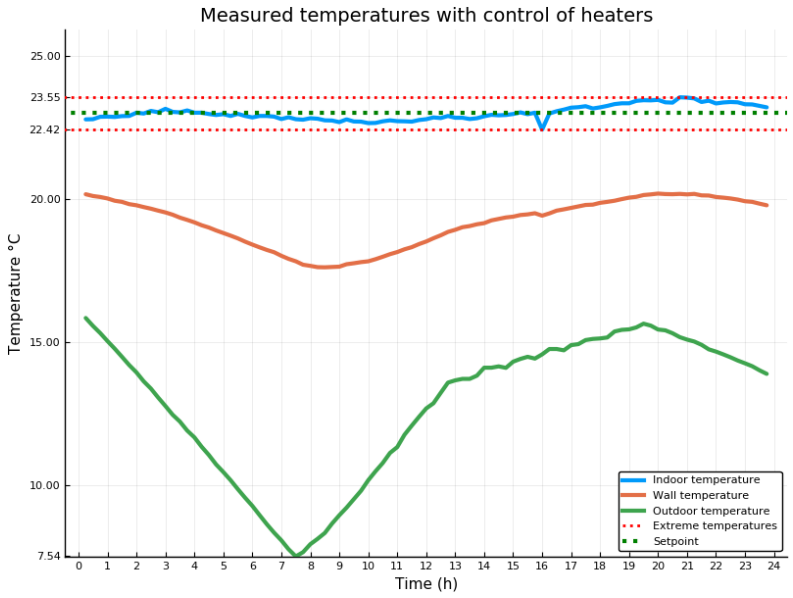

Figure 7.9: Temperature controlled

We observe that the temperature is effectively maintained at the prescribed thermal comfort set-point we chose, 23 Celsius degrees. We present in Figure 7.10 the corresponding heat generated by the heaters every 15 minutes.

[8] https://stedolan.github.io/jq/

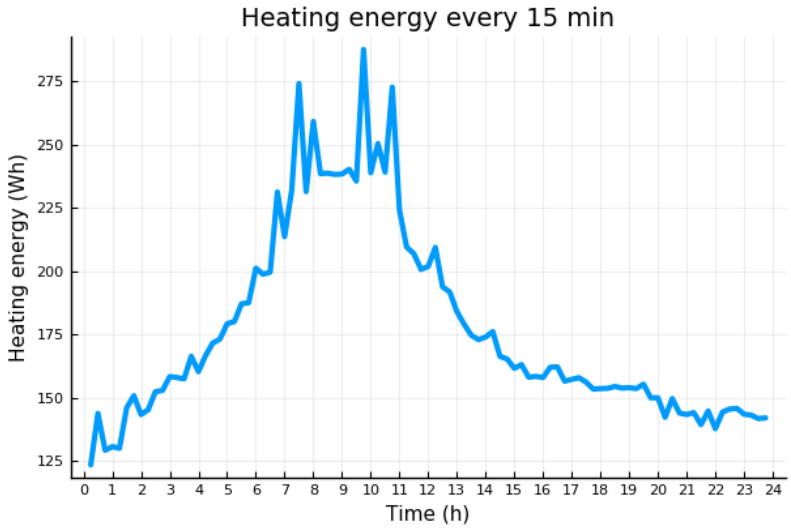

Figure 7.10: Heating energy

This heating strategy consumes 16.8 kWh over 24 hours which costs 2.29 euros with the cost of electricity displayed on the left on Figure 7.11. This cost in based on EDF blue tariff (with off peak hours and peak hours) in 2018 [9]. On the right of Figure 7.11 we present the cumulative expenses during the day.

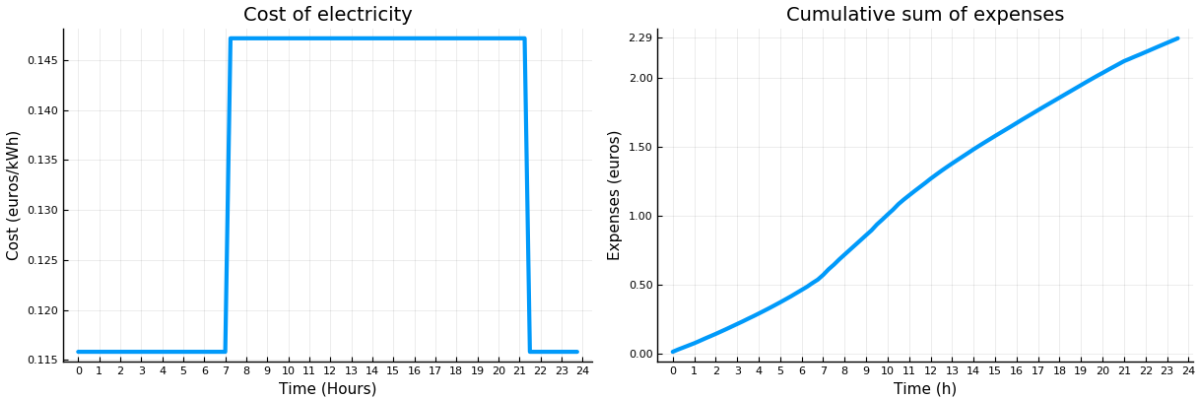

Figure 7.11: Cost of electricity and cumulative expenses

These results demonstrates that the strategy we implemented is able to control the temperature in a house. Now it remains to compare control results with the SDDP algorithm to a regular control strategy such as PID control or a more intelligent one which is Model Predictive Control. We plan to make such a comparison on a same outdoor temperature climate using Sense City climate chamber.

[9] https://particulier.edf.fr/fr/accueil/offres/electricite/tarif-bleu.html

7.5 Conclusion

We presented a method to implement an energy management system for the heating of a house taking into account stochastic thermal gains. Our approach is based on a RC model of the building thermal behaviour with a dedicated calibration strategy that does not require a human input. This calibration strategy provides RC model parameters as well as a stochastic model of the thermal gains that are not related to electrical heaters. Using this calibrated RC model and stochastic model of thermal gains we were able to compute a stochastic optimal control policy to minimize the energy consumption of the house while maintaining an acceptable thermal comfort. We used the Stochastic Dual Dynamic Programming algorithm to compute this policy. We built a software architecture able to reproduce this calibration and policy computation automatically based on sensors data every now and then. We built as well an API hosted in a private cloud to implement this control policy on smart plugs that operate on/off switches on the electrical heaters of the house. We presented results showing the effectiveness of the approach to control a real house temperature in the Equipment "Sense-City" with the SDDP algorithm without a human input. It remains to compare to regular control methods to demonstrate economical and environmental benefits. This study is a first step toward building a plug and play energy management system for the heating of buildings. We need to implement and assess the approach on a wide variety of buildings with longer datasets to continue developing this work. The advantage of the SDDP algorithm is to be much more efficient than a regular SDP for problems with numerous state variables. Our next step is to implement such a strategy on multiple buildings to control the energy of a whole small neighborhood.

Chapter 7. Bibliography

[166] N. Artiges. *From instrumentation to optimal predictive control towards buildings energy efficiency*. Theses, Université Grenoble Alpes, Jan. 2016.

[167] R. Bellman. *Dynamic Programming*. Princeton University Press, New Jersey, 1957.

[168] T. Berthou. *Development of building models for load curve forecast and design of energy optimization and load shedding strategies*. Theses, Ecole Nationale Supérieure des Mines de Paris, Dec. 2013.

[169] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, 1995.

[170] D. P. Bertsekas. *Dynamic Programming and Optimal Control: Approximate Dynamic Programming*. Athena Scientific, fourth edition, 2012.

[171] J. Brouns. *Development of numerical tools for building energy audit*. Theses, Université Paris-Est, Dec. 2014.

[172] F. Derkx, B. Lebental, T. Bourouina, F. Bourquin, C.-S. Cojocaru, E. Robine, and H. Van Damme. The Sense-City project. In *XVIIIth Symposium on Vibrations, Shocks and Noise*, page 9p, France, July 2012.

[173] A. Le Mounier. *Meta-optimisation for automatic calibration for building energetic models in order to proceed to anticipative management*. Theses, Université Grenoble Alpes, June 2016.

[174] D. Lee, S. Lee, P. Karava, and J. Hu. Approximate dynamic programming for building control problems with occupant interactions. In *2018 Annual American Control Conference (ACC)*, pages 3945–3950, June 2018.

[175] F. Oldewurtel, A. Parisio, C. N. Jones, M. Morari, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and K. Wirth. Energy efficient building climate control using stochastic model predictive control and weather predictions. In *Proceedings of the 2010 American Control Conference*, pages 5100–5105, June 2010.

[176] F. Pacaud, P. Carpentier, J.-P. Chancelier, and M. De Lara. Stochastic optimal control of a domestic microgrid equipped with solar panel and battery. preprint, Jan. 2018.

[177] A. Parisio, M. Molinari, D. Varagnolo, and K. H. Johansson. *Energy Management Systems for Intelligent Buildings in Smart Grids*, pages 253–291. Springer International Publishing, Cham, 2018.

[178] A. Parisio and S. Pacheco Gutierrez. Distributed model predictive control for building demand side management. In *European Control Conference*, 2 2018.

[179] P. Pflaum. *Energy management strategies for smart grids*. Theses, Gipsa-Lab ; Schneider Electric, Jan. 2017.

[180] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[181] M. Robillart. *Study of real time control strategies for energy efficient buildings*. Theses, Ecole Nationale Supérieure des Mines de Paris, Sept. 2015.

[182] M. Robillart, P. Schalbart, F. Chaplais, and B. Peuportier. Model reduction and model predictive control of energy-efficient buildings for electrical heating load shifting. *Journal of Process Control*, Apr. 2018.

Conclusion

Contributions

Our main contributions are the following, ordered by chapters.

Chapter 1 presents an abstract formalism for time decomposition of multi-stage stochastic optimization problems. It first introduces the most general way to define such a problem. Then it presents a time blocks decomposition method deeply inspired by Bellman equation. It is an opportunity to formally decompose two novel classes of problems namely, two time scales and decision-hazard-decision problems. These time blocks decomposition methods provide an abstract formalism that could allow to mix different Stochastic Dynamic Programming (SDP) [184] methods with the Stochastic Programming (SP) [188] ones. This chapter opens the door to the following ones as they all make use of time decomposition in a way or another.

Chapter 2 frames different methods to compute online control policies for stochastic dynamical systems. These methods are classified around the Bellman equation, that is a 1 stage time decomposition. Our main contribution is to exhibit the link between different well known methods such as SDP, Model Predictive Control (MPC) [183] or SP. It brings a special view on how information is used both offline and online to produce and compute online policies.

Chapter 3 compares different stochastic optimal control methods on a realistic energy management example. Its main purposes is to show that SDP methods are easier to implement and perform better than classical MPC methods. It also shows a way to increase the energy efficiency of subway stations by recovering unexploited energy (subways regenerative braking) and controlling an energy consuming equipment (ventilation) in a demand-side management fashion.

Chapter 4 present a MPC strategy to stabilize the voltage in a DC micro-grid. It is in particular an opportunity to introduce extensively the hierarchical architecture of micro grids and an high level Energy Management System interacts effectively with the equipment of a micro grid.

Chapter 5 present time decomposition algorithms that can be used to solve two time scales problems but also regular problems with a large number of time steps. We present three different application on a battery long term aging problem. The first one shows the economical benefits of controlling long term aging of a battery in a micro grid. The second one shows how the developed algorithms can be used to solve mono scale problems with a large number of time steps where classical SDP or Stochastic Dual Dynamic Programming (SDDP) [185] are limited. Finally the last one shows how one of the algorithms associated with SDDP can be used to compute efficiently an optimal battery sizing.

Chapter 6 presents the Stochastic Optimization Julia library developed during this PhD to solve all the previous energy management problems. Its purpose is to make the library accessible to potential developers and display how it can be used for real energy management applications or existing software.

Chapter 7 shows how we can apply technically stochastic optimization algorithms to control the temperature in a real house.

Perspectives

We present hereunder different perspectives that can follow our work.

As we did for MPC and SDP in Chapter 3, it would be interesting to compare Reinforcement Learning methods to the stochastic optimization ones we implemented for energy management.

In chapter 4 we apply MPC to voltage stability of a DC micro grid. This MPC solves a Mixed Integer Quadratic Program, minimizing the gap between variables of the system and set-points that are assumed provided by an upper level controller. The binary variables come from the constraint forbidding simultaneous charge and discharge of the battery. With an economic objective we prove in chapter 5 that this constraint can be removed. It would be interesting to apply a MPC without set-points but with value functions computed by a upper level to remove this constraint. Moreover it would be consistent with the two time scales modeling that we introduce in chapter 5.

The batteries aging model we use in chapter 5 only accounts for the cycles that the battery makes. It does not take into account calendar aging as well as temperature and depth of discharge. More precise models are required for a more accurate sizing and management of the equipment. It would probably be more challenging to apply our two time scales algorithms to these more detailed models.

In chapter 5 we state that our two time scales algorithms could be useful to improve the performance of time sensitive algorithms such as SDP, SDDP and stochastic programming methods such as Progressive Hedging [187]. To validate this point we should perform more extensive numerical experiments.

We applied chapter 5 methods to a real project in simulation. We would like to apply our energy management strategies to a real collective self consumption project and validate our sizing with detailed micro grid simulation software.

Chapter 5 presents time decomposition methods for problems with a large number of time steps. Under some stationary assumption we could compare our methods to infinite horizon value and policy iterations ones. We made some numerical experiments suggesting that the value functions obtained with Average Cost Value Iteration are pretty similar to the value functions obtained with our methods on a problem with a large number of time steps. However our methods allow to compute value functions in a deterministic time and with a relaxation of the stationary assumption. It would be interesting to exhibit a formal link between our methods and infinite horizon value iteration.

Chapter 6 introduces the Mixed Integer Dynamic Approximation Scheme (MIDAS) [186] implemented in DynOpt and a method to speed up its convergence with SDDP. We would like to apply the time decomposition algorithms introduced in chapter 5 to speed up further this algorithm. We would also like to extend it with other MIP representations of

nonlinear functions.

In chapter 7 we apply our control strategy to a small building. We will apply it to wider micro grid demonstrators.

Conclusion bibliography

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

[184] P. Carpentier, J.-P. Chancelier, G. Cohen, and M. De Lara. *Stochastic Multi-Stage Optimization. At the Crossroads between Discrete Time Stochastic Control and Stochastic Programming*. Springer-Verlag, Berlin, 2015.

[185] M. V. Pereira and L. M. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1-3):359–375, 1991.

[186] A. Philpott, F. Wahid, and F. Bonnans. MIDAS: A Mixed Integer Dynamic Approximation Scheme. Research report, Inria Saclay Ile de France, June 2016.

[187] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147, 1991.

[188] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

Table of bibliographies

Figure 7.12: Table of bibliographies

General bibliography

- M. R. Almassalkhi and I. A. Hiskens. Model-Predictive Cascade Mitigation in Electric Power Systems With Storage and Renewables; Part I: Theory and Implementation. *IEEE Transactions on Power Systems*, 30(1):67–77, Jan 2015.
- M. R. Almassalkhi and I. A. Hiskens. Model-Predictive Cascade Mitigation in Electric Power Systems With Storage and Renewables; Part II: Case-Study. *IEEE Transactions on Power Systems*, 30(1):78–87, Jan 2015.
- Anses. Pollution chimique de l'air des enceintes de transports ferroviaires souterrains et risques sanitaires associés chez les travailleurs. Technical report, Agence nationale de sécurité sanitaire de l'alimentation, de l'environnement et du travail, 2015.
- N. Artiges. *From instrumentation to optimal predictive control towards buildings energy efficiency*. Theses, Université Grenoble Alpes, Jan. 2016.
- K. Barty, P. Carpentier, and P. Girardeau. Decomposition of large-scale stochastic optimal control problems. *RAIRO Operations Research*, 44(3):167–183, 2010.
- R. F. Bastos, T. Dragicevic, J. M. Guerrero, and R. Q. Machado. Decentralized control for renewable DC Microgrid with composite energy storage system and UC voltage restoration connected to the grid. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 2016–2021, Oct 2016.
- R. Bellman. *Dynamic Programming*. Princeton University Press, New Jersey, 1957.
- T. Berthou. *Development of building models for load curve forecast and design of energy optimization and load shedding strategies*. Theses, Ecole Nationale Supérieure des Mines de Paris, Dec. 2013.
- D. P. Bertsekas. Convergence of discretization procedures in dynamic programming. *IEEE Transactions on Automatic Control*, 20(3):415–419, June 1975.
- D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, third edition, 2005.
- D. P. Bertsekas. Dynamic programming and suboptimal control: A survey from ADP to MPC. *European Journal of Control*, 11(4-5):310–334, 2005.
- D. P. Bertsekas. *Dynamic Programming and Optimal Control: Approximate Dynamic Programming*. Athena Scientific, fourth edition, 2012.

- D. P. Bertsekas and S. E. Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Athena Scientific, Belmont, Massachusetts, 1996.
- D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.
- D. P. Bertsekas and H. Yu. A unifying polyhedral approximation framework for convex optimization. *SIAM Journal on Optimization*, 21(1):333–360, 2011.
- J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *CoRR*, abs/1411.1607, 2014.
- J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.
- A. Bidram, A. Davoudi, F. L. Lewis, and J. M. Guerrero. Distributed Cooperative Secondary Control of Microgrids Using Feedback Linearization. *IEEE Transactions on Power Systems*, 28(3):3462–3470, 2013.
- J. Bonnans, Frederic, D. Giorgi, V. Grelard, B. Heymann, S. Maindrault, P. Martinon, O. Tissot, and J. Liu. Bocop – A collection of examples. Technical report, INRIA, 2017.
- J. Borwein and A. S. Lewis. *Convex analysis and nonlinear optimization: theory and examples*. Springer Science & Business Media, 2010.
- S. Bracco, F. Delfino, F. Pampararo, M. Robba, and M. Rossi. A dynamic optimization-based architecture for polygeneration microgrids with tri-generation, renewables, storage systems and electrical vehicles. *Energy Conversion and Management*, 96:511 – 520, 2015.
- J. Brouns. *Development of numerical tools for building energy audit*. Theses, Université Paris-Est, Dec. 2014.
- E. F. Camacho and C. Bordons. *Model predictive control*. Springer, 2007.
- E. F. Camacho, T. Samad, M. Garcia-Sanz, and I. Hiskens. Control for renewable energy and smart grids. Grand Challenges for Control, 2010.
- F. E. Camelli, G. Byrne, and R. Löhner. Modeling subway air flow using cfd. *Tunnelling and Underground Space Technology*, 43(Supplement C):20 – 31, 2014.
- P. Carpentier, J.-P. Chancelier, G. Cohen, and M. De Lara. *Stochastic Multi-Stage Optimization. At the Crossroads between Discrete Time Stochastic Control and Stochastic Programming*. Springer-Verlag, Berlin, 2015.
- P. Carpentier, J.-P. Chancelier, M. De Lara, F. Pacaud, and T. Rigaut. A template to design online policies for multistage stochastic optimization problems. working paper, Jan. 2019.
- P. Carpentier, J.-P. Chancelier, M. De Lara, and T. Rigaut. Time blocks decomposition of multistage stochastic optimization problem. 2018.

- P. Carpentier, J.-P. Chancelier, V. Leclère, and F. Pacaud. Stochastic decomposition applied to large-scale hydro valleys management. *European Journal of Operational Research*, 270(3):1086 – 1098, 2018.
- J.-P. Chancelier and M. De Lara. Fenchel-moreau conjugation inequalities with three couplings and application to stochastic bellman equation. *arXiv preprint arXiv:1804.03034*, 2018.
- F. Delfino, R. Minciardi, F. Pampararo, and M. Robba. A Multilevel Approach for the Optimal Control of Distributed Energy Resources and Storage. *IEEE Transactions on Smart Grid*, 5(4):2155–2162, July 2014.
- C. Dellacherie and P. Meyer. *Probabilités et potentiel*. Hermann, Paris, 1975.
- F. Derkx, B. Lebental, T. Bourouina, F. Bourquin, C.-S. Cojocaru, E. Robine, and H. Van Damme. The Sense-City project. In *XVIIIth Symposium on Vibrations, Shocks and Noise*, page 9p, France, July 2012.
- O. Dowson. The policy graph decomposition of multistage stochastic optimization problems. *Optimization Online*, 2018.
- O. Dowson and L. Kapelevich. Sddp. jl: a julia package for stochastic dual dynamic programming. *Optimization Online. URL http://www. optimization-online. org/DB_ HTML/2017/12/6388. html*, 2017.
- T. Dragicevic, X. Lu, J. Vasquez, and J. Guerrero. DC Microgrids-Part II: A Review of Power Architectures, Applications, and Standardization Issues. *Power Electronics, IEEE Transactions on*, 31(5):3528–3549, May 2016.
- T. Dragicevic, J. Vasquez, J. Guerrero, and D. Skrlec. Advanced LVDC Electrical Power Architectures and Microgrids: A step toward a new generation of power distribution networks. *Electrification Magazine, IEEE*, 2(1):54–65, March 2014.
- I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.
- I. V. Evstigneev. Measurable selection and dynamic programming. *Mathematics of Operations Research*, 1(3):267–272, 1976.
- H. Farhangi. The path of the smart grid. *Power and Energy Magazine, IEEE*, 8(1):18–28, January 2010.
- M. Farina, A. Guagliardi, F. Mariani, C. Sandroni, and R. Scattolini. Model predictive control of voltage profiles in mv networks with distributed generation. *Control Engineering Practice*, 34(Supplement C):18 – 29, 2015.
- C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

- A. Garulli, S. Paoletti, and A. Vicino. Models and Techniques for Electric Load Forecasting in the Presence of Demand Response. *IEEE Transactions on Control Systems Technology*, 23(3):1087–1097, May 2015.
- H. Gevret, J. Lelong, and X. Warin. STochastic OPTimization library in C++. Research report, EDF Lab, Sept. 2016.
- A. J. Gillespie, E. S. Johanson, and D. T. Montvydas. Energy storage in pennsylvania: Septa's novel and innovative integration of emerging smart grid technologies. *IEEE Vehicular Technology Magazine*, 9(2):76–86, 2014.
- A. González-Gil, R. Palacin, and P. Batty. Sustainable urban rail systems: Strategies and technologies for optimal management of regenerative braking energy. *Energy conversion and management*, 75:374–388, 2013.
- A. González-Gil, R. Palacin, P. Batty, and J. Powell. A systems approach to reduce urban rail energy consumption. *Energy Conversion and Management*, 80:509–524, 2014.
- D. Grange and S. Host. Pollution de l'air dans les enceintes souterraines de transport ferroviaire et santé. Technical report, Observatoire régional de santé Île-de-France, 2012.
- W. Greenwell and A. Vahidi. Predictive Control of Voltage and Current in a Fuel Cell-Ultracapacitor Hybrid. *IEEE Transactions on Industrial Electronics*, 57(6):1954–1963, June 2010.
- J. Guerrero, J. Vasquez, J. Matas, L. de Vicuna, and M. Castilla. Hierarchical Control of Droop-Controlled AC and DC Microgrids; A General Approach Toward Standardization. *Industrial Electronics, IEEE Transactions on*, 58(1):158–172, Jan 2011.
- P. Haessig, H. B. Ahmed, and B. Multon. Energy storage control with aging limitation. In *PowerTech, 2015 IEEE Eindhoven*, pages 1–6. IEEE, 2015.
- P. Haessig, T. Kovaltchouk, B. Multon, H. Ben Ahmed, and S. Lascaud. Computing an optimal control policy for an energy storage. In *EuroSciPy 2013, Bruxelles*, 2013.
- P. Haessig, B. Multon, H. Ben Ahmed, S. Lascaud, and L. Jamy. Aging-aware NaS battery model in a stochastic wind-storage simulation framework. In *PowerTech 2013*, pages 1–6, Grenoble, France, June 2013.
- O. Hernández-Lerma and J. B. Lasserre. *Discrete-time Markov control processes: basic optimality criteria*, volume 30. Springer Science & Business Media, 2012.
- B. Heymann, J. F. Bonnans, P. Martinon, F. J. Silva, F. Lanas, and G. Jiménez-Estévez. Continuous optimal control approaches to microgrid energy management. *Energy Systems*, pages 1–19, 2015.
- B. Heymann, J. F. Bonnans, F. Silva, and G. Jimenez. A stochastic continuous time model for microgrid energy management. In *Control Conference (ECC), 2016 European*, pages 2084–2089. IEEE, 2016.

- B. Heymann and P. Martinon. Optimal Battery Aging : an Adaptive Weights Dynamic Programming Algorithm. *Journal of Optimization Theory and Applications*, Aug. 2018.
- B. Heymann, P. Martinon, and F. Bonnans. Long term aging : an adaptative weights dynamic programming algorithm. working paper, July 2016.
- G. O. Inc. Gurobi Optimizer Reference Manual, 2014. http://www.gurobi.com.
- A. Iovine, G. Damm, E. De Santis, and M. D. Di Benedetto. Management controller for a dc microgrid integrating renewables and storages. *IFAC-PapersOnLine*, 50(1):90 – 95, 2017. 20th IFAC World Congress.
- A. Iovine, M. Jimenez Carrizosa, G. Damm, and P. Alou. Nonlinear control for dc microgrids enabling efficient renewable power integration and ancillary services for ac grids. *IEEE Transactions on Power Systems*, pages 1–1, 2018.
- A. Iovine, T. Rigaut, G. Damm, E. D. Santis, and M. D. D. Benedetto. Power management for a dc microgrid integrating renewables and storages. *Control Engineering Practice*, 85:59 – 79, 2019.
- A. Iovine, S. B. Siad, G. Damm, E. De Santis, and M. D. Di Benedetto. Nonlinear control of an AC-connected DC microgrid. In *Industrial Electronics Society, IECON 2016 - 42nd Annual Conference of the IEEE*, 24-27 October 2016.
- A. Iovine, S. B. Siad, G. Damm, E. D. Santis, and M. D. D. Benedetto. Nonlinear control of a dc microgrid for the integration of photovoltaic panels. *IEEE Transactions on Automation Science and Engineering*, 14(2):524–535, April 2017.
- E. Jimenez, M. J. Carrizosa, A. Benchaib, G. Damm, and F. Lamnabhi-Lagarrigue. A new generalized power flow method for multi connected DC grids. *International Journal of Electrical Power and Energy Systems*, 74:329 – 337, 2016.
- M. Jimenez Carrizosa, F. D. Navas, G. Damm, and F. Lamnabhi-Lagarrigue. Optimal power flow in multi-terminal HVDC grids with offshore wind farms and storage devices. *International Journal of Electrical Power and Energy Systems*, 65:291 – 298, 2015.
- P. Kundur, N. J. Balu, and M. G. Lauby. *Power system stability and control*. McGraw-Hill, 1994.
- R. H. Lasseter. Microgrids And Distributed Generation. *Intelligent Automation and Soft Computing*, 16(2):225–234, 2010.
- R. Le Goff Latimier. *Management and Sizing of an Electric Vehicle Fleet Associated with a Photovoltaic Plant : Stochastic and Distributed Co-optimizationStationary Valorisation of Electric Vehicle Batteries taking into account their aging and availibility*. Theses, Université Paris-Saclay, Sept. 2016.
- A. Le Mounier. *Meta-optimisation for automatic calibration for building energetic models in order to proceed to anticipative management*. Theses, Université Grenoble Alpes, June 2016.

- V. Leclère, P. Carpentier, J.-P. Chancelier, A. Lenoir, and F. Pacaud. Exact converging bounds for stochastic dual dynamic programming via fenchel duality. 2018.
- V. Leclère, H. Gérard, F. Pacaud, and T. Rigaut. Stochdynamicprogramming. jl a julia library for multistage stochastic optimization. `https://github.com/JuliaStochOpt/StochDynamicProgramming.jl`, 2017.
- D. Lee, S. Lee, P. Karava, and J. Hu. Approximate dynamic programming for building control problems with occupant interactions. In *2018 Annual American Control Conference (ACC)*, pages 3945–3950, June 2018.
- M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1):193 – 228, 1998. International Linear Algebra Society (ILAS) Symposium on Fast Algorithms for Control, Signals and Image Processing.
- M. Loève. *Probability Theory I*. New York, fourth edition, 1977.
- L. Meng, Q. Shafiee, G. F. Trecate, H. Karimi, D. Fulwani, X. Lu, and J. M. Guerrero. Review on Control of DC Microgrids and Multiple Microgrid Clusters. *IEEE Journal of Emerging and Selected Topics in Power Electronics*, 5(3):928–948, Sept 2017.
- F. Oldewurtel, A. Parisio, C. N. Jones, M. Morari, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and K. Wirth. Energy efficient building climate control using stochastic model predictive control and weather predictions. In *Proceedings of the 2010 American Control Conference*, pages 5100–5105, June 2010.
- D. E. Olivares, C. A. Canizares, and M. Kazerani. A centralized energy management system for isolated microgrids. *IEEE Transactions on Smart Grid*, 5(4):1864–1875, July 2014.
- D. E. Olivares, J. D. Lara, C. A. Canizares, and M. Kazerani. Stochastic-predictive energy management system for isolated microgrids. *IEEE Transactions on Smart Grid*, 6(6):2681–2693, Nov 2015.
- D. E. Olivares, A. Mehrizi-Sani, A. H. Etemadi, C. A. Canizares, R. Iravani, M. Kazerani, A. H. Hajimiragha, O. Gomis-Bellmunt, M. Saeedifard, and R. e. a. Palma-Behnke. Trends in Microgrid Control. *IEEE Trans. Smart Grid*, 5(4):1905–1919, 2014.
- F. Pacaud, P. Carpentier, J.-P. Chancelier, and M. De Lara. Stochastic optimal control of a domestic microgrid equipped with solar panel and battery. preprint, Jan. 2018.
- A. Papavasiliou, Y. Mou, L. Cambier, and D. Scieur. Application of stochastic dual dynamic programming to the real-time dispatch of storage under renewable supply uncertainty. *IEEE Transactions on Sustainable Energy*, 2017.
- A. Parisio, M. Molinari, D. Varagnolo, and K. H. Johansson. *Energy Management Systems for Intelligent Buildings in Smart Grids*, pages 253–291. Springer International Publishing, Cham, 2018.

- A. Parisio and S. Pacheco Gutierrez. Distributed model predictive control for building demand side management. In *European Control Conference*, 2 2018.
- A. Parisio, E. Rikos, and L. Glielmo. A model predictive control approach to microgrid operation optimization. *IEEE Transactions on Control Systems Technology*, 22(5):1813–1827, Sept 2014.
- A. Parisio, E. Rikos, and L. Glielmo. Stochastic model predictive control for economic/environmental operation management of microgrids: An experimental case study. *Journal of Process Control*, 43:24–37, 2016.
- M. V. Pereira and L. M. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1-3):359–375, 1991.
- V. Perelmuter. *Electrotechnical Systems: Simulation with Simulink and SimPowerSystems*. CRC Press, 2012.
- P. Pflaum. *Energy management strategies for smart grids*. Theses, Gipsa-Lab ; Schneider Electric, Jan. 2017.
- P. Pflaum, M. Alamir, and M. Y. Lamoudi. Comparison of a primal and a dual decomposition for distributed MPC in smart districts. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pages 55–60. IEEE, 2014.
- A. Philpott, F. Wahid, and F. Bonnans. MIDAS: A Mixed Integer Dynamic Approximation Scheme. Research report, Inria Saclay Ile de France, June 2016.
- W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.
- W. B. Powell. Clearing the jungle of stochastic optimization. *Informs*, pages 109–137, 2014.
- W. B. Powell and S. Meisel. Tutorial on stochastic optimization in energy part ii: An energy storage illustration. *IEEE Transactions on Power Systems*, 31(2):1468–1475, March 2016.
- I. Prodan and E. Zio. A model predictive control framework for reliable microgrid energy management. *International Journal of Electrical Power and Energy Systems*, 61:399 – 409, 2014.
- C. Prud'Homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici. Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods. *Journal of Fluids Engineering*, 124(1):70–80, Nov. 2001.
- M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

- J. Qian, A. R. Ferro, and K. R. Fowler. Estimating the resuspension rate and residence time of indoor particles. *Journal of the Air & Waste Management Association*, 58(4):502–516, 2008.

- C. Rackauckas and Q. Nie. Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1), 2017.

- T. Rigaut, P. Carpentier, J. Chancelier, M. D. Lara, and J. Waeytens. Stochastic optimization of braking energy storage and ventilation in a subway station. *IEEE Transactions on Power Systems*, pages 1–1, 2018.

- A. N. Riseth, J. N. Dewynne, and C. L. Farmer. A comparison of control strategies applied to a pricing problem in retail. *arXiv preprint arXiv:1710.02044*, 2017.

- M. Robillart. *Study of real time control strategies for energy efficient buildings*. Theses, Ecole Nationale Supérieure des Mines de Paris, Sept. 2015.

- M. Robillart, P. Schalbart, F. Chaplais, and B. Peuportier. Model reduction and model predictive control of energy-efficient buildings for electrical heating load shifting. *Journal of Process Control*, Apr. 2018.

- R. T. Rockafellar. Integrals which are convex functionals. *Pacific J. Math.*, 24:525–539, 1968.

- R. T. Rockafellar. Integrals which are convex functionals. II. *Pacific J. Math.*, 39:439–469, 1971.

- R. T. Rockafellar. *Convex analysis*. Princeton university press, 2015.

- R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147, 1991.

- R. T. Rockafellar and R. J.-B. Wets. *Variational Analysis*. Springer Science & Business Media, Berlin, 1998.

- R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

- J. Sandoval-Moreno, G. Besançon, and J. J. Martinez. Model predictive control-based power management strategy for fuel cell/wind turbine/supercapacitor integration for low power generation system. In *Power Electronics and Applications (EPE), 2013 15th European Conference on*, pages 1–10, Sept 2013.

- A. Shapiro. Analysis of Stochastic Dual Dynamic Programming Method. *European Journal of Operational Research*, 209:63–72, 2011.

- A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

- A. Shapiro, W. Tekaya, J. P. da Costa, and M. P. Soares. Final report for technical cooperation between georgia institute of technology and ons–operador nacional do sistema elétrico. *Georgia Tech ISyE Report*, 2012.
- H. Shuai, J. Fang, X. Ai, Y. Tang, J. Wen, and H. He. Stochastic optimization of economic dispatch for microgrid based on approximate dynamic programming. *IEEE Transactions on Smart Grid*, PP(99):1–1, 2018.
- J. S. Stein, W. F. Holmgren, J. Forbess, and C. W. Hansen. Pvlib: Open source photovoltaic performance modeling functions for matlab and python. In *2016 IEEE 43rd Photovoltaic Specialists Conference (PVSC)*, pages 3425–3430, June 2016.
- R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- M. Vaccarini, A. Giretti, L. Tolve, and M. Casals. Model predictive energy control of ventilation for underground stations. *Energy and buildings*, 116:326–340, 2016.
- R. M. Van Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663, 1969.
- J. P. Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, 57:3–57, 2015.
- E. Walther, M. Bogdan, and R. Cohen. Modelling of airborne particulate matter concentration in underground stations using a two size-class conservation model. *Science of The Total Environment*, 607:1313–1319, 2017.
- WHO. *Air Quality Guidelines: Global Update 2005. Particulate Matter, Ozone, Nitrogen Dioxide and Sulfur Dioxide*. World Health Organization, 2006.
- H. S. Witsenhausen. A standard form for sequential stochastic control. *Mathematical Systems Theory*, 7(1):5–11, 1973.
- H. S. Witsenhausen. On policy independence of conditional expectations. *Information and Control*, 28(1):65–75, 1975.
- X. Wu, X. Hu, S. Moura, X. Yin, and V. Pickert. Stochastic control of smart home energy management with plug-in electric vehicle battery energy storage and photovoltaic array. *Journal of Power Sources*, 333:203–212, 2016.
- J. Zou, S. Ahmed, and X. A. Sun. Stochastic dual dynamic integer programming. *Mathematical Programming*, Mar 2018.
- L. E. Zubieta. Are microgrids the future of energy?: DC microgrids from concept to demonstration to deployment. *IEEE Electrification Magazine*, 4(2):37–44, June 2016.