Optimisation des chaînes de production dans l'industrie sidérurgique : une approche statistique de l'apprentissage par renforcement

Matthieu Geist

UNIVERSITE PAUL VERLAINE DE METZ

UFR Mathématiques, Informatique, Mécanique et Automatique
Ecole Doctorale : IAEM Lorraine
Departement : Mathématiques
Spécialité : Mathématiques Appliquées

Thèse

présentée pour l'obtention du titre de

Docteur en Sciences de l'Université Paul Verlaine, Metz

par **Matthieu GEIST**

Optimisation des chaînes de production dans l'industrie sidérurgique : *une approche statistique de l'apprentissage par renforcement*

Soutenue le 9 novembre 2009

Membres du jury

Rapporteurs :	Rémi Munos	Directeur de Recherche INRIA, Lille
	Olivier Sigaud	Professeur, Université Pierre et Marie Curie, Paris
Examinateurs :	Mikhaël Balabane	Professeur, Université Paris 13, Paris
	Patrick Florchinger	Professeur, Université Paul Verlaine, Metz
	Gabriel Fricout	Docteur, ArcelorMittal Research, Maizières-lès-Metz
	Olivier Pietquin	Professeur adjoint, Supélec, Metz (codirecteur de thèse)
	Jean-Claude Vivalda	Directeur de Recherche INRIA, Metz (directeur de thèse)

Supélec
équipe IMS
2 rue Edouard Belin
57070 Metz

ArcelorMittal Research
cluster MC
voie Romaine
57280 Maizières-lès-Metz

INRIA Nancy-Grand Est
équipe-projet CORIDA
ISGMP Bat. A, île du Saulcy
F-57045 Metz Cedex 01

Le hasard est un instrument trop difficile à manier mon élu !
Les dieux eux-mêmes ne s'y sont pas risqués !
- Kiskill, servante des dieux

A mes parents.

Remerciements

Je souhaite tout d'abord remercier chaleureusement mon codirecteur de thèse et ami Olivier PIETQUIN. Il y a quelques années déjà, alors que j'étais encore élève-ingénieur, c'est lui qui m'a montré que la recherche, en fait, c'est rigolo. Cette première impression ne s'est jamais démentie au cours des années qui ont suivi. Bien sûr, je ne peux que le remercier pour sa disponibilité, ses bons conseils, ses bonnes idées, sa capacité étonnante à comprendre mes élucubrations scientifiques et j'en oublie. Mais c'est surtout pour m'avoir fait découvrir ce métier merveilleux que je lui serai à jamais reconnaissant. Grâce à lui, cela fait trois ans que je ne me lève pas pour aller travailler, mais pour aller m'amuser. Et ça, ça n'a pas de prix.

Je tiens également à remercier ArcelorMittal pour avoir financé cette thèse et tout particulièrement Gabriel FRICOUT qui a participé activement à son encadrement. Ses relectures attentives m'ont évité bon nombre d'erreurs et son excellente compréhension de mes travaux, dans un domaine qui n'est pas le sien, a permis une émulsion très constructive concernant les applications sidérurgiques. Sans lui, il m'aurait été beaucoup plus difficile de faire le pont entre les exigences académiques et industrielles.

Je remercie également mon directeur de thèse Jean-Claude VIVALDA pour la confiance et la liberté qu'il m'a accordé.

Je souhaite aussi vivement remercier mes rapporteurs Rémi MUNOS et Olivier SIGAUD d'avoir accepté ce rôle et d'avoir pris le temps de lire mon manuscrit. Je suis très honoré que ces deux experts de l'apprentissage par renforcement (pour ne pas dire cadors !) aient apprécié mon travail. Merci également à Mikhaël BALABANE et Patrick FLORCHINGER d'avoir accepté d'être membres de mon jury.

Je ne citerai pas tout le monde, mais je souhaite exprimer un grand merci à toutes les personnes de Supélec et d'ArcelorMittal que j'ai côtoyées quotidiennement. Je garderai un très bon souvenir de cette période et de la qualité indiscutable des rapports que nous avons eu.

Enfin, je souhaite remercier tout particulièrement ma famille et mes amis pour leur soutien indéfectible et inconditionnel.

Résumé

L'apprentissage par renforcement est la réponse du domaine de l'apprentissage numérique au problème du contrôle optimal. Dans ce paradigme, un agent informatique apprend à contrôler un environnement en interagissant avec ce dernier. Il reçoit régulièrement une information locale de la qualité du contrôle effectué sous la forme d'une récompense numérique (ou signal de renforcement), et son objectif est de maximiser une fonction cumulante de ces récompenses sur le long terme, généralement modélisée par une fonction dite de valeur. Le choix des actions appliquées à l'environnement en fonction de sa configuration est appelé une politique, et la fonction de valeur quantifie donc la qualité de cette politique. De façon générale, l'agent n'a pas de modèle (ni physique, ni statistique par exemple) de son environnement, ni de la fonction de récompense qui définit l'optimalité du contrôle. Cependant, une hypothèse usuelle que nous faisons est que l'environnement est Markovien, c'est-à-dire que l'effet de l'application d'une action ne dépend que de la configuration courante de l'environnement, et pas du chemin parcouru pour l'atteindre. Ce parangon est très général, et permet de s'intéresser à un grand nombre d'applications, comme la gestion des flux de gaz dans un complexe sidérurgique, que nous abordons dans ce manuscrit. Cependant, sa mise en application pratique peut être difficile. Tout d'abord, lorsque la description de l'environnement à contrôler est trop grande, une représentation exacte de la fonction de valeur (ou de la politique) n'est pas possible. Dans ce cas se pose le problème de la généralisation (ou de l'approximation de fonction) : il faut d'une part concevoir des algorithmes dont la complexité algorithmique ne soit pas trop grande, et d'autre part être capable d'inférer le comportement à suivre pour une configuration de l'environnement inconnue lorsque des situations proches ont déjà été expérimentées. Un autre problème vient du fait que, dans le cas le plus général, l'agent apprend à contrôler l'environnement tout en le contrôlant. Cela se traduit souvent par des phases successives d'évaluation de la qualité d'une politique et d'amélioration de cette dernière. Du point de vue apprentissage, cela induit des non-stationnarités (on évalue la qualité d'une politique qui est constamment modifiée), problème rarement traité dans la littérature comme tel. Cette imbrication de l'apprentissage et du contrôle induit également un problème connu sous le nom de dilemme entre exploration et exploitation. Pour chaque action qu'il choisit, l'agent doit décider entre une action qu'il considère comme optimale par rapport à sa connaissance imparfaite du monde et une autre action, considérée comme sous-optimale, visant à améliorer cette connaissance. Pour traiter ce problème efficacement, il faudrait pouvoir estimer la confiance qu'a l'agent en ses estimations. Si ces différentes difficultés sont connues, les méthodes de la littérature les traitent généralement séparément. Ainsi, une méthode pensée pour traiter du dilemme entre exploration et exploitation ne s'adaptera pas forcément au problème de la généralisation, et inversement.

D'un autre côté, le filtrage de Kalman est un paradigme général qui a pour but de

traquer l'état caché d'un système dynamique, potentiellement non-stationnaire, à partir d'observations indirectes de cet état. Ce cadre de travail présente plusieurs aspects intéressants. Les algorithmes résultants sont du second ordre et ils traquent la solution plutôt qu'ils ne convergent vers elle. Ils permettent également d'obtenir très naturellement une information d'incertitude sur les estimations faites, et des schémas d'approximation efficaces existent pour prendre en compte les non-linéarités. La contribution principale de ce manuscrit est un cadre de travail, nommé *Différences Temporelles de Kalman*, qui exprime le problème de l'approximation de la fonction de valeur (pour laquelle nous adoptons une représentation paramétrique) sous le parangon de Kalman. L'état caché à inférer est alors le vecteur des paramètres permettant de décrire la fonction de valeur, et les observations utilisées pour l'estimer sont les récompenses associées aux transitions dans l'espace d'état, une équation de Bellman servant de liant à ces différentes quantités. Cela permet de bénéficier des avantages propres au filtrage de Kalman, et donc de traiter de front plusieurs des problèmes de l'apprentissage par renforcement mentionnés. Cependant, sous l'hypothèse d'un bruit d'observation blanc, classique et nécessaire à l'obtention des différents algorithmes présentés (dont l'une des variantes est l'un des premiers algorithmes d'ordre deux du type itération de la valeur), nous montrons que l'approche proposée revient à minimiser un résidu quadratique de Bellman, ce qui se traduit par un biais de l'estimateur lorsque l'environnement à contrôler est stochastique. Nous analysons ce problème, et y apportons une solution en introduisant une famille de bruits colorés, qui peut très directement être liée au concept des traces d'éligibilité (concept commun en apprentissage par renforcement). Nous illustrons également l'utilisation de l'information d'incertitude disponible en présentant une forme d'apprentissage actif ainsi qu'un ensemble de méthodes visant à traiter le dilemme entre exploration et exploitation. Nous proposons aussi une nouvelle classe d'algorithmes acteur-critique, basés sur la notion de gradient naturel ainsi que sur les différences temporelles de Kalman. Autant que nous le sachions, ce sont les premières approches de ce type qui permettent un apprentissage en ligne, tout en proposant des mises à jour du second ordre tant pour l'acteur que pour le critique.

Table des matières

Table des figures

Liste des Algorithmes

Liste des tableaux

Chapitre 1

Position du problème

Ce chapitre introduit la problématique industrielle qui a motivé nos travaux. Cette dernière implique un certain nombre de contraintes qui nous ont amené à privilégier l'apprentissage par renforcement par rapport à d'autres approches. Nous présentons brièvement les contributions de cette thèse (un résumé plus complet en étant fait dans le dernier chapitre) ainsi que les publications associées, avant de proposer une vue d'ensemble du manuscrit avec un résumé rapide des différents chapitres.

1.1 Problématique industrielle

Le processus de fabrication de l'acier (du minerai brut à la livraison chez le client) implique de nombreuses étapes de production effectuées dans plusieurs usines. Tout au long de ce processus, de nombreuses contraintes hétérogènes doivent être prises en compte. Il y a tout d'abord les contraintes physiques et métallurgiques imposées tout le long du processus, qui elles-mêmes peuvent être très hétérogènes (pression de laminage, températures de recuit, planéité des tôles, *et caetera*). Il y a également de nombreuses contraintes logistiques (transport des produits entre usines, conséquences d'une panne ou d'un retard dans une usine, modification du carnet de commande, *et caetera*). Enfin il y a un certain nombre de contraintes environnementales telles que la minimisation de consommation d'énergie ou de production des déchets.

Pour optimiser ce processus de production dans son ensemble, ces différentes contraintes doivent être considérées simultanément, plusieurs optimisations locales n'entraînant pas nécessairement une optimisation globale. De plus, les différents éléments à optimiser sont très hétéroclites, ce qui suggère l'utilisation d'un modèle le plus générique possible. Il faut également considérer le fait que si certains processus sont bien compris et par conséquent modélisables, d'autres ne peuvent être contrôlés que par des heuristiques (par des machines ou le savoir-faire humain). En fait, il est possible de contrôler certaines machines (de façon automatique), mais pour d'autres il n'y a pas d'alternative à l'expertise humaine.

Dans ce contexte et sur le long terme, le projet `PROFL-213` d'ArcelorMittal Research dans lequel s'inscrivent ces travaux a pour objectifs de :

– développer une méthodologie générale de façon à modéliser aussi globalement et génériquement que possible un large problème d'optimisation, malgré la malédiction de la dimensionnalité[1] ;

[1] La malédiction de la dimensionnalité rend compte de l'explosion combinatoire de la complexité de

- développer des algorithmes qui permettent de trouver une solution au problème (optimale, voire sous-optimale mais de bonne qualité) ;
- utiliser ces algorithmes hors-ligne pour améliorer les routes métallurgiques respectivement aux contraintes mentionnées précédemment et en-ligne pour améliorer la réaction du processus dans le cas d'événements non prévus à l'origine.

Il faut également noter que les problèmes d'optimisation envisagés peuvent être dynamiques, dans le sens où une décision à un moment donné doit être prise en tenant compte de l'évolution future du système et pas seulement en fonction de son état actuel.

1.2 Approche adoptée

Ainsi la problématique posée est celle très générale du contrôle optimal d'un système dynamique, potentiellement stochastique et partiellement observable et dont un modèle n'est pas forcément disponible. Nous montrons dans un premier temps pourquoi les approches classiques de la théorie du contrôle ne nous ont pas semblé adéquates (sans en faire un état de l'art cependant), avant de présenter l'apprentissage numérique de façon générale et l'apprentissage par renforcement plus particulièrement, qui est la voie que nous avons choisi de suivre.

La théorie du contrôle est un domaine étudié depuis fort longtemps et qui traite le comportement des systèmes dynamiques et la façon d'influencer ce dernier. Parmi les exemples les plus connus peuvent être cités les contrôleurs LQG (Linéaire Quadratique Gaussien) ou encore PID (Proportionnel Intégral Dérivatif). Cependant, nous n'avons pas l'ambition de faire un état de l'art de ce domaine. La plupart des approches existantes supposent la connaissance (analytique) du système dynamique, or s'affranchir de modèle *a priori* est une des contraintes posées. Il serait envisageable de paramétrer le système (le modèle pouvant être générique, comme un réseau de neurones par exemple) et d'apprendre les paramètres *ad hoc* à partir de données du système en action. Cependant, c'est un problème difficile en soi (surtout si l'on ajoute une composante contrôle) et nous préférons l'approche par apprentissage plus générale exposée par la suite. De plus, les approches classiques de la théorie du contrôle supposent la linéarité de la dynamique du système, l'extension au cas non-linéaire étant souvent problématique. Les méthodes que nous avons choisi d'étudier présentent peu cette contrainte. Ainsi nous avons choisi de ne pas traiter la problématique posée à partir de méthodes de la théorie du contrôle car l'apprentissage par renforcement nous semble présenter certains avantages indéniables que nous exposerons par la suite. Cependant, ces deux domaines sont loin d'être disjoints : ils traitent des problèmes similaires, avec des approches qui peuvent se ressembler. Ils peuvent être considérés comme des points de vue parfois différents sur un même problème général.

L'apprentissage numérique (ou apprentissage automatique, *Machine Learning* en anglais) consiste de manière générale à apprendre un modèle à partir de données (numériques donc). Il peut être subdivisé en trois branches : apprentissage supervisé, non-supervisé et par renforcement. Un algorithme d'apprentissage supervisé apprendra un modèle de façon à faire correspondre à des entrées des sorties désirées. La régression ou la classification nécessitent souvent de telles approches (la différence entre les deux résidant en la nature des sorties, continues pour la régression et discrètes et en nombre fini pour la classification). En apprentissage non-supervisé, l'algorithme ne dispose que d'entrées pour apprendre ce

l'optimisation lorsque la dimension d'un problème augmente.

modèle, comme c'est le cas pour les méthodes de quantification vectorielle (algorithme des k moyennes, cartes de Kohonen, *neural gas*, *et caetera*). En apprentissage par renforcement, l'algorithme doit apprendre une politique de contrôle par interaction avec le système, une information locale de la qualité du contrôle étant donnée sous forme de récompense et l'objectif étant de maximiser le cumul de récompenses sur le long terme. Cette dernière approche sera développée par la suite. Informellement, en apprentissage non-supervisé il n'est rien dit à la machine (l'algorithme imposant tout de même souvent un *a priori* comme une structure de graphe par exemple). En supervisé il est dit à la machine "comment faire" (base d'exemples composée de couples entrée-sortie) et elle doit pouvoir interpoler à partir des cas observés. Enfin, en renforcement[2] il est dit à la machine combien vaut ce qu'elle a fait (la fonction de récompense) et cette dernière doit trouver "comment faire" (politique de contrôle). Nous développons plus avant cette dernière forme d'apprentissage.

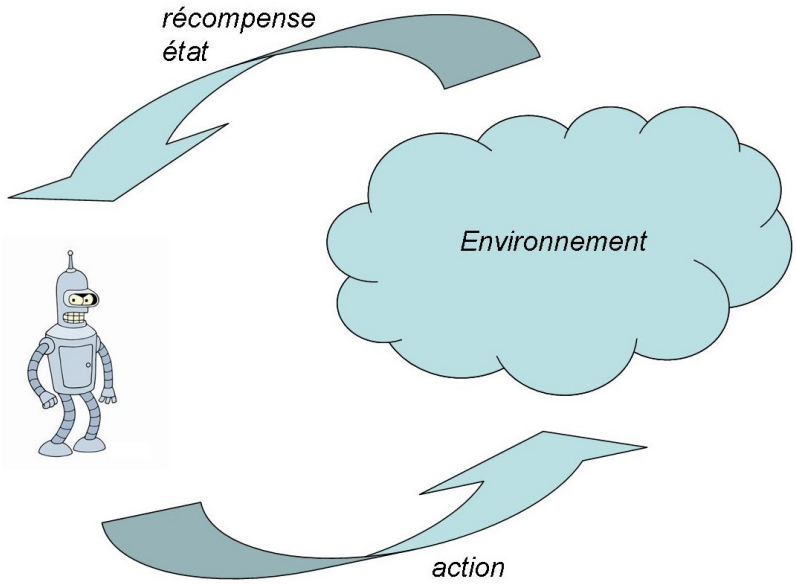


FIG. 1.1 – Paradigme général de l'apprentissage par renforcement.

Le paradigme, illustré sur la figure 1.1, est le suivant. Un agent interagit avec un environnement. A chaque instant (nous considérerons dans tout ce manuscrit une échelle de temps discrète), l'environnement est dans un état donné. L'agent choisit une action parmi un panel disponible. En réponse à cette action, l'environnement (ou de façon équivalente le système dynamique) change d'état en suivant sa propre dynamique, qui peut être stochastique. L'agent observe le nouvel état du système (voire une observation de cet état, par exemple si les capteurs sont bruités) et reçoit une récompense, qui est une information locale de la qualité du contrôle. L'objectif de l'agent est de maximiser le cumul de ces récompenses sur le long terme.

Un exemple simple et classique est celui du robot dans un labyrinthe. L'agent est le contrôleur du robot et l'environnement est composé du robot et du labyrinthe. L'état du système est la position du robot et les actions sont les différents déplacements possibles. L'objectif de l'apprentissage est de faire sortir le robot du labyrinthe, en conséquence une

[2]Le nom de renforcement vient de la psychologie animale, où il a été observé que l'association entre une situation et un comportement fortement récompensé semble "renforcée", alors que les comportements non récompensés sont dissociés des situations dans lesquels ils sont réalisés.

fonction de récompense possible est de donner une "punition" de valeur -1 tant que le robot est dans le labyrinthe et une récompense de 0 lorsqu'il trouve la sortie. En maximisant une fonction cumulante sur le long terme, le robot apprendra à trouver le plus court chemin vers la sortie. Généralement, ce cumul est modélisé en interne par une fonction de valeur associant à un état une valeur espérée du cumul de récompenses en partant de celui-ci.

Ce paradigme est très général. Le contrôle souhaité est spécifié via les récompenses, ce qui est plus universel que le respect d'une consigne, qui est un cas souvent considéré en théorie du contrôle plus classique. Cependant nous n'avons présenté pour l'instant que le paradigme, pas le moyen de le résoudre. Plusieurs approches sont envisageables. Cependant, nous nous concentrerons sur celles basées sur la programmation dynamique. Tout cela sera rendu plus clair dans les chapitres suivants. Remarquons simplement pour l'instant qu'avec ce paradigme très général le cahier des charges posé par la problématique industrielle peut être respecté (même si des solutions pratiques peuvent poser problème).

1.3 Contributions et publications

La contribution principale de cette thèse est l'introduction d'un nouveau cadre de travail général basé sur le filtrage de Kalman pour l'apprentissage de la fonction de valeur (qui associe à un état le cumul de récompenses associé que l'on cherche à maximiser). Cela permet de dériver un certain nombre de nouveaux algorithmes qui présentent des caractéristiques intéressantes par rapport à ceux de l'état de l'art. Une autre contribution est l'application de ces algorithmes à des problèmes rencontrés par l'industrie sidérurgique, notamment un problème de gestion des flux de gaz et nous l'espérons un pas en avant pour l'adoption de ce type d'approches dans l'industrie.

Les travaux effectués durant la thèse ont donné lieu à une certain nombre de publications [37, 41, 38, 39, 40, 46, 47, 42, 44, 45, 48, 43, 49] que nous énumérons ici :

- M. GEIST, O. PIETQUIN, G. FRICOUT, *A Sparse Nonlinear Bayesian Online Kernel Regression*. In Proceedings of the Second IEEE International Conference on Advanced Engineering Computing and Applications in Sciences (AdvComp 2008), I :199-204, Valencia (Spain), (best paper award), 2008.
- M. GEIST, O. PIETQUIN, G. FRICOUT, *Online Bayesian Kernel Regression from Nonlinear Mapping of Observations*. In Proceedings of the 18th IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2008) :309-314, Cancun (Mexico), 2008.
- M. GEIST, O. PIETQUIN, G. FRICOUT, *Filtrage bayésien de la récompense*. In actes des Journées Francophones de Planification, Décision et Apprentissage pour la conduite de systèmes (JFPDA 2008) :113-122, Metz (France), 2008.
- M. GEIST, O. PIETQUIN, G. FRICOUT, *Bayesian Reward Filtering*. In Proceedings of the European Workshop on Reinforcement Learning (EWRL 2008), S. Girgin et al., Springer Verlag, Lecture Notes in Artificial Intelligence, 5323 :96-109, Lille (France), 2008.
- M. GEIST, O. PIETQUIN, G. FRICOUT, *Kalman Temporal Differences : Uncertainty and Value Function Approximation*. In NIPS Workshop on Model Uncertainty and Risk in Reinforcement Learning, Vancouver (Canada), 2008.
- M. GEIST, O. PIETQUIN, G. FRICOUT, *Kalman Temporal Differences : the deterministic case*. In IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2009) :185-192, Nashville, TN, USA, 2009.

- M. GEIST, O. PIETQUIN, G. FRICOUT, *Kernelizing Vector Quantization Algorithms*. In European Symposium on Artificial Neural Networks (ESANN 2009) :541-546, Bruges, Belgium, 2009.
- M. GEIST, O. PIETQUIN, G. FRICOUT, *Différences Temporelles de Kalman*. In actes des Journées Francophones de Planification, Décision et Apprentissage pour la conduite de systèmes (JFPDA 2009), Paris, 2009.
- M. GEIST, O. PIETQUIN, G. FRICOUT, *Différences Temporelles de Kalman : le cas stochastique*. In actes des Journées Francophones de Planification, Décision et Apprentissage pour la conduite de systèmes (JFPDA 2009), Paris, 2009.
- M. GEIST, O. PIETQUIN, G. FRICOUT, *From Supervised to Reinforcement Learning : a Kernel-based Bayesian Filtering Framework*. In International Journal On Advances in Software, ISSN : 1942-2628, 2(1), 2009.
- M. GEIST, O. PIETQUIN, G. FRICOUT, *Tracking in Reinforcement Learning*. In Proceedings of the 16th International Conference on Neural Information Processing (ICONIP 2009), Bangkok (Thailand), 2009.
- M. GEIST, O. PIETQUIN, G. FRICOUT, *Différences temporelles de Kalman : cas déterministe*. In Revue d'Intelligence Artificielle, accepté pour publication, 2009.
- M. GEIST, O. PIETQUIN, G. FRICOUT, *Astuce du Noyau & Quantification Vectorielle*. In Actes du 17$^{\text{ème}}$ colloque sur la Reconnaissance des Formes et l'Intelligence Artificielle (RFIA 2010), Caen (France), 2010.

Le contenu de certaines de ces publications se retrouve dans ce manuscrit [46, 40, 42, 44, 48, 43]. D'autres peuvent être considérées comme des prémisses des travaux présentés (notamment une vision plus basée sur le filtrage bayésien que sur l'approche statistique qui prédomine ici), elles ne sont pas directement présentées ici [39, 38, 37, 41, 45]. Enfin, d'autres publications n'ont qu'un rapport connexe avec les travaux présentés [47, 49]. Tous les résultats de ce manuscrit n'ont pas encore été publiés (notamment les chapitres 5 à 7).

1.4 Vue d'ensemble du manuscrit

Le **chapitre 2** présente un état de l'art de l'apprentissage par renforcement. Nous y introduisons la notion de processus décisionnel de Markov, qui permet la modélisation du système dynamique à contrôler. Basiquement, un tel processus est composé de l'ensemble des états dans lesquels peut se trouver le système, des actions qui peuvent y être appliquées, du modèle de transition entre les états sous l'effet des différentes actions et de la fonction de récompense. Connaissant ce modèle (transitions et récompenses), les algorithmes de programmation dynamique permettent de déterminer la fonction de valeur optimale, qui à chaque état associe le cumul maximal de récompenses espéré. Pour ce faire, ils résolvent l'une des équations dites de Bellman, qui donnent soit la fonction de valeur d'une politique donnée (et l'équation est linéaire), soit directement la fonction de valeur optimale (l'équation correspondante n'est plus linéaire). Notons que plusieurs types de cumuls peuvent être considérés, nous les présentons brièvement. Cependant, nous nous focalisons sur le cumul dit γ-pondéré. Toujours connaissant le modèle, la politique optimale (qui est ce que l'on cherche finalement à obtenir) peut être déduite de la fonction de valeur obtenue à l'aide de l'un de ces algorithmes. Ces approches n'impliquent pas l'apprentissage de la politique optimale via les interactions entre un agent et son environnement, elles résolvent le problème connaissant sa description analytique. L'apprentissage par renforcement, tel que nous le présentons, peut-être vu comme de la programmation dynamique en ligne, asynchrone et

ne nécessitant pas de modèle *a priori* de l'environnement. Les premiers algorithmes que nous présentons (TD, SARSA et Q-learning) permettent de résoudre le cas tabulaire, pour lequel les états et les actions sont en petit nombre, ce qui permet une représentation exacte de la fonction de valeur. Pour ces approches, l'apprentissage se fait en ligne, à partir des interactions entre l'agent et son environnement. Cela se traduit par une imbrication entre contrôle et apprentissage, qui induit certaines problématiques que nous discutons. Notamment, la politique change au cours de l'apprentissage et en conséquence la fonction de valeur associée également. Il vaut donc mieux traquer cette dernière que chercher à converger vers elle. D'autre part, si les états et les actions sont en trop grand nombre, une représentation tabulaire de la fonction de valeur n'est plus possible. Une autre architecture doit être choisie pour représenter cette dernière, qui ne permet pas nécessairement une représentation exacte. Ceci complique l'apprentissage de la fonction de valeur. Le problème sous-jacent, connu sous le nom de généralisation, est un pan important du domaine. Nous en proposons un état de l'art qui présente une vue unifiée des algorithmes les plus classiques (en nous restreignant toutefois à ceux qui permettent un apprentissage en ligne). En supposant une représentation paramétrique de la fonction de valeur, nous exprimons l'ensemble de ces algorithmes sous la forme d'une équation de Widrow-Hoff, c'est-à-dire que les paramètres (qui décrivent la fonction de valeur) sont mis à jour proportionnellement à l'erreur de prédiction effectuée en utilisant l'estimation courante, chaque algorithme différant par le gain considéré pour cette erreur, ainsi que par la façon de l'obtenir. Nous concluons le chapitre en présentant un ensemble de caractéristiques, liées aux problèmes soulevés, que nous pensons devoir être prises en compte par les algorithmes traitant la généralisation. Ces caractéristiques servent de leitmotiv à nos travaux.

Dans le **chapitre 3** nous introduisons les différences temporelles de Kalman, qui constituent la contribution principale de cette thèse. Initialement, le filtrage de Kalman permet d'inférer l'état caché d'un système, potentiellement non-stationnaire, à partir d'observations indirectes de cet état. L'idée de l'approche que nous proposons est d'exprimer le problème de l'approximation de la fonction de valeur sous la forme d'un problème de filtrage : l'état caché à inférer est l'ensemble des paramètres de la fonction de valeur et l'observation qui en est faite est la récompense associée à une transition. Son utilisation est donc différente de la façon dont il est généralement considéré en filtrage, voire en contrôle (contrôleur LQG par exemple). Cela permet de bénéficier des avantages inhérents à l'approche de Kalman : l'état caché est modélisé par une variable aléatoire, cela permet d'obtenir une information d'incertitude sur les paramètres et donc sur l'estimation des valeurs : l'algorithme correspondant traque les paramètres plutôt que de converger vers eux, ce qui permet de s'adapter aux changements de politique dans un contexte d'apprentissage en ligne du contrôle optimal ; cet algorithme est également du second ordre, ce qui permet d'être efficace en termes d'échantillons. De plus, de façon à s'affranchir des conditions de linéarité, nous utilisons un schéma d'approximation, la transformation non-parfumée. Cela permet d'obtenir des algorithmes très généraux, qui ne supposent ni linéarité ni même dérivabilité. Ce dernier point se montre important pour obtenir un algorithme qui permet directement d'obtenir la fonction de valeur optimale, qui s'avère être l'un des premiers algorithmes du second ordre du type itération de la valeur. Nous proposons également une analyse de la classe d'algorithmes obtenus et nous montrons qu'ils minimisent en fait un résidu quadratique de Bellman. Autrement dit, ils cherchent à minimiser l'erreur quadratique entre les deux membres d'une des équations de Bellman, que nous avons brièvement mentionnées auparavant. Pour conclure ce chapitre, nous discutons les aspects de ce nouveau paradigme,

le comparons à certains travaux de la littérature et proposons un jeu d'expérimentations sur des problèmes classiques du domaine, de façon à mettre en avant les caractéristiques propres à l'approche proposée et à la comparer à l'état de l'art.

Les différences temporelles de Kalman partagent un inconvénient avec toutes les autres approches minimisant un résidu quadratique de Bellman : l'estimateur correspondant est biaisé si la dynamique du système à contrôler est stochastique. Dans le **chapitre 4** nous analysons l'effet du caractère stochastique des transitions sur le coût minimisé (ce n'est pas le résidu de Bellman, qui en est une conséquence). Il s'avère que ce biais résulte de l'utilisation d'un bruit d'observation blanc (le bruit d'observation sert à quantifier l'incertitude sur les récompenses observées, il est dit blanc si ses réalisations à différents instants sont indépendantes), hypothèse classique et nécessaire à l'obtention des équations de Kalman, mais trop forte dans le cadre de son utilisation dans un contexte d'approximation de la fonction de valeur. Nous introduisons alors un bruit coloré, qui permet d'obtenir un estimateur non biaisé de la fonction de valeur. Cette extension n'est ni naturelle ni triviale, elle implique de développer une méthode particulière pour pouvoir prendre en compte un bruit coloré à moyenne mobile dans le filtrage de Kalman, ce qui n'a pas été fait auparavant dans la littérature (probablement à cause d'un manque d'applications à ce problème théorique). Ce nouvel algorithme est analysé et nous montrons qu'il tend à minimiser un coût quadratique associant à la valeur de chaque état le cumul de récompenses associé qui serait obtenu par une méthode de Monte Carlo (c'est-à-dire en générant une trajectoire à partir de cet état et en calculant le cumul de récompenses associé). Une série d'expérimentations clôt le chapitre.

Le principe des traces d'éligibilité, que nous présentons au chapitre 2 et qui peut être vu comme une extension des algorithmes de différences temporelles classiques, est de ne pas mettre à jour uniquement la valeur de l'état courant en fonction de l'erreur de prédiction faite, mais de mettre à jour la valeur de tous les états rencontrés sur la trajectoire menant à l'état courant, en fonction d'une pondération (dépendant de l'état mis à jour) de cette erreur. Les algorithmes présentés au chapitre 3 peuvent être rapprochés des algorithmes de différences temporelles classiques, dans le sens où la mise à jour est locale, c'est-à-dire qu'elle se fait en fonction de l'erreur de prédiction faite sur la transition courante. Les algorithmes présentés au chapitre 4 peuvent quant à eux être rapprochés des méthodes de Monte Carlo, au vu de l'analyse de convergence proposée. Les mises à jour correspondantes sont donc globales, dans le sens où elle font usage des erreurs de prédiction sur l'ensemble de la trajectoire suivie. Comme pour les algorithmes de différences temporelles classiques le concept des traces d'éligibilité permet de jeter un pont entre ces deux extrêmes, nous l'adaptons au **chapitre 5** au cadre de travail proposé. L'approche est cependant différente de celle couramment considérée. En pratique, elle correspond à l'introduction d'une nouvelle famille de bruits colorés, dépendant d'un paramètre λ, dit facteur d'éligibilité. Les algorithmes des chapitres 3 et 4 seront montrés en être des cas particuliers. Nous analysons également les algorithmes résultants et montrons qu'ils tendent à minimiser une fonction de coût particulière. Nous ne pouvons pas nous prononcer sur le biais éventuel de l'estimateur correspondant. Cependant, empiriquement, nous montrons que cette nouvelle famille de bruits conduit à des estimations non biaisées de la fonction de valeur (excepté pour le cas particulier correspondant à l'algorithme du chapitre 3). De plus, toujours empiriquement, nous constaterons que faire varier la valeur de λ permet de réduire la variance de l'estimation.

L'apprentissage étant en ligne et apprentissage et contrôle étant imbriqués se pose le

problème de ce qui est appelé dilemme entre exploration et exploitation. L'agent doit à tout instant choisir entre une action exploitante (agir de façon optimale respectivement à sa connaissance imparfaite du monde) et plusieurs actions exploratrices (améliorer sa connaissance du monde). Pour traiter ce problème, nous pensons important de maintenir une information sur la confiance qu'a l'agent en sa représentation du monde. Dans le **chapitre 6**, nous montrons comment une incertitude sur les estimations de la fonction de valeur peut être calculée dans le contexte du cadre de travail proposé. Ceci résulte du fait que nous modélisons les paramètres par des variables aléatoires et que la fonction de valeur approchée est fonction de ces paramètres. Pour un état donné, la valeur estimée associée est donc également une variable aléatoire. Nous proposons une forme d'apprentissage actif utilisant cette information d'incertitude et permettant d'accélérer l'apprentissage dans certains cas précis. Nous introduisons également des adaptations d'approches classiques de la littérature visant à traiter le dilemme entre exploration et exploitation, la plupart ayant été introduites uniquement dans le cas tabulaire. Nous expérimentons ces différents schémas sur un problème de bandits à N bras. Ce test de référence consiste à trouver le plus rapidement et le plus sûrement possible celui des N bras qui apporte, en moyenne, le plus grand gain.

Toutes les approches que nous proposons peuvent être qualifiées de "critique pur". Seule une représentation de la fonction de valeur est maintenue et la politique d'action de l'agent en est déduite. Ce type d'approche présente certains inconvénients, comme une difficulté à garantir la qualité du contrôle ou encore des problèmes de calcul de maximum, ce que nous discutons au **chapitre 7**. Un autre type d'approche est dite "acteur pur", dont le principe est de ne maintenir qu'une représentation de la politique et de modifier les paramètres associés de façon à maximiser sa performance (ou autrement dit la fonction de valeur associée, qui n'a cependant pas de représentation propre), cette performance (ou son gradient) étant généralement estimée par une méthode de Monte Carlo. Les méthodes acteur-critique combinent ces deux approches. La politique (l'acteur) et la fonction de valeur (le critique) ont chacun leur représentation propre. Le critique estime la performance de l'acteur (qui contrôle le système) et cette estimation sert à modifier les paramètres de la politique. Basiquement, si une action a un effet plus bénéfique que prévu, son association avec l'état courant sera renforcée, dans le cas contraire non. Dans cette optique, nous présentons un algorithme acteur-critique dont l'acteur est basé sur une descente de gradient naturel (le gradient naturel étant un gradient pondéré par l'inverse de la matrice d'information de Fisher, ce que nous développerons) et le critique sur les différences temporelles de Kalman.

Le **chapitre 8** présente une application industrielle de l'apprentissage par renforcement. Plus précisément nous y traitons un problème de gestion de flux des gaz. Dans un complexe sidérurgique, certaines usines produisent du gaz, d'autres en consomment. Il est possible de brûler le gaz excédentaire (torches), d'en acheter sur un réseau extérieur s'il n'y en a pas assez, de router le gaz entre les différentes usines et de le stoker à certains endroits dans des accumulateurs. A cela il faut ajouter des contraintes sur les flux, les puissances et les capacités énergétiques des différents gaz. L'objectif est ici de maximiser le profit sur le long terme, sachant que les quantités de gaz produites et consommées évoluent au cours du temps. Nous proposons une modélisation du problème permettant de créer un simulateur sur lequel nous comparons nos algorithmes à certains autres pouvant être considérés comme proches de ceux employés actuellement dans l'industrie (basiquement une maximisation de la récompense immédiate) : l'approche proposée se montre supérieure.

Enfin, le **chapitre 9** présente un résumé étendu des travaux présentés et contributions

apportées, ainsi qu'un ensemble de perspectives qu'il serait intéressant de développer pour améliorer le cadre de travail introduit. Une de celles-ci est par exemple d'étendre les algorithmes introduits au cas de l'observabilité partielle, c'est-à-dire lorsque l'état du système que l'agent doit contrôlé n'est pas directement observable. Ceci devrait se faire de façon très naturelle dans le paradigme proposé.

Chapitre 2

Etat de l'art

Dans ce chapitre nous proposons un état de l'art sur l'apprentissage par renforcement. Bien sûr, ce dernier n'est pas exhaustif (le lecteur peut se référer à des ouvrages sur le sujet comme [98, 105, 11, 12, 90] pour plus de détails), mais orienté vers nos travaux. Il existe plusieurs approches pour traiter le paradigme de l'apprentissage par renforcement. Cependant, un pan important de la littérature est basé sur la programmation dynamique et c'est avec cette vision des choses que nous l'abordons. Nous présentons donc dans un premier temps la programmation dynamique, le formalisme associé (processus décisionnel de Markov pour la modélisation de l'environnement, fonction de valeur pour le cumul des récompenses, politique pour le contrôle, *et caetera*) et les algorithmes de résolution associés. Nous présentons ensuite les méthodes de base de l'apprentissage par renforcement, principalement les approches dites de différences temporelles, pour le cas où les espaces d'état et d'action sont finis et de petites dimensions, auquel cas une représentation tabulaire de la fonction de valeur est possible. Enfin nous passons en revue un certain nombre d'approches permettant de traiter les grands espaces d'état (que nous supposerons finis pour des raisons techniques, mais trop grands pour qu'une énumération des états soit envisageable). Remarquons que nous notons de façon générale les variables aléatoires en majuscules et leurs réalisations en minuscules, sauf éventuellement lorsque la variable aléatoire est précisée en indice de l'espérance, ce qui permet d'alléger les notations. Nous notons également pour un ensemble X fini $\mathcal{P}(X) = \{f \in [0,1]^X \mid \sum_{x \in X} f(x) = 1\}$ l'ensemble des densités de probabilités sur X.

2.1 Programmation dynamique

La programmation dynamique [8] peut être définie de façon très générale comme l'ensemble des techniques algorithmiques dont le principe est de déterminer la solution optimale d'un problème à partir d'une solution optimale d'un sous-problème. Dans le contexte où nous nous plaçons, ce sont toutes les méthodes permettant la résolution exacte (voir approchée) de l'équation de Bellman (que nous présentons par la suite), sans composante d'apprentissage.

2.1.1 Formalisme

Avant de traiter les algorithmes de résolution de la programmation dynamique, le formalisme doit d'abord être posé. Nous introduisons les notions de processus décisionnel de

Markov (modélisation de l'environnement à contrôler), de fonction de valeur (quantification de la qualité du contrôle), de politique (le contrôle lui même), ainsi que les équations de Bellman (caractérisation de la fonction de valeur d'une politique donnée et de la politique optimale) dont la résolution est l'une des composantes des algorithmes de programmation dynamique.

Processus décisionnel de Markov

Le formalisme associé à la programmation dynamique (mais aussi très largement à l'apprentissage par renforcement) est celui des processus décisionnels de Markov [8, 55, 90, 11] (MDP pour *Markov Decision Processes*). Ils permettent de modéliser l'environnement illustré sur la figure 1.1. Nous en donnons une définition.

Définition 2.1 (Processus décisionnel de Markov). *Un MDP est la donnée des éléments suivants :*

- S *l'espace des états (fini) ;*
- A *l'espace des actions (fini) ;*
- $P : s, a \in S \times A \rightarrow p(.|s, a) \in \mathcal{P}(S)$ *une famille de probabilités de transitions markoviennes ;*
- $R : s, a, s' \in S \times A \times S \rightarrow r \in \mathbb{R}$ *une fonction de récompense bornée ;*
- $\gamma \in [0, 1[$ *un facteur d'actualisation.*

Un processus décisionnel de Markov est donc composé d'un espace d'état, qui décrit l'ensemble des configurations possibles du système dynamique ; d'un espace d'action, qui décrit l'ensemble des actions qui peuvent être appliquées à ce système ; d'une famille de probabilités de transitions markoviennes, qui décrit la dynamique du système ; d'une fonction de récompense, qui est une information locale et immédiate de la qualité du contrôle ; d'un facteur d'actualisation, qui participe à la définition de la fonction de valeur et qui rend compte en quelque sorte de l'horizon sur lequel on souhaite maximiser le cumul de récompenses. Nous dirons de façon générale que le modèle est connu si les probabilités de transitions et la fonction de récompense le sont. Nous insistons sur le caractère markovien des probabilités de transitions : la probabilité de passer dans un nouvel état s' ne dépend que de l'état courant s et de l'action a choisie et en aucun cas du chemin parcouru pour atteindre l'état courant. Plus formellement, pour une trajectoire $s_1, \ldots, s_{i+1}$ au cours de laquelle les actions $a_1, \ldots, a_i$ ont été prises, nous avons :

$$p(s_{i+1}|s_i, a_i, s_{i-1}, a_{i-1}, \ldots, s_1, a_1) = p(s_{i+1}|s_i, a_i) \tag{2.1}$$

De plus, il est toujours possible de se ramener à un système de Markov[1], on ne perd donc pas en généralité.

La figure 2.1 illustre ce modèle. A chaque instant i (échelle des temps discrète), le système est dans l'état courant $s_i \in S$, une action $a_i \in A$ lui est appliquée, ce qui le mène aléatoirement selon $p(.|s_i, a_i)$ dans un nouvel état s_{i+1}. Cette transition génère une récompense $r(s_i, a_i, s_{i+1})$.

Il est à noter que plusieurs extensions du formalisme des MDP existent. Les processus décisionnels de Markov partiellement observables [126, 62, 76, 77] (POMDP pour *Partially Observable Markov Decision Processes*) sont une formalisation des MDP dans laquelle les

[1] Si le système n'est pas markovien, il suffit de remplacer les états par des historiques. Bien sûr, d'un point de vue pratique, cela peut poser problème.

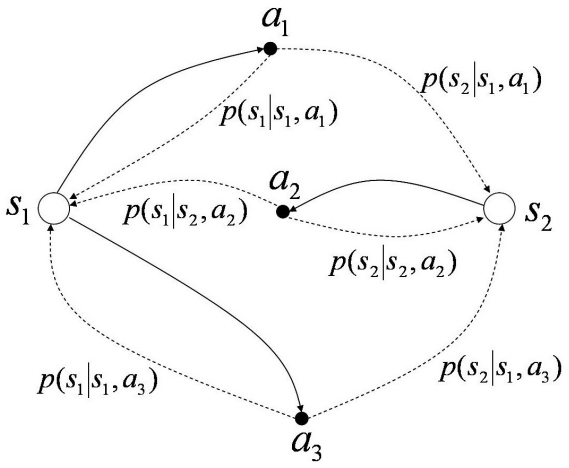


FIG. 2.1 – Processus décisionnel de Markov illustré.

états sont bien définis mais non directement observables. Cela peut modéliser une situation où le système est observé via des capteurs bruités, par exemple. Les décisions doivent être prises en fonction d'observations, une même observation pouvant correspondre à plusieurs états différents. Généralement, cela induit la perte du caractère markovien des transitions (considérées par rapport aux observations), ce qui complique de façon substantielle le problème. Les semi-processus décisionnels de Markov [90] (SMDP pour *Semi-MDP*) sont une extension des MDP au cas où le temps pris pour effectuer une action (le temps pouvant maintenant être continu) n'est plus constant. Ce formalisme est surtout utilisé dans le cadre des approches dites hiérarchiques (voir par exemple le cadre de travail des options, pour lequel des politiques définissent des méta-actions [108]). Les processus décisionnels de Markov factorisés (FMDP pour *Factored MDP*) sont un formalisme qui ne considère plus une représentation "plate" des espaces d'état et d'action, mais factorisée. Ils peuvent être vus comme une formulation particulière de réseaux bayésiens dynamiques [51] (DBN pour *Dynamic Bayesian Network*) contrôlés. Cette formulation est intéressante pour la résolution de problèmes d'optimisation multi-dimensionnels et ce n'est que récemment que des algorithmes d'apprentissage de la structure ont été proposés, voir par exemple [24]. Il existe encore d'autres extensions, comme les MDP décentralisés [10] (DEC-MDP pour *Decentralized MDP*) utilisés dans un contexte multi-agent. Nous n'avons cité que les plus connues et ces formalismes présentent tous un intérêt selon le type de situation à modéliser. Cependant, nous nous restreindrons dans la suite de ce manuscrit à la formulation la plus classique d'un MDP, telle que donnée définition 2.1.

Politique et fonction de valeur

Deux notions restent encore à introduire avant de pouvoir définir plus formellement l'objectif de la programmation dynamique et les moyens de l'atteindre. La première est la notion de politique, qui spécifie la façon dont est contrôlé le système. La seconde est la notion de fonction de valeur : elle quantifie la qualité d'une politique et c'est le gain que l'on cherche à maximiser.

Définition 2.2 (Politique). *De façon générale, une politique (ou stratégie) associe à un*

état donné une distribution de probabilités sur les actions :

$$\pi : s \in S \rightarrow \pi(.|s) \in \mathcal{P}(A) \tag{2.2}$$

Un cas particulier sont les politiques déterministes, qui à un état associent une action :

$$\pi : s \in S \rightarrow a = \pi(s) \in A \tag{2.3}$$

La politique définit donc la façon d'interagir avec l'environnement. Il est à noter que les politiques déterministes sont une classe importante dans la mesure où il est possible de montrer qu'il existe toujours une politique déterministe parmi les politiques optimales [98] (la notion d'optimalité étant rendue plus claire par la suite). Dans la suite de ce manuscrit nous considérerons généralement des politiques déterministes, sauf mention explicite du contraire.

La politique étant définie, reste à choisir un critère pour mesurer sa qualité. Nous avons déjà dit que l'objectif est de façon générale la maximisation du cumul des récompenses sur le long terme, cependant, plusieurs types de cumuls peuvent être considérés. Par exemple, on peut souhaiter maximiser la somme des récompense sur un horizon T fini et le critère est donc $\sum_{i=0}^{T} r_i$. Un autre critère possible est de maximiser la récompense moyenne sur un horizon infini, ce qui se traduit par $\lim_{T\rightarrow\infty} \frac{1}{T} \sum_{i=0}^{T} r_i$. Le critère que nous choisissons et considérons exclusivement dans la suite de ce manuscrit est le critère dit γ-pondéré, qui est une somme pondérée des récompenses sur un horizon infini : $\sum_{i=0}^{\infty} \gamma^i r_i$. En effet, hors le fait qu'il soit très classique dans la littérature et que d'un point de vue mathématique il assure l'existence de la série, la récompense étant bornée, il présente également un intérêt pratique ; il maximise la récompense sur une horizon infini, en accordant toutefois plus d'importance aux récompenses les plus immédiates, ce qui nous semble légitime, tout particulièrement dans un contexte industriel. La fonction de valeur associe simplement à chaque état le cumul espéré de récompenses et nous la définissons donc pour le critère γ-pondéré.

Définition 2.3 (Fonction de valeur). *La fonction de valeur associe à un état $s \in S$, pour une politique (éventuellement stochastique) π donnée, le cumul pondéré des récompenses espéré en partant de l'état s et en suivant la politique π jusqu'à la fin.*

$$V^{\pi} : s \in S \rightarrow V^{\pi}(s) = E\left[\sum_{i=0}^{\infty} \gamma^i R(S_i, A_i, S_{i+1}) | S_0 = s, \pi\right] \in \mathbb{R} \tag{2.4}$$

Dans l'équation (2.4), le caractère aléatoire est parfaitement quantifié par les probabilités de transition. Elle peut d'ailleurs être réécrite pour une politique déterministe de la façon suivante :

$$V^{\pi}(s) = \sum_{i=0}^{\infty} \gamma^i \sum_{s_{1:i+1} \in S^{i+1}} R(s_i, \pi(s_i), s_{i+1}) \prod_{k=1}^{i+1} p(s_k | s_{k-1}, \pi(s_{k-1})) \text{ où } s_0 = s \tag{2.5}$$

Pour une politique stochastique, le résultat est très proche, bien qu'un peu plus long à écrire (il suffit de remplacer les actions déterministes par une espérance sur les actions, la loi de probabilité associée étant la politique conditionnée à l'état *ad hoc*).

Equations de Bellman

L'objectif de la programmation dynamique est de découvrir l'une des politiques dont la fonction de valeur est maximale pour l'ensemble des états. En remarquant que $(S \to \mathbb{R}) \subset \mathbb{R}^{|S|}$ où $|S|$ est le cardinal de l'espace d'état, ou autrement dit que la fonction de valeur peut être vue comme un vecteur avec autant de composantes qu'il y a d'états, il est possible de munir les fonctions de valeur d'une relation d'ordre partielle :

$$V_1 \leq V_2 \Leftrightarrow \forall s \in S, \quad V_1(s) \leq V_2(s) \tag{2.6}$$

Un ordre partiel peut être défini sur les politiques, *via* les fonctions de valeur associées :

$$\pi_1 \leq \pi_2 \Leftrightarrow V^{\pi_1} \leq V^{\pi_2} \tag{2.7}$$

L'objectif est donc de déterminer la politique optimale π^* définie par :

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, V^{\pi} \tag{2.8}$$

Une première question naturelle est de savoir si ce maximum existe bien, l'ordre étant partiel. C'est effectivement le cas, voir par exemple [98] pour une preuve.

Un autre problème est de savoir évaluer une politique donnée, c'est-à-dire déterminer sa fonction de valeur. Pour cela, l'expression (2.5) est difficilement utilisable. Cependant, il est possible de définir la fonction de valeur de manière récursive :

$$\begin{aligned} V^{\pi}(s) &= E[\sum_{i=0}^{\infty} \gamma^i R(S_i, \pi(S_i), S_{i+1} | S_0 = s, \pi] \\ &= E_{s'|s,\pi(s)} \left[R(s, \pi(s), s') + \gamma V^{\pi}(s') \right] \\ &= \sum_{s' \in S} p(s'|s, \pi(s)) \left(R(s, \pi(s), s') + \gamma V^{\pi}(s') \right) \end{aligned} \tag{2.9}$$

La valeur d'un état est donc la moyenne (selon les probabilités de transitions) de la somme de la récompense obtenue suite à l'application de l'action spécifiée par la politique que l'on souhaite évaluer et de la valeur de l'état vers lequel transite le système, pondérée par le facteur d'actualisation. Cette équation, appelée *équation d'évaluation de Bellman*, définit un système linéaire de $|S|$ équations à $|S|$ inconnues permettant de déterminer la fonction de valeur d'une politique donnée et donc de quantifier sa qualité.

Une autre équation, non-linéaire celle-ci, permet de déterminer directement la fonction de valeur optimale $V^* = V^{\pi^*}$. Soit un état s donné, supposons la fonction de valeur optimale connue en les états s' vers lesquels peut transiter le système. La fonction de valeur optimale en l'état s maximise (à la pondération par les probabilités de transitions près) la récompense immédiate plus la valeur optimale en l'état s' vers lequel transite le système, pondérée par le facteur d'actualisation. C'est l'*équation d'optimalité de Bellman* :

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} p(s'|s, a) \left(R(s, a, s') + \gamma V^*(s') \right) \tag{2.10}$$

Ceci définit un système non-linéaire à $|S|$ équations et $|S|$ inconnues. Si la fonction de valeur optimale est connue, il est aisé d'en déduire la politique optimale, qui est gloutonne par rapport à cette dernière, c'est-à-dire qui vérifie :

$$\pi^*(s) = \underset{a \in A}{\operatorname{argmax}} \sum_{s' \in S} p(s'|s, a) \left(R(s, a, s') + \gamma V^*(s') \right) \tag{2.11}$$

Nous résumons ces équations importantes pour la suite, en définissant au passage les opérateurs T^π et T^* correspondants :

Théorème 2.1 (Equations de Bellman). *L'équation d'évaluation de Bellman permet de déterminer la fonction de valeur d'une politique π donnée :*

$$\begin{aligned} \forall s \in S, \quad V^\pi(s) &= \sum_{s' \in S} p(s'|s, \pi(s)) \left(R(s, \pi(s), s') + \gamma V^\pi(s') \right) \\ &\Leftrightarrow V^\pi = T^\pi V^\pi \end{aligned} \tag{2.12}$$

L'équation d'optimalité de Bellman permet de déterminer la fonction de valeur optimale V^* :

$$\begin{aligned} \forall s \in S, \quad V^*(s) &= \max_{a \in A} \sum_{s' \in S} p(s'|s, a) \left(R(s, a, s') + \gamma V^*(s') \right) \\ &\Leftrightarrow V^* = T^* V^* \end{aligned} \tag{2.13}$$

De plus, les opérateurs de Bellman T^π et T^* sont des contractions en norme infinie.

La norme infinie est ici le maximum des éléments ($\|V\|_\infty = \max_{s \in S} V(s)$) et un opérateur T est une contraction s'il existe un réel positif $k < 1$ tel que pour deux fonctions de valeur quelconques on ait toujours $\|TV_1 - TV_2\|_\infty \leq k\|V_1 - V_2\|_\infty$. Cette propriété de contraction des opérateurs de Bellman, prouvée dans de nombreux ouvrages comme [98] par exemple, est importante. D'une part elle garantit l'existence et l'unicité des solutions (théorème 2.2 du point fixe de Banach) et d'autre part elle fournit une méthode de résolution directe de l'équation d'optimalité.

Théorème 2.2 (Théorème du point fixe de Banach). *Soit $\mathcal{U}$ un espace de Banach (espace vectoriel normé complet) et T une contraction sur $\mathcal{U}$. Alors :*

1. *$\exists! u^* \in \mathcal{U}$ tel que $Tu^* = u^*$;*
2. *$\forall u_0 \in \mathcal{U}$, la suite $(u_n)_{n \in \mathbb{N}}$ définie par $u_{n+1} = Tu_n$ converge vers u^*.*

Notons que si la fonction de valeur optimale est bien unique, plusieurs politiques différentes peuvent lui correspondre.

2.1.2 Résolution

Plusieurs algorithmes existent pour déterminer la politique optimale (via la fonction de valeur associée). Par exemple un algorithme basé sur la programmation linéaire permet de trouver la fonction de valeur optimale, nous ne présenterons pas ici (là encore, voir [98]). Nous nous focalisons sur deux méthodes dont les idées sont utilisées de façon intensive en apprentissage par renforcement, à savoir les algorithmes d'itération de la politique et d'itération de la valeur.

Itération de la politique

La première approche que nous présentons est l'itération de la politique. L'algorithme est initialisé avec une politique quelconque. Son principe est d'évaluer la fonction de valeur de la politique courante (ce que nous appelons évaluer la politique), puis d'améliorer cette

politique en considérant la politique gloutonne par rapport à la fonction de valeur précédemment calculée. Autrement dit, dans un état donné, l'action choisie est celle qui mène (en moyenne) au plus grand cumul de récompenses. Il est important de noter que ce n'est pas nécessairement l'action choisie par la politique, sauf si cette dernière est optimale. Plus formellement, si à l'itération i la politique π_i est évaluée, la politique améliorée π_{i+1} est définie par :

$$\forall s \in S, \quad \pi_{i+1}(s) = \underset{a \in A}{\operatorname{argmax}} \sum_{s' \in S} p(s'|s,a) \left(R(s,a,s') + \gamma V^{\pi_i}(s') \right) \tag{2.14}$$

Cette approche est résumée dans l'algorithme 2.1.

Algorithme 2.1 : Itération de la politique

Initialisation;
Politique π_0;
$i \leftarrow 0$;
tant que $\pi_{i+1} \neq \pi_i$ **faire**
  Evaluation de la politique;
  Résoudre $V_i = T^{\pi_i} V_i$;
  Amélioration de la politique;
  pour tous les $s \in S$ **faire**
    $\pi_{i+1}(s) = \operatorname{argmax}_{a \in A} \sum_{s' \in S} p(s'|s,a)(R(s,a,s') + \gamma V_i(s'))$;
  $i \leftarrow i + 1$;

Avec ce schéma d'amélioration, il peut être montré qu'il y a amélioration de la politique, c'est-à-dire que $\pi_{i+1} \geq \pi_i$. Si l'on a égalité, alors l'équation d'optimalité de Bellman est vérifiée et l'algorithme a convergé. De plus, le nombre de politiques étant fini, cet algorithme converge en un nombre fini d'itérations (ce nombre d'itérations étant empiriquement beaucoup plus faible que la cardinal de l'espace des politiques). Il est à noter que l'évaluation de la politique revient à résoudre un système linéaire, la complexité de cette évaluation est donc en $O(|S|^3)$. L'existence d'une solution à ce système est assurée par le théorème du point fixe de Banach (V_i est le point fixe de la contraction T^{π_i}). Le calcul de la politique gloutonne associée est en $O(|S|^2|A|)$. La complexité de cet algorithme est donc en $O(|S|^2|A| + |S|^3)$ par itération. Les idées de cet algorithme, c'est-à-dire évaluation d'une politique suivie d'une amélioration, sont très utilisées dans les algorithmes d'apprentissage par renforcement, comme nous le verrons plus tard.

Itération de la valeur

La seconde approche que nous présentons, basée sur l'équation d'optimalité de Bellman, vise à déterminer directement la fonction de valeur optimale V^*. Etant donné que l'opérateur de Bellman T^* est une contraction et que V^* en est l'unique point fixe, le théorème 2.2 de Banach assure qu'une récursion du type $V_{i+1} = T^* V_i$ convergera vers la fonction de valeur optimale, quelle que soit l'initialisation V_0. L'approche est résumée dans l'algorithme 2.2.

Chaque itération consiste en l'application de l'opérateur d'optimalité de Bellman, la complexité algorithmique associée est en $O(|S|^2|A|)$. Cet algorithme ne converge pas en

Algorithme 2.2 : Itération de la valeur

Initialisation;
Fonction de valeur V_0;
$i \leftarrow 0$;
tant que $\|V_{i+1} - V_i\|_\infty \geq \epsilon$ **faire**
    Itérer la valeur;
    $V_{i+1} = T^* V_i$;
    $i \leftarrow i + 1$;

un nombre fini d'itérations, c'est pourquoi un critère d'arrêt est introduit. Celui-ci permet de borner l'erreur. En effet, il est possible de montrer que si le critère d'arrêt est vérifié à l'itération i, alors $\|V_i - V^*\| \leq \frac{2\gamma}{1-\gamma}\epsilon$. Cependant, cette borne ne garantit pas la qualité de la politique gloutonne associée.

2.2 Apprentissage par renforcement

Dans cette section nous nous intéressons à l'apprentissage par renforcement et plus particulièrement aux méthodes dites de différences temporelles. Un certain nombre de contraintes sont relâchées par rapport à ce que nous avons vu dans la section 2.1 pour la programmation dynamique, comme la connaissance du modèle *a priori*. Cependant, nous supposerons toujours les espaces d'état et d'action suffisamment petits (*i.e.* de cardinaux suffisamment faibles) pour qu'une représentation tabulaire soit possible, autrement dit pour que la fonction de valeur et la politique puissent être représentées par des vecteurs (d'un point de vue machine, formellement c'est toujours le cas sous l'hypothèse d'espaces finis). Le cas où l'espace d'état est trop grand pour une telle représentation sera traité dans la section 2.3.

2.2.1 Position de l'apprentissage par renforcement par rapport à la programmation dynamique

Pour les méthodes d'apprentissage par renforcement que nous considérons ici, le formalisme est le même que pour la programmation dynamique. L'environnement à contrôler est toujours modélisé par un processus décisionnel markovien et l'objectif est encore de déterminer la politique optimale, c'est-à-dire celle dont la fonction de valeur est maximale pour chaque état du système. Cependant, par rapport à la programmation dynamique, un certain nombre de contraintes sont levées. Premièrement, le modèle (c'est-à-dire, nous le rappelons, les probabilités de transitions et la fonction de récompense) n'est plus supposé connu *a priori*. Nous supposerons toutefois les espaces d'état et d'action connus. Il y a donc une composante d'apprentissage qui entre en compte, dans la mesure où l'on va chercher à apprendre le contrôle optimal à partir d'interactions avec l'environnement ; cela devient plus qu'un problème d'optimisation. De plus, une caractéristique souhaitée de cet apprentissage est qu'il se fasse en ligne, c'est-à-dire que la fonction de valeur (et/ou la politique) soit mise à jour après chaque interaction avec l'environnement et ce sans utiliser tout l'historique des interactions. Il est également probable que cette mise à jour soit locale et ne concerne pas tous les états, elle est donc asynchrone. L'apprentissage par renforcement (tel

que nous le traitons ici) peut donc être vu comme une version en ligne, asynchrone et sans connaissance *a priori* du modèle de la programmation dynamique.

Quand le modèle n'est pas connu, une première approche pourrait être de l'apprendre à partir d'interactions avec le système ($|S|^2|A|$ paramètres pour la fonction de récompense et autant pour le modèle de transition) pour lui appliquer les algorithmes classiques de programmation dynamique. Le problème de cette approche est qu'elle n'est pas en ligne et surtout qu'elle passe mal à l'échelle. Une autre approche pourrait être d'estimer la valeur des différents états à l'aide d'une méthode de Monte Carlo[2]. Plus précisément, le principe serait de simuler un grand nombre de trajectoires partant d'un état $s \in S$ donné et de moyenner les cumuls de récompenses observées pour chaque trajectoire. On pourrait alors remplacer l'évaluation de la politique (faite par résolution du système linéaire correspondant si le modèle est connu) par ce type d'approche dans l'algorithme d'itération de la politique (la phase d'amélioration de la politique nécessitant cependant également de recourir à Monte Carlo, le modèle n'étant pas connu), ce qui pourrait permettre d'aboutir à la politique optimale. Là encore le caractère en ligne est perdu, un simulateur serait nécessaire et de plus les estimateurs de Monte Carlo présentent généralement une forte variance, sans compter qu'un tel apprentissage serait lent (simuler un grand nombre de trajectoires pour chaque état du système). Les méthodes de différences temporelles que nous présentons maintenant conservent quant à elles les caractéristiques désirées : apprentissage en ligne sans connaissance du modèle.

2.2.2 Algorithmes de différences temporelles

Les algorithmes que nous présentons ici visent à apprendre la fonction de valeur (ou la Q-fonction, que nous présentons par la suite) en ligne à partir d'interactions avec l'environnement. Ils reposent sur une comparaison entre la récompense que l'on s'attend à observer, cette prédiction étant basée sur l'estimation courante de la fonction de valeur et la récompense effectivement observée. Les mises à jour associées sont du type Widrow-Hoff[3] et font partie du domaine de l'estimation stochastique (qui n'est pas propre à l'apprentissage par renforcement). Nous présentons d'abord les algorithmes TD, SARSA et Q-learning, ainsi que leurs extensions aux traces d'éligibilité (qui permettent une mise à jour plus globale de la fonction de valeur, c'est-à-dire pas seulement de l'état courant mais de tous les états visités jusqu'à celui-ci). Dans ce contexte, nous introduisons la notion d'itération de la politique généralisée, ainsi que les architectures acteur-critique qui en sont un cas particulier important. Enfin, nous présentons les problèmes induits par l'imbrication de l'apprentissage et du contrôle, notamment le dilemme entre exploration et exploitation et la non-stationnarité de la fonction de valeur cible. Les algorithmes et résultats donnés se retrouvent dans des ouvrages qui font référence [98, 105, 12].

TD, SARSA et Q-learning

Nous nous intéressons dans un premier temps à l'algorithme TD (*Temporal Difference*), qui vise à apprendre la fonction de valeur d'une politique π donnée. Il s'agit donc d'éva-

[2]De façon très générale, une méthode de Monte Carlo vise à calculer une valeur numérique en utilisant des procédés aléatoires.

[3]De façon très générale, une équation de Widrow-Hoff consiste à mettre à jour un modèle, la correction étant proportionnelle à l'erreur faite entre une observation et la prédiction (basée sur le modèle courant) de cette observation.

luer la qualité d'une politique imposée, phase nécessaire dans un contexte d'itération de la politique. Autrement dit, c'est un problème d'estimation pour l'instant et non de contrôle. Rappelons que i désigne de façon générale l'indice temporel. Comme le problème est l'évaluation de la fonction de valeur d'une politique déterministe, seuls les états et les récompenses ont besoin d'être observés, pas les actions qui sont choisies par la politique. Etant donnée une transition (s_i, r_i, s_{i+1}) générée par cette dernière, il est possible d'écrire de façon informelle l'équation de Bellman (ou de façon formelle en supposant les transitions déterministes) :

$$V^\pi(s_i) = r_i + \gamma V^\pi(s_{i+1}) \tag{2.15}$$
$$\Leftrightarrow r_i = V^\pi(s_i) - \gamma V^\pi(s_{i+1}) \tag{2.16}$$

Dans l'équation (2.15), le membre de droite est le plus informatif, dans la mesure où il contient la récompense (la fonction de valeur ne peut pas être directement observée). On peut aussi remarquer que si les transitions sont stochastiques, bien que cette équation ne soit pas correcte (il manque l'espérance sur les transitions), le membre de droite est un estimateur sans biais du membre de gauche. L'algorithme TD consiste à mettre à jour la fonction de valeur proportionnellement à l'erreur de prédiction effectuée. Il y a deux façons de voir cette erreur, qui mènent ici au même résultat. La première, la plus classique, consiste à dire que l'on prédit la valeur de l'état courant avec son estimation et que l'on considère l'observation de cette valeur comme étant la somme de la récompense et de la valeur de l'état vers lequel transite le système. Cette valeur étant elle-même estimée, on devrait parler plutôt de "pseudo-observation", ce qui est connu sous le terme de *bootstrapping* [105, Ch. 6.2]. Le deuxième point de vue, moins usuel, consiste à dire que l'on cherche à prédire la récompense à partir de la différence entre les valeurs des deux états consécutifs de la transition. Ces deux points de vue mènent à la même mise à jour dans le cas qui nous intéresse, mais nous verrons lorsque nous aborderons les algorithmes directs et résiduels dans la section 2.3 les différences qu'ils impliquent. Plus formellement, en notant $\hat{V}$ l'estimation de la fonction de valeur V, l'algorithme TD consiste à mettre à jour la valeur de l'état courant s_i de la façon suivante :

$$\hat{V}^\pi(s_i) \leftarrow \hat{V}^\pi(s_i) + \alpha_i \delta_i \text{ où } \delta_i = r_i + \gamma \hat{V}^\pi(s_{i+1}) - \hat{V}^\pi(s_i) \tag{2.17}$$

Dans cette dernière expression, δ_i est appelée erreur de différence temporelle (*temporal difference error* en anglais) et α_i est un taux d'apprentissage. Si ce dernier vérifie les conditions classiques suivantes :

$$\sum_{i=0}^{\infty} \alpha_i = \infty \text{ et } \sum_{i=0}^{\infty} \alpha_i^2 < \infty \tag{2.18}$$

alors on peut montrer que $\hat{V}^\pi$ converge vers la fonction de valeur V^π associée à la politique π. Cette notion d'erreur de différence temporelle est primordiale en apprentissage par renforcement et la définition suivante peut en être donnée (nous n'en donnons pas d'expression formelle, dans la mesure où comme nous le verrons quelques variantes existent).

Définition 2.4 (Erreur de différence temporelle). *L'erreur de différence temporelle, pour une transition donnée, est la différence entre la récompense observée et la prédiction de cette récompense, déterminée en utilisant les estimations de la valeur des deux états consécutifs de la transition.*

Algorithme 2.3 : TD

Initialisation;
Initialisation de la fonction de valeur $\hat{V}$;

pour $i = 1, 2, \ldots$ **faire**

Observer la transition (s_i, r_i, s_{i+1});

Mettre à jour la fonction de valeur ;
$\delta_i = r_i + \gamma \hat{V}^\pi(s_{i+1}) - \hat{V}^\pi(s_i)$;
$\hat{V}^\pi(s_i) \leftarrow \hat{V}^\pi(s_i) + \alpha_i \delta_i$;

L'algorithme TD permet donc d'évaluer la fonction de valeur d'une politique donnée. Cependant, le problème posé est celui de la détermination de la politique optimale. Si l'on envisage d'utiliser TD comme algorithme d'évaluation dans un schéma du type itération de la politique, le problème de déterminer la politique gloutonne se pose. En effet, cela nécessite de connaître le modèle de transition, ce dont on cherche justement à s'affranchir. C'est dans cette optique qu'est introduite la Q-fonction, ou fonction de qualité[4] :

Définition 2.5 (Fonction de qualité). *La fonction de qualité associe à un couple état-action $(s, a) \in S \times A$, pour une politique π donnée, le cumul pondéré des récompenses espéré en partant de l'état s et en y choisissant l'action a, puis en suivant la politique π jusqu'à la fin.*

$$Q^\pi : S \times A \to \mathbb{R} \tag{2.19}$$
$$(s, a) \to Q^\pi(s, a) = E\left[\sum_{i=0}^{\infty} \gamma^i R(S_i, \pi(S_i), S_{i+1} | S_0 = s, A_0 = a, \pi\right]$$

Ainsi, par rapport à la fonction de valeur, la Q-fonction laisse un degré de liberté supplémentaire sur le choix de la première action. L'intérêt principal de cette fonction est qu'il est maintenant possible de choisir une action gloutonne simplement en maximisant la fonction de qualité sur l'ensemble des actions pour un état fixé, ce qui ne nécessite plus de modèle de transition :

$$\forall s \in S, \quad \pi_{\text{glouton}}(s) = \underset{a \in A}{\operatorname{argmax}}\, Q^\pi(s, a) \tag{2.20}$$

De la même façon que les équations de Bellman ont été définies pour la fonction de valeur, elles peuvent l'être pour la fonction de qualité :

Théorème 2.3 (Equations de Bellman). *L'équation d'évaluation de Bellman permet de déterminer la fonction de qualité d'une politique π donnée :*

$$\forall (s, a) \in S \times A, \quad Q^\pi(s, a) = \sum_{s' \in S} p(s'|s, a) \left(R(s, a, s') + \gamma Q^\pi(s', \pi(s'))\right)$$
$$\Leftrightarrow Q^\pi = T^\pi Q^\pi \tag{2.21}$$

[4]Historiquement, le terme Q-fonction vient de la notation utilisée [121]. Le terme fonction de qualité est plus rarement employé dans la littérature. Cependant, nous le pensons pertinent, dans la mesure où la Q-fonction permet de mesurer, pour un état donné, la qualité d'une action par rapport aux autres. Nous utiliserons donc les deux termes indifféremment. Cette Q-fonction peut également être appelée fonction valeur-action.

L'équation d'optimalité de Bellman permet de déterminer la fonction de qualité optimale Q^* *:*

$$\forall (s,a) \in S \times A, \quad Q^*(s,a) = \sum_{s' \in S} p(s'|s,a) \left(R(s,a,s') + \gamma \max_{b \in A} Q^*(s',b) \right)$$
$$\Leftrightarrow Q^* = T^* Q^* \tag{2.22}$$

Les opérateurs de Bellman T^π *et* T^* *sont des contractions en norme infinie.*

Les mêmes notations sont utilisées pour les opérateurs de Bellman pour la fonction de valeur (théorème 2.1) et de qualité (théorème 2.3), la distinction se fait aisément en fonction du contexte. Notons qu'il existe un lien assez évident entre la fonction de qualité et la fonction de valeur, à savoir que pour un état donné, si l'action est choisie en accord avec la politique, la qualité est égale à la valeur :

$$\forall s \in S, \quad V^\pi(s) = Q^\pi(s, \pi(s)) \tag{2.23}$$
$$\forall s \in S, \quad V^*(s) = \max_{a \in A} Q^*(s,a) \tag{2.24}$$

La notion de Q-fonction ayant été présentée, nous nous intéressons maintenant aux approches basées sur cette dernière. L'algorithme TD a un intérêt historique et pédagogique, mais comme nous l'avons vu il est peu pratique.

L'algorithme SARSA a pour objectif l'apprentissage de la Q-fonction d'une politique donnée[5] à partir de transitions du type $(s_i, a_i, r_i, s_{i+1}, a_{i+1})$, transitions qui donnent son nom à l'algorithme. A politique fixée, un MDP définit une chaîne de Markov valuée[6] sur l'espace d'état, mais également une chaîne de Markov valuée sur l'espace joint état-action. Dans le premier cas, l'espace d'état est le même, la récompense est $R^\pi(s,s') = R(s, \pi(s), s')$ et le modèle de transition $\Pr(s'|s) = p(s'|s, \pi(s))$. Dans le second cas, l'espace d'état devient $S \times A$, la récompense est toujours R et le modèle de transition devient $\Pr(s', a'|s, a) = p(s'|s,a)\pi(a'|s)$. La mise à jour de SARSA est donc très similaire à celle de TD :

$$\hat{Q}^\pi(s_i, a_i) \leftarrow \hat{Q}^\pi(s_i, a_i) + \alpha_i \delta_i \text{ où } \delta_i = r_i + \gamma \hat{Q}^\pi(s_{i+1}, a_{i+1}) - \hat{Q}^\pi(s_i, a_i) \tag{2.25}$$

Si le taux d'apprentissage est choisi vérifiant (2.18), SARSA converge. Nous le résumons dans l'algorithme 2.4. Celui-ci peut servir pour la phase d'évaluation de l'algorithme d'itération de la politique, la phase d'amélioration étant alors faite en accord avec l'équation (2.20). Cependant, il n'est pas forcément nécessaire d'avoir parfaitement évalué une politique pour l'améliorer (ce qui prendrait de toute façon ici un nombre infini d'interactions entre l'agent et l'environnement). Nous discutons ce point dans la section 2.2.3.

Le dernier algorithme que nous passons en revue ici est le Q-learning. Il a pour but de déterminer directement la fonction de qualité optimale, la politique suivie étant (plus ou moins) arbitraire. Sa mise à jour est basée sur l'équation d'optimalité de Bellman. Pour une transition (s_i, a_i, r_i, s_{i+1}), elle prend la forme suivante :

$$\hat{Q}^*(s_i, a_i) \leftarrow \hat{Q}^*(s_i, a_i) + \alpha_i \delta_i \text{ où } \delta_i = r_i + \gamma \max_{a \in A} \hat{Q}^*(s_{i+1}, a) - \hat{Q}^*(s_i, a_i) \tag{2.26}$$

[5] Dans la littérature, l'algorithme SARSA est souvent associé à une politique ϵ-gloutonne et est donc souvent compris dans un sens d'itération de la politique généralisée (nous présentons ces notions par la suite). Cependant, dans ce manuscrit, lorsque nous référerons à SARSA ce sera généralement dans le sens d'évaluation de la Q-fonction d'une politique fixée, sans composante de contrôle.

[6] Une chaîne de Markov valuée est un MDP pour lequel il n'y a qu'une action.

Algorithme 2.4 : SARSA

Initialisation;
Initialisation de la fonction de qualité $\hat{Q}^\pi$;

pour $i = 1, 2, \ldots$ **faire**

Observer la transition $(s_i, a_i, r_i, s_{i+1}, a_{i+1})$;

Mettre à jour la Q-fonction ;
$\delta_i = r_i + \gamma \hat{Q}^\pi(s_{i+1}, a_{i+1}) - \hat{Q}^\pi(s_i, a_i)$;
$\hat{Q}^\pi(s_i, a_i) \leftarrow \hat{Q}^\pi(s_i, a_i) + \alpha_i \delta_i$;

Par rapport à SARSA, cela revient à choisir l'action gloutonne (pour la mise à jour) en l'état vers lequel transite le système, plutôt que l'action effectivement choisie par l'agent. Si le taux d'apprentissage est choisi vérifiant (2.18) et que chaque paire état-action est visitée un nombre infini de fois, alors Q-learning converge vers la Q-fonction optimale. Nous le résumons dans l'algorithme 2.5.

Algorithme 2.5 : Q-learning

Initialisation;
Initialisation de la fonction de qualité $\hat{Q}^*$;

pour $i = 1, 2, \ldots$ **faire**

Observer la transition (s_i, a_i, r_i, s_{i+1});

Mettre à jour la Q-fonction ;
$\delta_i = r_i + \gamma \max_{a \in A} \hat{Q}^*(s_{i+1}, a) - \hat{Q}^*(s_i, a_i)$;
$\hat{Q}^*(s_i, a_i) \leftarrow \hat{Q}^*(s_i, a_i) + \alpha_i \delta_i$;

L'algorithme Q-learning est dit *off-policy*[7], la politique évaluée (via la fonction de qualité) étant la politique optimale alors que la politique servant à générer les transitions ne l'est *a priori* pas. De façon générale, un algorithme sera dit *on-policy* si la politique évaluée est la politique comportementale (qui sert à générer les transitions) et *off-policy* si elles sont différentes. L'intérêt du caractère *off-policy* est assez évident dans le cas du Q-learning, l'algorithme permettant d'apprendre directement la politique optimale. De façon plus générale ce type d'algorithme peut s'avérer utile, par exemple pour réutiliser des trajectoires déjà parcourues et générées avec une certaine politique pour en apprendre une nouvelle, même si le schéma d'apprentissage est du type itération de la politique, ou encore si l'on ne souhaite pas que l'agent contrôle le système (dans le milieu industriel par exemple on préférera généralement se fier à des contrôleurs éprouvés) sans pour autant lui interdire de proposer une politique améliorée.

Les algorithmes TD, SARSA (que nous dirons être du type itération de la politique, car ils se rapportent à l'équation d'évaluation de Bellman) et Q-learning (que nous dirons être du type itération de la valeur, car il se rapporte à l'équation d'optimalité de Bellman) font des mises à jour locales de la fonction de valeur. Le concept de traces d'éligibilité que nous introduisons à présent permet de leur donner un caractère plus global, en propageant

[7] Nous utilisons la dénomination anglaise, n'ayant pas trouvé de traduction satisfaisante.

l'erreur de différence temporelle à un instant i le long de la trajectoire qui vient d'être suivie par l'agent.

Extension aux traces d'éligibilité

Nous avons vu que l'algorithme TD met à jour la valeur d'un état donné en fonction de l'erreur de différence temporelle δ_i. Cette erreur est définie avec une prédiction à un pas de temps de la fonction de valeur. Supposons qu'à partir de l'état s_i une trajectoire de longueur k soit observée, il est possible de définir des erreurs de différences temporelles en fonction de prédictions à plus grands horizons :

$$\begin{cases} \delta_i^{(1)} = r_i + \gamma V(s_{i+1}) - V(s_i) \\ \delta_i^{(2)} = r_i + \gamma r_{i+1} + \gamma^2 V(s_{i+2}) - V(s_i) \\ \vdots \\ \delta_i^{(k)} = r_i + \gamma r_{i+1} + \cdots + \gamma^{k-1} r_{i+k-1} + \gamma^k V(s_{i+k}) - V(s_i) \end{cases} \tag{2.27}$$

Il est donc possible de définir un algorithme TD à k pas, où l'erreur de différence temporelle d'ordre k est utilisée :

$$\hat{V}(s_i) \leftarrow \hat{V}(s_i) + \alpha_i \delta_i^{(k)} \tag{2.28}$$

Plutôt que de considérer un ordre k particulier, il est possible de moyenner les différentes erreurs de différences temporelles possibles. Pour l'approche des traces d'éligibilité [105], cette moyenne se fait à l'aide d'une pondération géométrique d'un facteur $\lambda \in [0, 1]$ dit d'éligibilité :

$$\delta_i^{\lambda} = (1 - \lambda) \sum_{k=1}^{\infty} \lambda^{k-1} \delta_i^{(k)} \tag{2.29}$$

Dans cette équation, le facteur $1 - \lambda$ normalise la somme des poids. Ceci permet de définir l'algorithme TD(λ) sous sa forme "vue avant" (*forward view*) :

$$\hat{V}(s_i) \leftarrow \hat{V}(s_i) + \alpha_i \delta_i^{\lambda} \tag{2.30}$$

Si le facteur d'éligibilité est nul, on se ramène au TD classique, s'il vaut 1 (et que le MDP possède un état terminal) il est possible de montrer que la méthode de Monte Carlo dont nous avons brièvement parlé auparavant est obtenue : l'estimation de la valeur de l'état courant est mise à jour avec un cumul pondéré de récompenses obtenu en simulant une trajectoire complète et sans utiliser d'estimation de la valeur en un autre état. Le concept des traces d'éligibilité jette donc un pont entre ces deux approches.

Cependant, cet algorithme n'est pas pratique. Il nécessite pour mettre à jour un état donné d'observer la trajectoire qui y commence jusqu'à la fin. Heureusement, il est possible de montrer que cette "vue avant'" est équivalente à une "vue arrière" (*backward view*) beaucoup plus pratique [105]. Son principe est de maintenir un vecteur $\Lambda \in \mathbb{R}^{|S|}$ appelé trace d'éligibilité (initialisé à 0 et pour lequel $\Lambda(s)$ désigne la composante relative à l'état s). A un instant i, la trace de chacun des états est réduite d'un facteur $\gamma\lambda$ et la trace de l'état courant s_i est incrémentée de 1. La mise à jour de la valeur se fait alors pour tous les états, en fonction de l'erreur de différence temporelle δ_i (d'ordre 1), l'erreur étant pondérée par la trace de cet état :

$$\forall s \in S, \hat{V}(s) \leftarrow \hat{V}(s) + \alpha_i \Lambda(s_i) \delta_i \text{ où } \delta_i = r_i + \gamma \hat{V}(s_{i+1}) - \hat{V}(s_i) \tag{2.31}$$

Ainsi l'erreur δ_i est propagée à tous les états visités par l'agent sur la trajectoire menant à s_i, la pondération de cette mise à jour étant d'autant plus faible que l'état a été visité il y a longtemps. Nous illustrons ce caractère global des mises à jour sur la figure 2.2. La trajectoire suivie est représentée sur la figure de gauche et seul l'état terminal fournit une récompense non nulle. Sur la figure du milieu est illustré l'effet d'une mise à jour faite par TD(0), c'est-à-dire TD, le premier algorithme que nous ayons présenté. Seule la valeur de l'état à partir duquel est obtenue la récompense est mise à jour, tous les autres états sont ignorés. Sur la figure de droite est illustré l'effet d'une mise à jour faite par TD(λ), le facteur d'éligibilité n'étant plus nul. Tous les états de la trajectoire sont mis à jour, l'amplitude de cette dernière étant d'autant plus forte que l'état correspondant est proche de la récompense (la notion de distance n'étant pas euclidienne, mais liée à la trajectoire). La rétropropagation des récompenses informatives est donc beaucoup plus efficace dans ce cas.

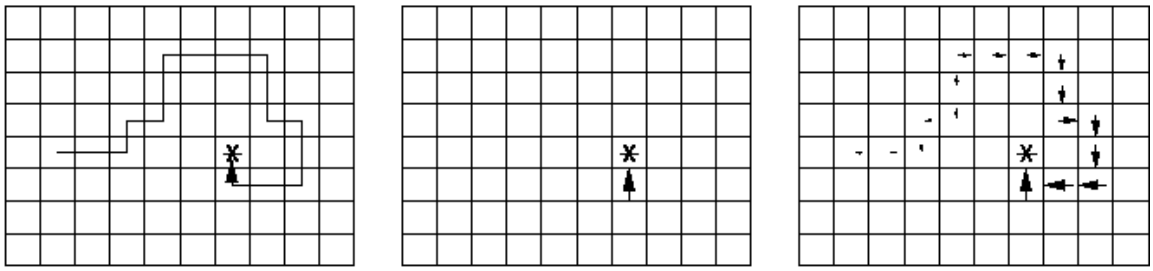


FIG. 2.2 – Illustration des traces d'éligibilité [105].

Cette approche, appelée TD(λ) "vue arrière", ou plus simplement TD(λ), est résumée dans l'algorithme 2.6 (lorsque la fonction de valeur V ou l'ensemble des traces Λ sont écrits sans argument, ils désignent le vecteur correspondant). De la même façon que l'algorithme SARSA se déduit facilement de l'algorithme TD une fois la notion de Q-fonction introduite, l'algorithme SARSA(λ) se déduit facilement de l'algorithme TD(λ) (en maintenant une trace pour chaque couple état-action). Nous ne le présentons pas ici. Notons également qu'il existe quelques variantes de l'algorithme TD(λ), selon la façon dont la trace de l'état courant est incrémentée. Le lecteur peut se référer à [105] pour plus de détails.

Nous allons à présent nous intéresser au cas de l'apprentissage *off-policy* et expliquer en quoi le principe de trace d'éligibilité ne s'y applique pas (du moins directement). Plaçons-nous dans le cadre de l'évaluation de la fonction de qualité et supposons qu'une politique cible π doit être apprise alors qu'une politique comportementale b est suivie. Soit une transition $(s_i, a_i, r_i, s_{i+1}, a_{i+1})$ observée. Une mise à jour permettant l'apprentissage de la fonction de qualité de π est la suivante :

$$\hat{Q}^\pi(s_i, a_i) \leftarrow \hat{Q}^\pi(s_i, a_i) + \alpha_i \left(r_i + \gamma \hat{Q}^\pi(s_{i+1}, \pi(s_{i+1})) - \hat{Q}^\pi(s_i, a_i) \right) \tag{2.32}$$

Par rapport à la mise à jour classique de SARSA, celle-ci permet d'assurer la convergence car l'action a_{i+1}, choisie par la politique b, y est remplacée par l'action $\pi(s_{i+1})$ qui correspond à la politique π. L'action a_i n'a pas besoin d'être remplacée, étant donné que la Q-fonction permet un degré de liberté sur la première action. Si une version *off-policy* de SARSA peut être assez naturellement dérivée, ce n'est pas le cas pour SARSA(λ). En effet, plaçons-nous dans le cas de la "vue avant". Si l'erreur de différence temporelle d'ordre 1 reste

Algorithme 2.6 : TD(λ)

Initialisation;
Initialisation de la fonction de valeur $\hat{V}$ et des traces d'éligibilité $\Lambda = \mathbf{0}$;

pour $i = 1, 2, \ldots$ **faire**

Observer la transition (s_i, r_i, s_{i+1});

Calcul de l'erreur de différence temporelle ;
$\delta_i = r_i + \gamma \hat{V}(s_{i+1}) - \hat{V}(s_i)$;

Incrément de la trace de l'état courant ;
$\Lambda(s_i) \leftarrow \Lambda(s_i) + 1$;

Mise à jour de la valeur et des traces ;
$V \leftarrow V + \alpha_i \delta_i \Lambda$;
$\Lambda \leftarrow \gamma\lambda\Lambda$;

toujours valable, ce grâce au degré de liberté sur la première action, les erreurs de différences temporelles d'ordres supérieurs du type (2.27) ne sont plus valables. En effet elles se basent sur une prédiction à l'ordre k des récompenses, le cumul des récompenses étant échantillonné selon la politique b, alors que c'est le cumul de récompenses échantillonné selon π qui est nécessaire. Utiliser l'algorithme SARSA(λ) dans le cas *off-policy*, même avec une mise à jour modifiée du type (2.32), conduirait à une estimation biaisée de la fonction de qualité Q^π. Des approches existent pour combiner apprentissage *off-policy* et traces d'éligibilité. Elles sont principalement basées sur la notion d'échantillonnage d'importance[8] [105, 86, 54]. Cependant, il y en a assez peu dans la littérature et nous ne les développons pas plus avant ici.

Si nous avons développé cette notion de trace d'éligibilité et le problème qu'elle pose lorsqu'elle est combinée à un apprentissage *off-policy*, c'est que certaines variations algorithmiques du cadre de travail proposé dans ce manuscrit présentent des aspects fort similaires, même si les arguments développés pour obtenir nos algorithmes sont sensiblement différents. Cela est discuté chapitre 4.

2.2.3 Apprentissage et contrôle

Nous avons vu jusqu'à présent comment apprendre la fonction de valeur, de la politique courante pour TD et SARSA, de la politique optimale pour le Q-learning, à partir d'interactions avec l'environnement. Cependant, l'objectif de l'apprentissage par renforcement est d'apprendre le contrôle optimal du système dynamique et c'est de cette composante de contrôle dont nous parlons à présent. Pour TD et SARSA, la politique courante est évaluée, mais il reste encore à l'améliorer. Pour le Q-learning, la politique optimale est apprise directement. Cependant, dans le paradigme le plus général de l'apprentissage par renforcement, l'agent ne se contente pas d'observer des transitions, il contrôle le système en même temps. Un autre point est que l'algorithme Q-learning implique de calculer le maximum sur l'ensemble des actions à chaque mise à jour, ce qui peut s'avérer très coûteux

[8] L'échantillonnage d'importance (ou échantillonnage préférentiel, *importance sampling* en anglais) est une technique très générale qui vise à estimer une quantité liée à une distribution particulière à partir d'échantillons générés suivant une autre distribution.

selon la taille de l'espace d'action et en conséquence lui faire préférer d'autres approches (le même problème se pose pour la détermination d'une politique gloutonne et donc dans un cadre d'itération de la politique).

Nous présentons d'abord le concept d'itération de la politique généralisée ainsi que le dilemme entre exploration et exploitation qui lui est intimement lié. Nous présentons également brièvement les architectures dites acteur-critique ainsi qu'un certain nombre de problèmes induits par cette imbrication de l'apprentissage et du contrôle, parfois peu mentionnés dans la littérature comme la non-stationnarité potentielle de la fonction de valeur cible induite.

Itération de la politique généralisée

Le terme itération de la politique généralisée [105] (*generalized policy iteration* en anglais) réfère à l'idée générale d'interaction des processus d'évaluation et d'amélioration de la politique, indépendamment de la granularité et d'autres détails, come l'aspect (a)synchrone ou la connaissance *a priori* d'un modèle. Il est illustré figure 2.3.

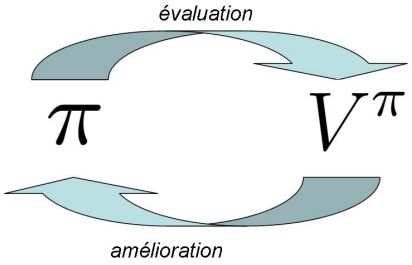


FIG. 2.3 – Itération de la politique généralisée.

Le schéma d'itération de la politique présenté section 2.1 sur la programmation dynamique entre dans ce cadre. Un autre exemple est l'itération de la politique dite optimiste, pour laquelle on n'attend pas d'avoir fini d'évaluer une politique donnée avant de l'améliorer. Les architectures acteur-critique que nous présentons par la suite entrent également dans ce cadre. Ce schéma général est important pour l'apprentissage par renforcement. En effet, si l'on considère les algorithmes de différences temporelles que sont TD et SARSA, ils peuvent être utilisés pour évaluer la fonction de valeur ou de qualité, mais ils doivent être combinés à un schéma d'amélioration de la politique. Un problème lié à l'itération de la politique généralisée est le dilemme entre exploration et exploitation.

Dilemme entre exploration et exploitation

Dans le paradigme de l'apprentissage par renforcement, l'agent doit contrôler le système tout en apprenant le contrôle optimal et ce sans connaissance *a priori* du modèle. Une question assez naturelle se pose alors. L'agent va-t-il choisir une action qu'il juge optimale par rapport à sa connaissance (imparfaite) du monde, c'est-à-dire une action exploitante (action gloutonne sur base de l'estimation courante de la fonction de valeur) ? Va-t-il plutôt choisir une action qu'il juge sous-optimale par rapport à sa connaissance actuelle du monde, c'est-à-dire une action exploratrice, de façon à améliorer sa connaissance

du monde ? Ce problème est connu sous le nom de dilemme entre exploration et exploitation. Plusieurs schémas ont été proposés dans la littérature, le plus connu étant peut-être la politique ϵ-gloutonne (ϵ-*greedy* en anglais). Son principe est de choisir une action gloutonne avec une probabilité $1 - \epsilon$ et une action exploratrice (*id est* au hasard parmi les actions non-gloutonnes) avec une probabilité ϵ. Il est possible de montrer que l'algorithme SARSA combiné à ce schéma d'exploration/exploitation (qui est une forme d'itération de la politique généralisée) converge vers la Q-fonction optimale, sous certaines conditions. Une autre approche classique consiste à générer les actions selon une distribution de Gibbs. Le principe est de définir une politique stochastique de contrôle qui associe une probabilité de choisir une action d'autant plus grande que sa valeur sera importante, autrement dit (le rôle de l'exponentielle étant de projeter les valeurs dans $\mathbb{R}^+$, sans altérer leur ordre) :

$$\pi(a|s) = \frac{\exp\left(\frac{\hat{Q}(s,a)}{\tau}\right)}{\sum_{b \in A} \exp\left(\frac{\hat{Q}(s,b)}{\tau}\right)} \tag{2.33}$$

Dans cette dernière expression, τ est un facteur dit de température. Plus ce facteur sera grand et plus les actions seront équiprobables. Une approche classique consiste à commencer avec une forte température (politique stochastique quasi-uniforme), puis à baisser cette dernière au cours du temps (politique de plus en plus déterministe, favorisant les fortes valeurs estimées).

Les deux approches que nous venons de présenter sont parmi les plus simples et les plus classiques. Elles permettent d'assurer toutes les deux que toutes les paires état-action sont visitées un nombre suffisant de fois, condition généralement nécessaire à la convergence des algorithmes de différences temporelles. Cependant, ces schémas d'exploration ne sont pas dirigés. En effet, lorsque l'agent choisit une action exploratrice, il peut être souhaitable qu'il ne choisisse pas une action qui puisse être dangereuse pour le système (ce qui est en partie assuré par le schéma de Gibbs, si la Q-fonction est correctement estimée), ou bien encore qu'il choisisse une action plus incertaine (*i.e.* dont la valeur est incertaine) de façon à accélérer l'apprentissage. Il existe de nombreux travaux visant à proposer de nouveaux schémas d'exploration/exploitation, comme par exemple [4, 64, 74, 103, 101, 122]. Notons que la plupart de ces approches se basent sur l'utilisation d'une incertitude liée à la valeur et diffèrent par la façon dont cette incertitude est utilisée.

Nous insistons sur cette notion d'incertitude sur les valeurs. En effet, quand l'agent doit prendre une décision, c'est-à-dire choisir entre exploration et exploitation et le cas échéant dans quelle direction explorer l'environnement, il ne peut que bénéficier de cette information d'incertitude. Indépendamment de la façon dont elle est utilisée, nous pensons important qu'un algorithme permettant d'estimer la fonction de qualité évalue également l'incertitude liée à cette estimation et nos travaux sont orientés dans ce sens, même si la façon effective d'utiliser cette information reste principalement du domaine des perspectives (quelques propositions seront tout de même faites dans le chapitre 6).

Architectures acteur-critique

Les architectures acteur-critique sont une forme particulière d'itération de la politique généralisée. Nous les décrivons brièvement ici et les discutons plus avant dans le chapitre 7. Leur principe général est illustré figure 2.4, adaptée de [105].

Dans les approches que nous avons vues jusqu'à présent, la politique est déduite de la fonction de qualité (politique ϵ-gloutonne), ou est définie à partir de cette dernière (politique

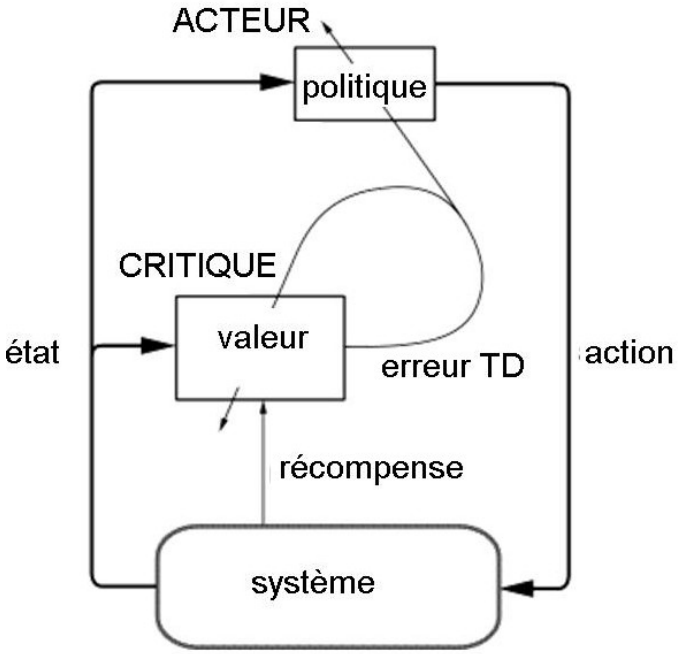


FIG. 2.4 – Architecture acteur-critique générale.

de Gibbs). Au contraire, pour les architectures acteur-critique, la politique (l'acteur) a une représentation séparée de la fonction de qualité (le critique), l'ensemble des deux formant l'agent. Le principe général est le suivant. A chaque pas de temps, l'acteur choisit une action et l'applique au système, qui change alors d'état. Cet état est transmis au critique et à l'acteur. Le critique reçoit également une information de récompense, ce qui lui permet de calculer une erreur de différence temporelle. Cette dernière est utilisée d'une part pour mettre à jour le critique (la fonction de valeur ou de qualité) et d'autre part pour mettre à jour l'acteur. De façon très générale, si l'erreur est positive, l'action correspondante devrait être renforcée (la récompense observée est plus grande que celle prédite), le contraire si l'erreur est négative. C'est la base psychologique du renforcement [113]. Une fois l'acteur mis à jour, il choisit une nouvelle action à appliquer au système et le cycle des interactions et mises à jour continue.

L'un des intérêts indéniables de ce type de méthode est de pouvoir s'affranchir du calcul du maximum sur les actions. En effet, nous avons vu que le Q-learning nécessite une telle recherche de maxima à chaque mise à jour. Si l'on considère un schéma d'itération de la politique généralisée (politique ϵ-gloutonne par exemple), les phases d'amélioration de la politique nécessitent également une recherche de maximum sur les actions (choix de l'action gloutonne). Cela peut s'avérer très coûteux si l'espace des actions possède un grand cardinal. Pour les architectures acteur-critique, les paramètres de l'acteur sont mis à jour en fonction de l'erreur de différence temporelle calculée par le critique, ce qui permet d'améliorer la politique sans passer par une recherche de maximum. Un autre intérêt de ce type d'approche vient du fait que lorsque la politique est définie à partir de la Q-fonction, il est difficile d'obtenir des garanties sur la qualité de la politique, même si l'on a des garanties sur la fonction de qualité (comme une borne sur l'erreur en norme infinie fonction

du nombre d'itérations). On peut donc avoir une très bonne estimation de la Q-fonction optimale et une politique associée loin d'être optimale (le contraire pouvant aussi être vrai). Cela vient du fait que c'est moins la valeur exacte d'un couple état-action que la différence des valeurs des différentes actions pour un même état qui est important. Pour les architectures acteur-critique, la politique possède sa propre représentation et le critique n'est là que pour évaluer sa performance, ce qui a tendance à réduire le problème que nous venons d'exposer.

De l'importance de traquer la fonction de valeur cible

Le dernier point que nous passons en revue concernant l'imbrication de l'apprentissage et du contrôle est assez peu abordé dans la littérature. Dans le cas général d'un schéma d'itération de la politique généralisée, nous avons vu que des phases d'évaluation et d'amélioration de la politique se succèdent. En conséquence, la fonction de valeur que va chercher à estimer l'algorithme de différences temporelles va évoluer au cours du temps, en même temps que la politique courante. Cela se traduit par un problème de non-stationnarité de la fonction de valeur cible de l'algorithme d'apprentissage, ce qui peut être un problème s'il la suppose stationnaire. Cela apparaît notamment dans le cadre des architectures acteur-critique, ou la politique définie par l'acteur change potentiellement après chaque action. Pour y remédier, il est proposé dans [13] d'utiliser deux échelles de temps. En pratique, des taux d'apprentissage différents sont choisis pour l'acteur et le critique, de façon à ce que le premier apparaisse comme stationnaire au second. Nous y reviendrons chapitre 7. Pour notre part, nous pensons qu'il est préférable de "traquer" la fonction de valeur plutôt que de chercher à converger vers cette dernière[9]. Cela devrait permettre de s'affranchir de ce problème causé par l'imbrication du contrôle et de l'apprentissage. Cela devrait également permettre de prendre en compte des environnements non-stationnaires. De plus, même en cas de stationnarité de la cible et de l'environnement, il existe des raisons pour préférer une traque à une convergence, comme présenté par [106]. Nous pensons ce point important et avons orienté nos travaux dans cette optique, comme nous le montrons par la suite.

2.3 Approximation de la fonction de valeur

Nous avons présenté les principes de la programmation dynamique et des algorithmes de différences temporelles. Jusqu'à présent, les espaces d'état et d'action ont été supposés suffisamment petits pour qu'une représentation tabulaire soit possible. Nous nous intéressons maintenant au cas où l'espace d'état est trop grand, ce qui nécessite d'introduire un schéma d'approximation. Nous présentons dans un premier temps la problématique générale de l'approximation de fonction de valeur, avant de passer en revue de façon unifiée un certain nombre d'approches paramétriques traitant ce problème. Nous discutons finalement les qualités que l'on souhaite d'un approximateur de fonction de valeur, ce qui a motivé nos travaux dont les premiers résultats sont présentés dans le chapitre suivant.

[9] Par convergence nous entendons la recherche d'une solution supposée stationnaire et qui ne nécessite donc plus de modifications une fois trouvée. Par traque nous entendons la recherche d'une solution sans supposer qu'elle est stationnaire, ce qui ajoute une notion d'adaptation.

2.3.1 Problématique

Le problème que nous posons ici est l'estimation de la fonction de valeur lorsque l'espace d'état est trop grand pour les approches usuelles[10]. On pourrait aller jusqu'à supposer l'espace d'état continu. Cependant, cela implique de considérer l'opérateur sup en lieu et place de l'opérateur max et des intégrales en lieu et place des sommes, entre autres problèmes d'ordre technique [16]. Nous supposons donc dans la suite de ce manuscrit que l'espace d'état est fini mais de cardinal beaucoup trop grand pour que les approches présentées jusqu'ici conviennent.

Le problème posé par les grands espaces d'état est souvent appelé généralisation, il est en fait double. Le premier aspect est la nécessité d'estimer la fonction de valeur pour l'ensemble des états visités, alors qu'ils sont trop nombreux pour une représentation exacte. Les approches que nous présentons reposent sur l'approximation paramétrique : plutôt que de représenter la fonction de valeur par un vecteur avec autant de composantes que d'états, elle est représentée par un nombre de paramètres très largement inférieur au nombre d'états. Autrement dit, on passe d'une représentation du type $\hat{V}(s)$ à une autre du type $\hat{V}_\theta(s)$, par exemple la représentation linéaire formulée par $\hat{V}_\theta(s) = \theta^T \phi(s)$ où θ est le vecteur de paramètres, $\phi(s)$ un vecteur de fonctions de bases et $|\theta| \ll |S|$. Il s'agit alors d'apprendre une bonne représentation compacte de la fonction de valeur, en sachant que de façon générale la fonction d'intérêt ne peut pas être représentée parfaitement sous cette forme. L'autre aspect est la nécessité de pouvoir estimer la valeur d'un état qui n'a jamais été visité à partir de ce qui a été appris dans des états "proches"[11], ce qui ressemble au problème d'interpolation en régression classique.

Nous passons en revue dans la suite un certain nombre de méthodes (paramétriques) traitant le problème d'approximation de la fonction de valeur. Cependant, notons dès à présent que la plupart de ces algorithmes sont conçus pour résoudre ce problème en particulier, mais ignorent d'autres aspects qui pourraient bénéficier au traitement général de l'apprentissage par renforcement. Plus précisément, ces approches traitent l'estimation de la fonction de valeur dans de grands espaces d'état, mais elles ne permettent pas à la fois de prendre en compte des problèmes non-stationnaires et de dériver une information d'incertitude utilisable pour aborder le dilemme entre exploration et exploitation, alors que nous avons montré précédemment que ces aspects sont importants pour le paradigme général. De plus, nous nous concentrons exclusivement sur des approches dites en ligne, c'est-à-dire qui permettent une mise à jour de la représentation de la fonction de valeur après chaque interaction, sans nécessiter (explicitement) de réutiliser toutes les informations obtenues sur une trajectoire. D'autres méthodes, dites hors-ligne, nécessitent des trajectoires complètes pour mettre à jour la représentation de la fonction de valeur. Nous ne les aborderons pas plus avant dans ce manuscrit, mais l'on peut toutefois citer quelques articles de référence [9, 52, 80, 73, 31].

2.3.2 Etat de l'art

L'ensemble des algorithmes que nous passons en revue font intervenir une formule de mise à jour des paramètres de la forme générique suivante :

$$\theta_i = \theta_{i-1} + K_i \delta_i \tag{2.34}$$

[10] C'est ce que Bellman [7] appelle la malédiction de la dimensionnalité, c'est-à-dire l'explosion combinatoire de la complexité de l'optimisation lorsque la taille du problème augmente.

[11] La notion de métrique dans un MDP n'est pas triviale, voir par exemple [34, 35, 36].

Dans cette expression, θ_{i-1} est l'ancienne représentation (paramétrique) de la fonction de valeur, θ_i est la mise à jour de cette représentation en accord avec la dernière transition observée, δ_i est l'erreur de différence temporelle et K_i est un gain indiquant dans quelle direction corriger la représentation de la fonction de valeur. Nous discutons d'abord les deux premiers termes, qui sont communs à l'ensemble des approches, avant de discuter les gains qui sont propres à chaque algorithme et particulièrement la façon dont ils sont obtenus.

Si les espaces d'état S et d'action A ont des cardinaux suffisamment petits, une représentation exacte de la fonction de valeur est possible et θ est alors un vecteur avec autant de composantes qu'il y a d'états (ou de couples état-action pour la Q-fonction). C'est la représentation tabulaire qui a déjà été présentée. Si ces espaces sont trop larges[12] une approximation $\hat{V}_\theta(s)$ est nécessaire. Un choix classique en apprentissage par renforcement est la paramétrisation linéaire, pour laquelle la fonction de valeur est approchée comme suit :

$$\hat{V}_\theta(s) = \sum_{j=1}^{p} w_j \phi_j(s) = \theta^T \phi(s) \tag{2.35}$$
$$\text{où } \theta^T = \begin{pmatrix} w_1 & \dots & w_p \end{pmatrix} \text{ et } \phi(s)^T = \begin{pmatrix} \phi_1(s) & \dots & \phi_p(s) \end{pmatrix}$$

Dans cette expression, le vecteur de paramètres θ est constitué de l'ensemble des poids w_j et $(\phi_j)_{1 \leq j \leq p}$ est un ensemble de fonctions de base définies à l'avance. Un exemple (classique) parmi d'autres de fonction de base est le noyau gaussien de centre s_j et d'écart-type σ_j :

$$\phi_j(s) = \exp\left(-\frac{(s - s_j)^2}{2\sigma_j^2}\right) \tag{2.36}$$

Beaucoup d'algorithmes d'approximation de la fonction de valeur nécessitent une telle représentation pour assurer la convergence [95], ou même pour être appliqués [19]. D'autres représentations sont possibles, comme par exemple un réseau de neurones pour lequel θ est composé des poids des connexions synaptiques associées. Dans ce cas, la paramétrisation peut être non-linéaire. Remarquons que le choix de l'architecture de l'approximateur est un problème en soit. Il existe des approches visant à déterminer de façon automatique cette architecture en fonction du problème à traiter [67, 78, 60, 125, 87]. Cependant, dans le reste de ce manuscrit[13] nous supposerons cette architecture choisie à l'avance par le praticien et nous nous intéresserons à la façon d'apprendre la valeur des paramètres.

Dans l'équation (2.34), le terme δ_i est l'erreur de différence temporelle (erreur TD pour *Temporal Difference*). Supposons qu'au temps i la transition $(s_i, a_i, r_i, s_{i+1}, a_{i+1})$ soit observée. Pour les algorithmes de renforcement de type TD, c'est-à-dire ceux qui visent l'évaluation de la fonction de valeur pour une politique donnée π, l'erreur TD est :

$$\delta_i = r_i + \gamma \hat{V}_{\theta_{i-1}}(s_{i+1}) - \hat{V}_{\theta_{i-1}}(s_i) \tag{2.37}$$

Pour les algorithmes de type SARSA, c'est-à-dire ceux qui ont pour but d'évaluer la fonction de qualité d'une politique donnée π, étant donnée l'approximation $\hat{Q}_{\theta_{i-1}}$ de la Q-fonction, l'erreur TD est :

$$\delta_i = r_i + \gamma \hat{Q}_{\theta_{i-1}}(s_{i+1}, a_{i+1}) - \hat{Q}_{\theta_{i-1}}(s_i, a_i) \tag{2.38}$$

[12] Nous nous concentrerons sur les espaces d'état, en supposant le nombre d'actions relativement faible, jusqu'à mention du contraire.

[13] Dans le chapitre 8, nous proposons tout de même une semi-automatisation de la paramétrisation, qui est une adaptation d'une approche présentée par [28].

Enfin, pour les algorithms de type Q-learning, c'est-à-dire ceux dont l'objectif est de calculer la Q-fonction optimale, l'erreur TD est de la forme suivante :

$$\delta_i = r_i + \gamma \max_{b \in A} \hat{Q}_{\theta_{i-1}}(s_{i+1}, b) - \hat{Q}_{\theta_{i-1}}(s_i, a_i) \tag{2.39}$$

Le type de différence temporelle utilisé est directement lié au type d'équation de Bellman à résoudre et donc au fait que l'algorithme associé appartient à la famille de l'itération de la politique ou de la valeur.

Le gain K_i quant à lui est dépendant de l'algorithme. Nous présentons un certain nombre d'entre eux, en commençant par une reformulation des algorithmes classiques déjà présentés section 2.2.2.

Algorithmes classiques de différences temporelles

Les algorithmes TD, SARSA et Q-learning peuvent se mettre sous la forme (2.34) en choisissant l'erreur de différence temporelle *ad hoc*, c'est-à-dire respectivement (2.37-2.39) et une représentation tabulaire pour la fonction de valeur ou la Q-fonction, c'est-à-dire :

$$\hat{V}_\theta(s) = \theta^T e(s) \tag{2.40}$$

où $e(s)$ est un vecteur unitaire nul partout sauf en la composante correspondant à l'état s (et une représentation similaire de la Q-fonction est possible en considérant l'espace joint état-action). Pour cette famille d'algorithmes et pour une transition $(s_i, a_i, r_i, s_{i+1}, a_{i+1})$, le gain se met sous la forme :

$$K_i = \alpha_i e_i \tag{2.41}$$

où α_i est le taux d'apprentissage usuel et $e_i = e(s_i)$ est le vecteur unitaire présenté ci-avant ($e_i = e(s_i, a_i)$ si la Q-fonction est estimée). Il est aisé de vérifier que les algorithmes 2.3, 2.4 et 2.5 se mettent sous cette forme générale.

Il est également possible d'écrire l'algorithme TD(λ) sous cette forme (SARSA(λ) également, mais nous ne le ferons pas, car l'extension est facile). Si une représentation tabulaire est choisie et si l'erreur de différence temporelle (2.37) est considérée, alors TD(λ) peut se mettre sous la forme générale (2.34) en choisissant le gain suivant :

$$K_i = \alpha_i \sum_{j=1}^{i} (\gamma\lambda)^{i-j} e_j \tag{2.42}$$

Cette mise à jour est une réécriture de l'algorithme 2.6. Maintenant que nous avons montré comment les algorithmes tabulaires entraient dans le cadre proposé, nous nous intéressons aux algorithmes qui utilisent une approximation de la fonction de valeur.

Algorithmes directs

Les algorithmes que nous présentons maintenant sont appelés algorithmes de différences temporelles avec approximation de la fonction de valeur [105], ou encore algorithmes directs en suivant la dénomination de [6]. La paramétrisation de la fonction de valeur est quelconque. Remarquons tout de même que des preuves de convergence n'existent que pour le cas linéaire [114, 95] et ce sous certaines conditions, bien que des applications couronnées

de succès existent avec une paramétrisation non-linéaire comme TD-Gammon[14] [111]. Les trois types d'erreurs de différences temporelles (2.37-2.39) peuvent être considérés. Le raisonnement étant le même, nous nous concentrons sur l'algorithme TD avec approximation de la valeur.

Si la fonction de valeur pouvait être directement observée, c'est-à-dire si l'on avait une base d'exemples composée de couples entrée-sortie du type état-valeur de l'état (éventuellement bruitée) $(s_i, v_i = V(s_i) + n_i)$ (ici n_i est un bruit blanc et v_i une réalisation bruitée de la fonction de valeur), alors une approche de type descente de gradient, classique en apprentissage supervisé, pourrait être envisagée. La fonction coût de ce problème de régression est la suivante :

$$J(\theta) = \|V - \hat{V}_\theta\|^2 \tag{2.43}$$

où $\|.\|^2$ est la norme L_2 usuelle. Une approche classique pour minimiser ce coût quadratique est d'effectuer une descente de gradient, ce qui correspond à la mise à jour suivante :

$$\begin{aligned}\theta_i &= \theta_{i-1} - \frac{1}{2}\alpha_i \nabla_\theta \left(v_i - \hat{V}_{\theta_{i-1}}(s_i)\right)^2 \\ &= \theta_{i-1} + \alpha_i \nabla_\theta \hat{V}_{\theta_{i-1}}(s_i) \left(v_i - \hat{V}_{\theta_{i-1}}(s_i)\right)\end{aligned} \tag{2.44}$$

Malheureusement, dans le paradigme de l'apprentissage par renforcement, la valeur d'un état n'est accessible que via les récompenses observées. Le principe des algorithmes dits directs est donc de remplacer la valeur v_i, non directement observable, par une approximation basée sur la récompense et l'estimation de la valeur de l'état suivant, soit $r_i + \gamma \hat{V}_{\theta_{i-1}}(s_{i+1})$, le raisonnement étant le même que celui développé pour l'équation (2.15). La mise à jour correspondante est alors :

$$\theta_i = \theta_{i-1} + \alpha_i \nabla_\theta \hat{V}_{\theta_{i-1}}(s_i) \left(r_i + \gamma \hat{V}_{\theta_{i-1}}(s_{i+1}) - \hat{V}_{\theta_{i-1}}(s_i)\right) \tag{2.45}$$

L'algorithme peut donc se mettre sous la forme de l'équation (2.34), le gain étant défini par :

$$K_i = \nabla_\theta \hat{V}_{\theta_{i-1}}(s_i) \tag{2.46}$$

Cette approche qui consiste à remplacer la valeur d'un état par une approximation de cette valeur basée sur la récompense observée et l'estimation de la valeur de l'état suivant est appelée *bootstrapping* [105]. Si le gain est remplacé par le gradient de la Q-fonction et que les erreurs de différences temporelles (2.38) ou (2.39) sont considérées, les généralisations correspondantes de SARSA et Q-learning sont obtenues. Il est également à noter que dans le cas particulier d'une représentation tabulaire les algorithmes directs sont les algorithmes classiques de différences temporelles. En effet, dans ce cas nous avons $K_i = \nabla_\theta(\theta_{i-1}^T e(s_i)) = e(s_i)$.

Algorithmes résiduels

Comme pour les algorithmes directs, nous présentons l'approche pour l'évaluation de la fonction de valeur et les extensions se font en remplaçant la fonction de valeur par la fonction de qualité et en considérant l'une des deux équations de Bellman associées. La

[14] TD-Gammon est actuellement le meilleur programme informatique jouant au backgammon, il atteint le niveau des premiers joueurs mondiaux humains. Cela a d'ailleurs contribué à relancer la recherche en apprentissage par renforcement dans les années 90.

paramétrisation de la fonction de valeur est ici quelconque (non nécessairement linéaire). Nous rappelons l'équation d'évaluation de Bellman pour l'évaluation de la valeur, ainsi que l'opérateur de Bellman T^π associé (2.12). Dans la suite, nous omettons l'exposant π et écrirons plus simplement T cet opérateur. La fonction de valeur de la politique π est le point fixe de cet opérateur. Le principe des algorithmes résiduels est de minimiser le résidu quadratique de Bellman, c'est-à-dire le carré de la difference entre les membres droit et gauche de l'équation correspondante. La fonction de coût est donc :

$$J(\theta) = \|\hat{V}_\theta - T\hat{V}_\theta\|^2 \tag{2.47}$$

Ce coût est assez légitime : s'il est annulé, l'équation de Bellman est alors vérifiée. Sous cette forme, il prend naturellement en compte la structure du MDP, via l'opérateur T. Cependant, les probabilités de transitions n'étant pas connues, nous définissons l'opérateur $\hat{T}$, qui est une version échantillonnée de l'opérateur T :

$$\hat{T}V(s) = R(s, \pi(s), s') + \gamma V(s') \text{ avec } s' \sim p(.|s, \pi(s)) \tag{2.48}$$

Notons que si les transitions sont déterministes, alors les opérateurs T et $\hat{T}$ sont égaux, ce qui est un cas particulier important.

Le principe des algorithmes résiduels tels qu'introduits par [6] est d'effectuer une descente de gradient de façon à minimiser le coût (2.47), en remplaçant l'opérateur T par l'opérateur $\hat{T}$, étant donné que des trajectoires sont observées et que les probabilités de transitions ne sont pas connues. La mise à jour correspondante pour une transition (s_i, r_i, s_{i+1}) est :

$$\begin{aligned}
\theta_i &= \theta_{i-1} - \frac{1}{2}\alpha_i \nabla_\theta \left(\hat{V}_{\theta_{i-1}}(s_i) - \hat{T}\hat{V}_{\theta_{i-1}}(s_i)\right)^2 \\
&= \theta_{i-1} - \frac{1}{2}\alpha_i \nabla_\theta \left(r_i + \gamma \hat{V}_{\theta_{i-1}}(s_{i+1}) - \hat{V}_{\theta_{i-1}}(s_i)\right)^2 \\
&= \theta_{i-1} + \alpha_i \nabla_\theta \left(\hat{V}_{\theta_{i-1}}(s_i) - \gamma \hat{V}_{\theta_{i-1}}(s_{i+1})\right)\left(r_i + \gamma \hat{V}_{\theta_{i-1}}(s_{i+1}) - \hat{V}_{\theta_{i-1}}(s_i)\right)
\end{aligned} \tag{2.49}$$

Cet algorithme peut donc se mettre sous la forme générale de la mise à jour (2.34), en choisissant comme gain :

$$K_i = \alpha_i \nabla_\theta \left(\hat{V}_{\theta_{i-1}}(s_i) - \gamma \hat{V}_{\theta_{i-1}}(s_{i+1})\right) \tag{2.50}$$

Plusieurs remarques peuvent êtres faites concernant cet algorithme. Premièrement, pour l'extension à la Q-fonction et à l'équation d'optimalité de Bellman, le gain devient pour une transition observée (s_i, a_i, r_i, s_{i+1}) :

$$K_i = \alpha_i \nabla_\theta \left(\hat{Q}_{\theta_{i-1}}(s_i, a_i) - \max_{b \in A} \gamma \hat{Q}_{\theta_{i-1}}(s_{i+1}, b)\right) \tag{2.51}$$

Il implique donc de calculer le gradient de l'opérateur max. S'il est présenté dans [6], cet algorithme est très peu pratique, à cause de ce problème. Deuxièmement, si les transitions sont stochastiques, la fonction de coût effectivement minimisée est biaisée par rapport au coût de l'équation (2.47), à cause de l'emploi de l'opérateur échantillonné en lieu et place de l'opérateur de Bellman. En effet, il est possible de vérifier [3] que :

$$E_{s'|s,\pi(s)}[(\hat{T}V(s) - V(s))^2] = (TV(s) - V(s))^2 + \mathrm{Var}_{s'|s,\pi(s)}(\hat{T}V(s)) \tag{2.52}$$

Le biais est donc un terme de variance qui a tendance à favoriser les fonctions régulières (si V est constante, le biais est nul). Le biais agit donc comme un terme de régularisation. Si ajouter un tel terme est une approche classique en apprentissage supervisé (notamment pour éviter ce que l'on appelle le sur-apprentissage, c'est-à-dire apprendre "par cœur" la base d'exemple), il n'est pas intéressant ici dans la mesure où il ne peux pas être contrôlé. Il est proposé dans [6] de débiaiser l'estimateur en faisant du double échantillonnage, c'est-à-dire échantillonner de façon indépendante deux états suivants pour un état donné et utiliser l'un de ces états pour le gain et l'autre pour l'erreur de différence temporelle. Soient s'_{i+1} et s''_{i+1} deux états échantillonnés indépendamment selon les probabilités de transitions conditionnées à s_i et a_i. La mise à jour doit alors être :

$$\theta_i = \theta_{i-1} + \alpha_i \nabla_\theta \left(\hat{V}_{\theta_{i-1}}(s_i) - \gamma \hat{V}_{\theta_{i-1}}(s'_{i+1})\right)\left(r_i + \gamma \hat{V}_{\theta_{i-1}}(s''_{i+1}) - \hat{V}_{\theta_{i-1}}(s_i)\right) \tag{2.53}$$

Cependant, cela suppose de disposer d'un modèle génératif et ralentit empiriquement l'apprentissage.

Différences temporelles par moindres carrés

Nous présentons maintenant l'algorithme LSTD [19] (*Least-Squares Temporal Differences*), qui permet l'évaluation de la fonction de valeur (ou de qualité), mais pas l'optimisation directe de la Q-fonction. Nous le présentons pour l'évaluation de la fonction de valeur, l'extension au cas de la fonction de qualité étant immédiate. La paramétrisation est supposée linéaire, comme spécifié par l'équation (2.35). L'objectif ici est toujours de minimiser le résidu quadratique de Bellman (2.47), avec une approche par moindres carrés plutôt que par descente de gradient et avec un autre type de réponse pour le problème causé par les transitions stochastiques. Le coût empirique que l'on cherche à minimiser est le suivant :

$$J_i(\theta) = \sum_{j=1}^{i} \left(\hat{V}_\theta(s_j) - T\hat{V}_\theta(s_j)\right)^2 \tag{2.54}$$

Pour simplifier notre propos, supposons comme dans [19] que la fonction de valeur est effectivement linéaire en les paramètres (2.35) et qu'il existe donc un vecteur θ^* tel que $V = \hat{V}_{\theta^*}$. De l'équation d'évaluation de Bellman on peut alors obtenir une forme de régression sur les paramètres, en notant $\bar{R}(s,a) = E_{s'|s,a}[R(s,a,s')]$ la récompense moyenne sur une transition :

$$\begin{aligned} V(s) &= \sum_{s' \in S} p(s'|s,\pi(s)) \left(R(s,\pi(s),s') + \gamma V(s')\right) \\ \Leftrightarrow \bar{R}(s,\pi(s)) &= \left(\phi(s) - \gamma \sum_{s' \in S} p(s'|s,\pi(s))\phi(s')\right)^T \theta^* \end{aligned} \tag{2.55}$$

Pour un état donné s_i, nous réécrivons cette équation sous une forme plus générique :

$$y_i = x_i^T \theta^* \tag{2.56}$$

Si les probabilités de transitions et la fonction de récompense étaient connues, alors y_i serait une sortie observable, x_i une entrée observable et la solution des moindres carrés (que nous

ne détaillons pas, c'est très classique) minimise le coût (2.54) :

$$\theta_i = \left(\sum_{j=1}^{i} x_j x_j^T\right)^{-1} \left(\sum_{j=1}^{i} x_j y_j\right) \tag{2.57}$$

Cependant l'estimation doit se baser sur l'observation de trajectoires, sans connaissance des probabilités de transitions. Les entrées x_i et les sorties y_i ne sont donc pas directement observables, uniquement les transitions (s_i, r_i, s_{i+1}) et les quantités qui y sont liées, comme $\phi(s_i)$ par exemple. Nous réécrivons donc l'équation (2.55) :

$$\begin{aligned} \underbrace{r_i}_{\text{sortie observée}} &= \big(\underbrace{\phi(s_i) - \gamma\phi(s_{i+1})}_{\text{entrée observée}} - \underbrace{\gamma(E_{s'|s_i,\pi(s_i)}[\phi(s')]) - \phi(s_{i+1})}_{\text{bruit sur l'entrée observée}}\big)^T \theta^* \\ &+ \underbrace{r_i - \bar{R}(s_i, \pi(s_i))}_{\text{bruit sur la sortie observée}} \end{aligned} \tag{2.58}$$

Sous cette forme, deux termes de bruit apparaissent. Le terme de bruit sur la sortie n'est pas gênant, il est naturellement pris en compte par les moindres carrés (car il est blanc, comme montré dans [19]). Le fait qu'il y ait un bruit sur l'entrée observée est plus problématique, cette situation est connue sous le nom d'erreur dans les variables [110] (*errors-in-variables* en anglais). La solution des moindres carrés ne peut être directement exprimée en fonction des entrées et sorties observées, elle conduirait à une estimation biaisée du vecteur de paramètres optimal. Une solution à ce problème est l'approche dite des variables instrumentales [110]. Le principe est d'introduire une variable dite instrumentale, corrélée avec l'entrée (différence entre l'entrée observée et le bruit sur cette entrée) et décorrélée d'avec le bruit sur l'entrée observée et de modifier l'estimateur de θ^* en utilisant cette nouvelle variable (voir [110, 19] pour les détails). Pour LSTD, la variable instrumentale choisie est $\phi(s_i)$ (il peut être montré qu'elle vérifie les bonnes propriétés), ce qui mène à l'estimateur suivant :

$$\theta_i = \left(\sum_{j=1}^{i} \phi(s_j)(\phi(s_j) - \gamma\phi(s_{j+1}))^T\right)^{-1} \left(\sum_{j=1}^{i} \phi(s_j) r_j\right) \tag{2.59}$$

Notons que d'autres variables instrumentales pourraient être choisies, le problème du choix optimal de cette variable étant traité dans [115]. Reste à mettre à jour cet estimateur en ligne, c'est-à-dire obtenir une équation sous la forme (2.34). Cela peut être fait grâce au lemme d'inversion matricielle, parfois appelé formule de Sherman-Morrison, que nous donnons ici.

Théorème 2.4 (Lemme d'inversion matricielle). *Soit A une matrice carrée inversible et u et v deux vecteurs. Supposons que $1 + v^T A^{-1} u \neq 0$. Alors :*

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u} \tag{2.60}$$

En appliquant ce lemme d'inversion matricielle à l'estimateur (2.59) il est possible de mettre l'algorithme LSTD sous la forme (2.34) où le gain K_i est défini récursivement par :

$$K_i = \frac{C_{i-1}\phi(s_i)}{1 + (\phi(s_i) - \gamma\phi(s_{i+1}))^T C_{i-1}\phi(s_i)} \tag{2.61}$$

$$C_i = C_{i-1} - \frac{C_{i-1}\phi(s_i)(\phi(s_i) - \gamma\phi(s_{i+1}))^T C_{i-1}}{1 + (\phi(s_i) - \gamma\phi(s_{i+1}))^T C_{i-1}\phi(s_i)} \tag{2.62}$$

Il est à noter que cet algorithme (sous sa forme hors ligne) a été étendu à un schéma d'itération de la politique, l'approche résultante se nommant LSPI [73] (*Least-Squares Policy Iteration*). Il a également été étendu pour prendre en compte les traces d'éligibilité [18]. Une version moins coûteuse en terme de complexité algorithmique a aussi été proposée [50].

Processus gaussiens et différences temporelles

La dernière approche que nous passons en revue est l'algorithme GPTD [28] (*Gaussian Process Temporal Differences*). C'est la plus proche dans les concepts qui la motivent de la contribution que nous introduisons dans le chapitre suivant. L'algorithme GPTD se décline en deux versions principalement, l'une paramétrique et l'autre non-paramétrique. Nous nous focaliserons sur la première, décrite dans [27]. Définissons dans un premier temps ce qu'est un processus gaussien [91].

Définition 2.6 (Processus gaussien). *Un processus aléatoire F est un ensemble de variables aléatoires, à chacune étant assignée un index. Le processus F est gaussien si les variables appartenant à tout sous-ensemble fini de F sont jointement gaussiennes (autrement dit le vecteur dont les composantes sont ces variables aléatoires est gaussien). Si F est indexé par $x \in \mathcal{X}$, sa distribution est entièrement spécifiée par sa moyenne et sa covariance :*

$$E[F(x)] = f(x) \text{ avec } f \in \mathbb{R}^{\mathcal{X}} \tag{2.63}$$

$$\text{cov}(F(x), F(x')) = k(x, x') \text{ où } k \text{ est un noyau de Mercer} \tag{2.64}$$

Un noyau de Mercer est une application symétrique positive (voir définition 8.7 pour plus de détails).

Le principe général de GPTD est de modéliser la fonction de valeur $\hat{V}$ par un processus gaussien, de lier les récompenses aux valeurs via un modèle génératif, puis de chercher la distribution de la fonction de valeur en un état donné conditionnée aux récompenses observées. L'*a priori* gaussien suivant est posé sur la fonction de valeur estimée : $\hat{V} \sim \mathcal{N}(0, k(.,.))$. A partir de maintenant nous supposons les transitions déterministes et la politique fixée (le problème est toujours celui de l'évaluation de la fonction de valeur). L'équation d'évaluation de Bellman peut s'écrire :

$$R(s_i, \pi(s_i), s_{i+1}) = \hat{V}(s_i) - \gamma \hat{V}(s_{i+1}) + N(s_i) \tag{2.65}$$

où N est supposé être un bruit blanc gaussien qui ne dépend pas de la fonction de valeur mais peut dépendre de l'état courant, ce qui est une hypothèse forte. Par abus de notation, la politique étant fixée et les transitions étant déterministes, nous notons $R(s_i)$ en lieu et place de $R(s_i, \pi(s_i), s_{i+1})$. Pour une trajectoire de longueur i les processus aléatoires suivants sont définis :

$$R_i = \begin{pmatrix} R(s_1) & \dots & R(s_i) \end{pmatrix}^T \tag{2.66}$$

$$V_i = \begin{pmatrix} \hat{V}(s_1) & \dots & \hat{V}(s_i) \end{pmatrix}^T \tag{2.67}$$

$$N_i = \begin{pmatrix} N(s_1) & \dots & N(s_i) \end{pmatrix}^T \tag{2.68}$$

Définissons également la matrice de taille $(i-1) \times i$:

$$\mathbf{H}_i = \begin{pmatrix} 1 & -\gamma & 0 & \dots & 0 \\ 0 & 1 & -\gamma & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 1 & -\gamma \end{pmatrix} \tag{2.69}$$

Avec ces notations, il est possible de réécrire l'équation (2.65) pour l'ensemble de la trajectoire :

$$R_{i-1} = \mathbf{H}_i V_i + N_{i-1} \tag{2.70}$$

Cette dernière équation est un modèle génératif, qui lie le processus aléatoire observé (les récompenses) au processus non observable (les valeurs estimées), le premier étant une transformation du second à laquelle est ajouté un bruit aléatoire. Comme annoncé, l'objectif est de déterminer la densité de probabilité de la valeur d'un état s donné conditionnée à la trajectoire, soit $\hat{V}(s)|R_{i-1}$. Mais avant cela nous introduisons d'abord quelques notations supplémentaires :

$$\mathbf{k}_i(s) = \begin{pmatrix} k(s_1, s) & \dots & k(s_i, s) \end{pmatrix}^T \tag{2.71}$$

$$\mathbf{K}_i = \mathrm{Var}(V_i) = (k(s_a, s_b))_{1 \leq a,b \leq i} \tag{2.72}$$

$$\Sigma_i = \mathrm{Var}(N_i) = \mathrm{diag}(\sigma_1^2, \dots, \sigma_i^2) \tag{2.73}$$

Soit un état s donné, l'*a priori* sur la fonction de valeur étant centré et le bruit blanc nous pouvons définir le vecteur gaussien joint suivant :

$$\begin{pmatrix} \hat{V}(s) \\ V_i \\ N_{i-1} \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} k(s,s) & \mathbf{k}_i(s)^T & \mathbf{0}^T \\ \mathbf{k}_i(s) & \mathbf{K}_i & \mathbf{0}^T \\ \mathbf{0} & \mathbf{0} & \Sigma_{i-1} \end{pmatrix} \right) \tag{2.74}$$

Comme nous souhaitons conditionner la valeur aux récompenses, c'est plutôt le vecteur joint des valeurs et des récompenses qui nous intéresse. En utilisant le modèle génératif, I_i étant la matrice identité de taille i, nous avons :

$$\begin{pmatrix} \hat{V}(s) \\ V_i \\ R_{i-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & I_i & 0 \\ 0 & \mathbf{H}_i & I_{i-1} \end{pmatrix} \begin{pmatrix} \hat{V}(s) \\ V_i \\ N_{i-1} \end{pmatrix} \tag{2.75}$$

La transformation linéaire d'un vecteur gaussien étant toujours gaussien et en utilisant cette transformation linéaire, nous avons :

$$\begin{pmatrix} \hat{V}(s) \\ V_i \\ R_{i-1} \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} k(s,s) & \mathbf{k}_i(s)^T & \mathbf{k}_i(s)^T \mathbf{H}_i^T \\ \mathbf{k}_i(s) & \mathbf{K}_i & \mathbf{K}_i \mathbf{H}_i^T \\ \mathbf{H}_i \mathbf{k}_i(s) & \mathbf{H}_i \mathbf{K}_i & \mathbf{H}_i \mathbf{K}_i \mathbf{H}_i^T + \Sigma_{i-1} \end{pmatrix} \right) \tag{2.76}$$

Avant de continuer, nous donnons le théorème de Gauss-Markov, qui permet de déterminer la distribution d'un vecteur aléatoire gaussien conditionnée à un autre connaissant leur distribution jointe.

Théorème 2.5 (Gauss-Markov). *Soit X et Y deux vecteurs aléatoires jointement gaussiens :*

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m_x \\ m_y \end{pmatrix}, \begin{pmatrix} K_{xx} & K_{xy} \\ K_{yy} & K_{yx} \end{pmatrix} \right) \tag{2.77}$$

Alors $X|Y$ est également gaussien de distribution $\mathcal{N}(\hat{X}, P)$ avec :

$$\hat{X} = m_x + K_{xy}K_{yy}^{-1}(Y - m_y) \tag{2.78}$$
$$P = K_{xx} - K_{xy}K_{yy}^{-1}K_{yx} \tag{2.79}$$

En appliquant ce résultat à l'équation (2.76), nous avons donc :

$$\hat{V}(s)|R_{i-1} \sim \mathcal{N}(\hat{v}_i(s), p_i(s)) \tag{2.80}$$
$$\text{avec } \begin{cases} \hat{v}_i(s) = \mathbf{k}_i(s)^T\mathbf{H}_i^T(\mathbf{H}_i\mathbf{K}_i\mathbf{H}_i^T + \Sigma_{i-1})^{-1}R_{i-1} \\ p_i(s) = k(s,s) - \mathbf{k}_i(s)^T\mathbf{H}_i^T(\mathbf{H}_i\mathbf{K}_i\mathbf{H}_i^T + \Sigma_{i-1})^{-1}\mathbf{H}_i\mathbf{k}_i(s) \end{cases}$$

Le problème est que pour l'estimation de la valeur la complexité algorithmique est quadratique en la longueur de la trajectoire. Dans [28, 27], une représentation non-paramétrique et parcimonieuse est développée, mais nous ne la présenterons pas plus avant dans ce chapitre. Dans [27], une représentation paramétrique de la fonction de valeur est introduite. Dans ce cas, comme précédemment, la paramétrisation est linéaire de la forme :

$$\hat{V}_\theta(s) = \sum_{j=1}^{p} w_j\phi_j(s) = \phi(s)^T\theta \tag{2.81}$$

Pour que $\hat{V}_\theta$ soit un processus gaussien, il faut que θ soit un vecteur aléatoire gaussien. Son *a priori* est donc gaussien et avec peu de perte de généralité fixé à :

$$\theta \sim \mathcal{N}(\mathbf{0}, I_p) \tag{2.82}$$

Cela détermine la distribution du processus aléatoire gaussien $\hat{V}_\theta$, qui est totalement décrit par ses deux premiers moments :

$$E[\hat{V}_\theta(s)] = \phi(s)^T E[\theta] = 0 \tag{2.83}$$
$$\text{cov}(\hat{V}_\theta(s), \hat{V}_\theta(s')) = \phi(s)^T E[\theta\theta^T]\phi(s') = \phi(s)^T\phi(s') \tag{2.84}$$

Le choix de la représentation ne modifie pas les résultats de l'équation (2.80). Il suffit de remarquer que $k(s,s') = \phi(s)^T\phi(s')$ et ainsi de suite. De la même façon qu'une distribution *a posteriori* a été déterminée pour $\hat{V}$, une distribution *a posteriori* peut être déterminée pour θ. Nous introduisons d'abord la matrice de taille $p \times i$ définie par :

$$\mathbf{\Phi}_i = \begin{bmatrix} \phi(s_1) & \dots & \phi(s_i) \end{bmatrix} \tag{2.85}$$

Etant donné la représentation particulière de la fonction de valeur, le modèle génératif (2.70) se réécrit :

$$R_{i-1} = \mathbf{H}_i\mathbf{\Phi}_i\theta + N_{i-1} \tag{2.86}$$

En appliquant des résultats classiques des distributions gaussiennes jointes et le théorème de Gauss-Markov, exactement comme avant, nous obtenons une distribution *a posteriori* sur les paramètres :

$$\theta|R_{i-1} \sim \mathcal{N}(\hat{\theta}_i, P_i) \tag{2.87}$$
$$\text{avec } \begin{cases} \hat{\theta}_i = \mathbf{\Phi}_i^T\mathbf{H}_i^T(\mathbf{H}_i\mathbf{\Phi}_i^T\mathbf{\Phi}_i\mathbf{H}_i^T + \Sigma_{i-1})^{-1}R_{i-1} \\ P_i = I_p - \mathbf{\Phi}_i^T\mathbf{H}_i^T(\mathbf{H}_i\mathbf{\Phi}_i^T\mathbf{\Phi}_i\mathbf{H}_i^T + \Sigma_{i-1})^{-1}\mathbf{H}_i\mathbf{\Phi}_i \end{cases}$$

Avec quelques manipulations algébriques et en utilisant le lemme d'inversion matricielle, ce résultat peut se mettre sous la forme de l'équation (2.34), avec le gain K_i défini récursivement par :

$$K_i = \frac{P_{i-1}(\phi(s_i) - \gamma\phi(s_{i+1}))}{\sigma_{i-1}^2 + (\phi(s_i) - \gamma\phi(s_{i+1}))^T P_{i-1}(\phi(s_i) - \gamma\phi(s_{i+1}))} \tag{2.88}$$

$$P_i = P_{i-1} - \frac{P_{i-1}(\phi(s_i) - \gamma\phi(s_{i+1}))(\phi(s_i) - \gamma\phi(s_{i+1}))^T P_{i-1}^T}{\sigma_{i-1}^2 + (\phi(s_i) - \gamma\phi(s_{i+1}))^T P_{i-1}(\phi(s_i) - \gamma\phi(s_{i+1}))} \tag{2.89}$$

Notons que la matrice P_i ne sert pas seulement à définir le gain récursivement, elle a une vraie signification physique : elle quantifie l'incertitude sur les paramètres sachant quelles récompenses ont été observées jusqu'au temps i. Cette forme ressemble à la mise à jour d'un filtre de Kalman [65], nous discuterons ce point plus avant lorsque nous montrerons que l'un des cas particuliers de nos algorithmes correspond au GPTD paramétrique.

2.3.3 Propriétés souhaitables d'un approximateur de fonction de valeur

La contribution principale de cette thèse est une nouvelle classe d'approximateurs de la fonction de valeur et de la fonction de qualité, pour des schémas d'itération de la politique et d'itération de la valeur. Avant de les présenter, nous passons en revue un certain nombre de propriétés que nous pensons importantes :

efficacité en terme d'échantillons : une première propriété intéressante est l'efficacité en terme d'échantillons. Elle consiste en l'apprentissage le plus rapide possible de la fonction de valeur, autrement dit en l'obtention d'une bonne estimation de la valeur avec peu de transitions observées. Cette propriété est très généralement souhaitable, pas seulement dans le cadre de l'apprentissage par renforcement. Dans un contexte industriel, elle peut être très importante. En effet, si l'apprentissage se fait à partir de données issues d'une machine de production, il vaut mieux qu'il nécessite le moins possible de ces données, ces dernières ayant un coût non négligeable ;

traque de la cible : nous pensons important qu'un approximateur de la fonction de valeur puisse prendre en compte la non-stationnarité. Un cas d'application immédiat est un environnement non-stationnaire. Par exemple, pour une machine de production, l'usure des composants peut entraîner une dérive lente de la dynamique du système et cet aspect devrait être pris en compte. D'autres raisons pour préférer traquer la solution plutôt que de chercher à converger vers elle, même si l'environnement est stationnaire, sont données par [106]. Un autre aspect, peut-être plus important encore, est la non-stationnarité induite par l'imbrication du contrôle et de l'apprentissage dont nous avons déjà parlé dans la section 2.2.3. Considérer un algorithme d'apprentissage de la fonction de valeur comme composante d'évaluation d'un schéma d'itération de la politique généralisée entraîne la non-stationnarité de la fonction de valeur cible, étant donné que les phases d'évaluation de la fonction de valeur et d'amélioration de la politique se succèdent. Par exemple, l'algorithme LSTD présenté section 2.3.2 n'est pas robuste à la non-stationnarité de la fonction de valeur cible et c'est pour cela qu'il n'est pas utilisé dans des schémas d'itération optimiste de la politique (amélioration de la politique avant d'avoir fini d'estimer la fonction de valeur, politique ϵ-gloutonne par exemple) ou d'architecture acteur-critique en ligne. Notons que cet aspect est fortement lié au fait que l'apprentissage se fasse en ligne. S'il se fait hors-ligne, il

est possible de générer suffisamment d'expériences pour apprendre la valeur d'une politique donnée et LSTD convient dans ce cas (voir [73] pour un schéma d'itération de la politique et [82] pour un schéma acteur-critique épisodique) ;

apprentissage en ligne : l'apprentissage est dit en ligne si la représentation est mise à jour à chaque nouvelle transition observée, hors-ligne si une trajectoire est nécessaire. L'inconvénient de ce second type d'approche est que si une transition supplémentaire est observée, toute la trajectoire doit être reconsidérée pour calculer la nouvelle représentation. D'un autre côté, un algorithme en ligne peut tout à fait être considéré dans un contexte hors-ligne. De plus, dans le paradigme le plus général de l'apprentissage par renforcement, l'agent apprend le contrôle en interagissant avec le système dynamique, il est donc souhaitable que l'algorithme soit en ligne pour espérer pouvoir garantir la réactivité de l'agent. C'est donc une propriété que nous pensons importante pour un algorithme d'approximation de la fonction de valeur ;

information d'incertitude : nous avons présenté le dilemme entre exploration et exploitation dans la section 2.2.3 et le bénéfice qui pourrait être tiré d'une information d'incertitude. En effet, lorsqu'il doit prendre une décision, l'agent se base sur l'estimation de la valeur de différentes actions, mais il pourrait tirer parti d'une information concernant la confiance qu'il a en ces estimations. Si bénéficier d'une telle information d'incertitude ne résout pas le dilemme entre exploration et exploitation, c'est un outil qui peut s'avérer utile. Dans le cas tabulaire, de nombreuses approches existent pour estimer cette incertitude, comme par exemple [23, 104, 122, 101]. Dans le cas de l'approximation de la fonction de valeur, c'est beaucoup plus rare, l'algorithme GPTD [28] présenté section 2.3.2 est le seul dont nous ayons connaissance ;

prise en compte des non-linéarités : cet aspect a deux facettes. Premièrement, les représentations non-linéaires de la fonction de valeur (réseau de neurones par exemple) permettent généralement des représentations plus compactes que les paramétrisations linéaires. Il serait donc intéressant de pouvoir les prendre en compte. Un autre aspect que nous jugeons important est la capacité à traiter l'équation d'optimalité de Bellman. Cette dernière comprend l'opérateur max, qui est non-linéaire (et même non-dérivable analytiquement) ;

complexité algorithmique : les algorithmes d'approximation de la fonction de valeur sont principalement motivés par la trop grande complexité algorithmique et en mémoire de la représentation tabulaire, dans le cas ou l'espace d'état est trop grand. Il ne faudrait pas qu'un tel algorithme présente une trop grande complexité, surtout s'il a vocation à être en ligne (par exemple, une complexité en le nombre d'état, même si linéaire, serait rapidement rédhibitoire).

Nous avons ainsi présenté un certain nombre de propriétés que nous jugeons importantes pour un approximateur de la fonction de valeur, ce qui peut tenir lieu de cahier des charges pour le cadre de travail général des différences temporelles de Kalman que nous présentons dans le chapitre suivant.

Chapitre 3

Différences temporelles de Kalman

Le problème que nous traitons peut être exprimé comme suit : étant donnée une représentation de la fonction de valeur (ou de qualité) fixée et choisie *a priori* et résumée par un vecteur de paramètres θ, étant donné un schéma de différence temporelle (ou de façon équivalente une équation de Bellman) et étant données les contraintes que nous avons posées dans la section 2.3.3, quel est le meilleur gain K ? Pour répondre à cette question un point de vue statistique est adopté et le cadre de travail du filtrage de Kalman [65] est suivi. En effet, à l'origine le filtrage de Kalman a pour but de traquer l'état caché d'un système dynamique potentiellement non-stationnaire via des observations indirectes de cet état. L'approche présente plusieurs aspects intéressants. Tout d'abord, elle prend naturellement en compte la non-stationnarité, dans la mesure où elle traque la solution plutôt qu'elle ne converge vers elle. Ensuite, l'état caché étant modélisé par une variable aléatoire, une information d'incertitude sur les estimations faites peut être naturellement obtenue. De plus, l'algorithme résultant est du second ordre, donc *a priori* efficace en termes d'échantillons. Enfin, des schémas d'approximation existent pour prendre en compte des non-linéarités. L'idée sous-jacente à l'approche proposée est donc d'exprimer le problème d'approximation de la fonction de valeur comme un problème de filtrage, de façon à bénéficier des avantages de l'approche de Kalman. Les paramètres relatifs à la représentation de la fonction de valeur sont l'état caché à traquer et les récompenses sont les observations indirectes de ces paramètres, via l'une des équations de Bellman et les transitions associées.

Ainsi nous présentons dans ce chapitre les différences temporelles de Kalman (ou KTD pour *Kalman Temporal Differences*), qui est (avec ses variantes) la contribution principale de ce manuscrit. Si le filtrage de Kalman [65, 99] a été utilisé de façon intensive dans le cadre de l'estimation (estimation de l'état d'un système lorsque celui-ci n'est pas directement observable) et du contrôle (par exemple contrôleur linéaire-quadratique-gaussien, pour ne citer que le plus connu), l'approche est différente ici, dans la mesure où ce sont les paramètres de la représentation de la fonction de valeur que l'on cherche à déterminer. Nous nous plaçons donc dans un cadre d'estimation, mais sensiblement différent de ce qui est fait habituellement. Notons tout de même que si ce type d'approche est nouveau en apprentissage par renforcement, le filtrage de Kalman (sous différentes variantes) a déjà été utilisé pour l'estimation de paramètres dans le cadre de l'apprentissage supervisé [88, 89, 92, 33, 32, 116, 109].

Dans un premier temps nous reformulons le problème de l'approximation de la fonction de valeur (ou de qualité) de façon *ad hoc* au point de vue adopté. Cette formulation générique est ensuite utilisée pour dériver la forme la plus générale des différences temporelles de

Kalman. Nous nous intéressons ensuite au cas de l'évaluation de la fonction de valeur (ou de qualité) lorsque la paramétrisation est linéaire. Dans ce cas, les équations de l'algorithme général peuvent être résolues de façon analytique. Nous nous intéressons ensuite au cas non-linéaire (non-linéarité de la paramétrisation, mais aussi opérateur max pour l'équation d'optimalité de Bellman). Pour le traiter, nous présentons d'abord un schéma d'approximation appelé transformation non-parfumée [57] (ou UT pour *unscented transform*). Ce schéma d'approximation est combiné à l'algorithme KTD le plus général pour fournir une famille d'algorithmes pratiques. Nous en discutons également la complexité algorithmique et analysons leur convergence dans un cadre général. Le chapitre se conclut avec une série d'expérimentations sur des problèmes classiques en apprentissage par renforcement, ces tests ayant pour objectif d'illustrer les différents aspects de KTD, à savoir efficacité en termes d'échantillons, robustesse à la non-stationnarité et illustration de l'information d'incertitude disponible.

3.1 Formulation espace-d'état

Une première étape dans la construction du cadre de travail que nous proposons est de reformuler l'approximation de la fonction valeur en adoptant un point de vue statistique, sous la forme d'un problème de filtrage. La quantité que l'on cherche à estimer est l'ensemble des paramètres décrivant la fonction de valeur. Comme nous souhaitons pouvoir disposer d'une information d'incertitude sur ces estimations, nous modélisons les paramètres comme des variables aléatoires. De plus, comme l'on souhaite que les algorithmes aient un caractère adaptatif, c'est-à-dire qu'ils traquent la solution plutôt que de converger vers elle, il est nécessaire de spécifier un modèle d'évolution pour le vecteur de paramètres. Sans autre connaissance *a priori*, nous suivons le principe du rasoir d'Occam et adoptons un modèle de marche aléatoire. Les paramètres forment donc le processus caché que l'on souhaite estimer. Pour ce faire, il faut pouvoir l'observer, même indirectement. Dans notre cas, il est observé via les récompenses obtenues lors de l'interaction entre l'agent et son environnement. Il faut également spécifier un modèle d'observation, liant les paramètres aux récompenses. Nous avons déjà mentionné dans le chapitre 2 que la différence des valeurs en deux états consécutifs d'une transition pouvait servir de prédiction de la récompense associée. C'est ce modèle d'observation que nous adoptons. Même si la fonction de récompense est déterministe, ce processus d'observation est aléatoire, d'une part car il ne fait pas intervenir les probabilités de transition (et l'équation de Bellman ne peut être formellement vérifiée), d'autre part parce que le choix de la paramétrisation ne permet pas nécessairement une représentation exacte de la fonction de valeur d'intérêt. Ainsi, le problème d'approximation de la fonction de valeur est reformulé en deux équations, l'une décrivant le processus d'évolution des paramètres et l'autre le processus d'observation liant ces derniers aux récompenses. Nous présentons maintenant tout cela plus formellement.

Un point de vue très général est adopté. Pour cela un certain nombre de notations sont définies. Ainsi, une transition sera notée :

$$t_i = \begin{cases} (s_i, s_{i+1}) \\ (s_i, a_i, s_{i+1}, a_{i+1}) \\ (s_i, a_i, s_{i+1}) \end{cases} \tag{3.1}$$

selon que l'objectif est l'évaluation de la fonction de valeur, de la Q-fonction ou l'optimisation de Q (c'est-à-dire la détermination directe de la fonction de qualité optimale). De

façon similaire et pour les mêmes cas nous introduisons la fonction g définie par :

$$g_{t_i}(\theta_i) = \begin{cases} \hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) \\ \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \hat{Q}_{\theta_i}(s_{i+1}, a_{i+1}) \\ \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \max_{b \in A} \hat{Q}_{\theta_i}(s_{i+1}, b) \end{cases} \tag{3.2}$$

Ainsi tous les schémas de différences temporelles peuvent s'écrire génériquement par :

$$\delta_i = r_i - g_{t_i}(\theta_i) \tag{3.3}$$

Avec ces notations, $g_{t_i}(\theta_i)$ peut être vu comme la prédiction de la récompense au temps i en accord avec la représentation θ_i et δ_i peut être vu comme un terme d'innovation qui quantifie l'information gagnée en observant la nouvelle récompense r_i. Ces notions seront raffinées plus tard.

Comme annoncé, un point de vue statistique est adopté. L'approximation de la fonction de valeur est exprimée sous une forme probabiliste inspirée du paradigme du filtrage et le vecteur de paramètres θ est modélisé comme étant une variable aléatoire suivant une marche aléatoire, ce processus étant générateur des récompenses. Le problème peut ainsi s'exprimer dans un forme dite *espace-d'état*[1] :

$$\begin{cases} \theta_i = \theta_{i-1} + v_i & \text{(équation d'évolution)} \\ r_i = g_{t_i}(\theta_i) + n_i & \text{(équation d'observation)} \end{cases} \tag{3.4}$$

Cette expression est fondamentale pour le paradigme proposé. La première équation est l'équation dite d'évolution (parfois appelée équation de process ou encore équation d'état), elle spécifie que le vecteur de paramètres suit une marche aléatoire dont la moyenne correspond à l'estimation optimale. Le bruit d'évolution v_i est centré, blanc, indépendant et de variance P_{v_i}. Cette variance dépend du problème traité et doit être spécifiée par le praticien (nous discuterons ce choix dans la partie expérimentale). Notons d'une part que cette équation n'est pas une mise à jour des paramètres (qui sera abordée plus tard) mais leur évolution naturelle au cours du temps et d'autre part que cette formulation permet de prendre en compte la non-stationnarité éventuelle de la fonction de valeur. La seconde équation est l'équation dite d'observation, elle lie la transition observée ainsi que la récompense associée à la fonction de valeur (ou de qualité) au travers d'une des équations de Bellman. Le bruit d'observation n_i est supposé blanc (hypothèse primordiale à l'obtention des équations de Kalman, comme nous le verrons), centré, indépendant et de variance P_{n_i}. Cette variance dépend également du problème, elle doit aussi être spécifiée par le praticien (nous la discuterons dans la partie expérimentale). Ce modèle de bruit a deux causes. D'une part, l'équation d'observation est écrite pour une transition donnée, elle ne transcrit pas exactement l'équation de Bellman, le moyennage selon les probabilités de transition étant ignoré. Il n'y a donc pas de raison pour qu'il y ait égalité stricte entre r_i et $g_{t_i}(\theta_i)$, à moins éventuellement que les transitions soient déterministes, ce qui peut être modélisé par un bruit (qui contient donc la stochasticité du MDP). D'autre part, la solution de l'équation

[1] La dénomination espace-d'état (*state-space* en anglais) vient du filtrage de Kalman et de l'automatique (représentation d'état) et n'a pas de lien avec la notion d'état d'un MDP. Originellement, le filtrage de Kalman a pour but d'inférer l'état caché d'un système à partir d'observations. Dans le cadre de travail proposé, cet état caché est le vecteur de paramètres et les observations sont les récompenses et transitions, à ne pas confondre avec l'état du système. Nous utilisons donc le même terme pour les deux notions, le contexte permettant facilement de les différencier.

de Bellman considérée n'appartient pas nécessairement à l'espace fonctionnel engendré par l'ensemble des paramètres (la structure d'approximation étant fixée *a priori*). Cette incapacité potentielle de l'approximateur à représenter de façon exacte la fonction d'intérêt est connue sous le terme de biais inductif en apprentissage supervisé (à ne pas confondre avec le biais des algorithmes résiduels par exemple). Nous le modélisons également à l'aide de ce bruit.

Ainsi, dans le modèle proposé, les récompenses sont générées par le processus aléatoire sur les paramètres, via une des équations de Bellman. Formellement, si le problème est l'évaluation de la fonction de valeur à politique fixée dans un environnement stationnaire, le bruit d'évolution est nul. Cependant, même dans ce cas, son introduction peut-être bénéfique. Premièrement, si son amplitude n'est pas trop forte, il n'empêche pas l'estimation correcte de la valeur (dans le cadre des équations de Kalman que nous introduirons par la suite), cela parce qu'il est choisi centré. Deuxièmement, cela permet de prendre en compte les non-stationnarités éventuelles dont nous avons déjà discuté. Enfin, toujours dans le cadre des équations de Kalman à venir, il permet une certaine robustesse aux optima locaux, son effet pouvant être compris par analogie avec les techniques de recuit simulé. Si l'estimation du vecteur de paramètres atteint un minimum local, le bruit d'évolution aura pour effet de le sortir de la cuvette dans laquelle il est bloqué. D'autres raisons pour préférer une traque à une convergence, même dans un cas purement stationnaire, sont évoquées par [106]. Le problème est de savoir comment estimer les paramètres de la fonction de valeur et nous l'abordons maintenant.

3.2 Formulation générale de l'algorithme

Dans cette section nous introduisons la forme la plus générale de l'algorithme KTD, qui servira de base à tous les algorithmes pratiques que nous présentons dans la suite de ce chapitre. Cet algorithme est obtenu par la minimisation d'une fonction de coût que nous présentons dans un premier temps. La minimisation effective de ce coût est ensuite abordée, ce qui permet d'obtenir les équations (de Kalman) qui définissent l'algorithme KTD.

3.2.1 Coût minimisé

L'objectif pourrait être d'estimer le vecteur de paramètres qui minimise l'espérance de l'erreur quadratique conditionnée aux récompenses observées depuis l'origine des temps. Le coût associé s'écrit :

$$J_i(\theta) = E\left[\|\theta_i - \theta\|^2 | r_{1:i}\right] \text{ avec } r_{1:i} = r_1, \ldots, r_i \tag{3.5}$$

De façon générale, l'estimateur minimisant l'erreur quadratique moyenne est l'espérance conditionnelle :

$$\underset{\theta}{\operatorname{argmin}}\, J_i(\theta) = \hat{\theta}_{i|i} = E\left[\theta_i | r_{1:i}\right] \tag{3.6}$$

Cependant, à part pour des cas spécifiques (notamment le cas où les équations d'évolution et d'observation sont linéaires et les bruits gaussiens), cet estimateur ne peut pas être calculé analytiquement. Une solution approchée est de trouver le meilleur estimateur *linéaire* (qui est optimal dans le cas linéaire gaussien). Il peut être écrit sous une forme similaire à l'équation (2.34) :

$$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i \tilde{r}_i \tag{3.7}$$

Dans l'équation (3.7), $\hat{\theta}_{i|i}$ est l'estimation au temps i, $\hat{\theta}_{i|i-1} = E[\theta_i|r_{1:i-1}]$ est la prédiction de cette estimation en accord avec les récompenses observées dans le passé $r_{1:i-1}$ et l'innovation

$$\tilde{r}_i = r_i - \hat{r}_{i|i-1} \tag{3.8}$$

est la différence entre la récompense observée r_i et sa prédiction $\hat{r}_{i|i-1} = E[r_i|r_{1:i-1}]$ basée sur les récompenses observées dans le passé $r_{1:i-1}$ (nous rappelons que cette espérance concerne le caractère aléatoire des paramètres).

Pour le modèle de marche aléatoire adopté, le bruit d'évolution étant blanc et centré, nous avons :

$$\begin{aligned}\hat{\theta}_{i|i-1} &= E[\theta_i|r_{1:i-1}] \\ &= E[\theta_{i-1} + v_i|r_{1:i-1}] \text{ (équation d'évolution)} \\ &= E[\theta_{i-1}|r_{1:i-1}] \text{ (bruit blanc centré)} \\ &= \hat{\theta}_{i-1|i-1}\end{aligned} \tag{3.9}$$

Un raisonnement similaire peut être mené pour la prédiction de la récompense :

$$\begin{aligned}\hat{r}_{i|i-1} &= E[r_i|r_{1:i-1}] \\ &= E[g_{t_i}(\theta_i) + n_i|r_{1:i-1}] \text{ (équation d'observation)} \\ &= E[g_{t_i}(\theta_i)|r_{1:i-1}] \text{ (bruit blanc centré)}\end{aligned} \tag{3.10}$$

Notons également que l'innovation[2] (3.8) n'est pas exactement l'erreur de différence temporelle définie dans l'équation (3.3), qui est une variable aléatoire en conséquence de sa dépendance au vecteur aléatoire θ_i : c'est son espérance conditionnée aux récompenses précédemment observées. Etant donnée la mise à jour postulée (3.7), il s'agit de déterminer le gain K_i qui permette la minimisation du coût (3.5).

3.2.2 Gain optimal

En utilisant des égalités classiques[3], la fonction de coût peut se réécrire de la façon suivante (l'opérateur trace associant à une matrice carrée la somme de ses éléments diagonaux) :

$$\begin{aligned}J_i(\theta) &= E\left[\|\theta_i - \theta\|^2|r_{1:i}\right] \\ &= E\left[(\theta_i - \theta)^T(\theta_i - \theta)|r_{1:i}\right] \\ &= \text{trace}\left(E\left[(\theta_i - \theta)(\theta_i - \theta)^T|r_{1:i}\right]\right) \\ &= \text{trace}\left(\text{cov}\left(\theta_i - \theta|r_{1:i}\right)\right)\end{aligned} \tag{3.11}$$

Une première étape pour calculer le gain optimal est d'exprimer la covariance de l'erreur sur les paramètres conditionnées aux récompenses comme une fonction du gain K_i. Mais

[2]Le terme innovation vient du domaine du traitement du signal et est usuel dans le cadre du filtrage de Kalman.

[3]En effet, pour un vecteur $x = (x_1, \ldots, x_n)^T$, $\|x\|^2 = x^T x = \sum_{i=1}^n x_i^2$ et xx^T est une matrice symétrique de terme générale $(x_i x_j)_{1\leq i,j\leq n}$ dont la trace vaut donc $\sum_{i=1}^n x_i^2$.

d'abord quelques notations supplémentaires sont introduites (rappelons également la définition de l'innovation (3.8)) :

$$\begin{cases} \tilde{\theta}_{i|i} = \theta_i - \hat{\theta}_{i|i} & \text{et} \quad \tilde{\theta}_{i|i-1} = \theta_i - \hat{\theta}_{i|i-1} \\ P_{i|i} = \text{cov}\left(\tilde{\theta}_{i|i}|r_{1:i}\right) & \text{et} \quad P_{i|i-1} = \text{cov}\left(\tilde{\theta}_{i|i-1}|r_{1:i-1}\right) \\ P_{r_i} = \text{cov}\left(\tilde{r}_i|r_{i|i-1}\right) & \text{et} \quad P_{\theta r_i} = E\left[\tilde{\theta}_{i|i-1}\tilde{r}_i|r_{1|i-1}\right] \end{cases} \tag{3.12}$$

En utilisant la mise à jour postulée (3.7), les différents estimateurs étant non biaisés, la covariance peut être développée :

$$\begin{aligned} P_{i|i} &= \text{cov}\left(\theta_i - \hat{\theta}_{i|i}|r_{1:i}\right) \\ &= \text{cov}\left(\theta_i - \left(\hat{\theta}_{i|i-1} + K_i\tilde{r}_i\right)|r_{1:i-1}\right) \\ &= \text{cov}\left(\tilde{\theta}_{i|i-1} - K_i\tilde{r}_i|r_{1:i-1}\right) \\ &= P_{i|i-1} - P_{\theta r_i}K_i^T - K_iP_{\theta r_i}^T + K_iP_{r_i}K_i^T \end{aligned} \tag{3.13}$$

Le gain optimal peut ainsi être obtenu en minimisant la trace de la matrice $P_{i|i}$, covariance de l'erreur sur les paramètres conditionnées aux récompenses. Cela peut être fait en annulant son gradient.

Notons tout d'abord que le gradient étant linéaire, pour trois matrices de dimensions *ad hoc* A, B et C, B étant symétrique, nous avons les identités algébriques suivantes :

$$\nabla_A\left(\text{trace}\left(ABA^T\right)\right) = 2AB \tag{3.14}$$

$$\nabla_A\left(\text{trace}\left(AC^T\right)\right) = \nabla_A\left(\text{trace}\left(CA^T\right)\right) = C \tag{3.15}$$

et donc en utilisant l'équation (3.13) et les identités précédentes nous avons :

$$\begin{aligned} &\nabla_{K_i}\left(\text{trace}\left(P_{i|i}\right)\right) = 0 \\ &\Leftrightarrow K_i = P_{\theta r_i}P_{r_i}^{-1} \end{aligned} \tag{3.16}$$

En injectant le gain optimal (3.16) dans l'expression de la matrice de covariance de l'erreur conditionnée aux récompenses observées (3.13), nous en obtenons une expression simplifiée :

$$P_{i|i} = P_{i|i-1} - K_iP_{r_i}K_i^T \tag{3.17}$$

Il est à noter qu'aucune hypothèse gaussienne n'a été faite pour obtenir ces résultats (hypothèse que nous utiliserons tout de même pour l'analyse de convergence proposée section 3.6). Dans le cas linéaire, la mise à jour (3.7) que nous contraignons à la linéarité l'est vraiment.

3.2.3 Algorithme

L'algorithme le plus général de différences temporelles de Kalman, qui se subdivise en trois parties, peut maintenant être obtenu. Au temps i, la transition t_i et la récompense r_i sont observées et les estimations $\hat{\theta}_{i-1|i-1}$ et $P_{i-1|i-1}$ des moments d'ordres un et deux du vecteur aléatoire des paramètres sont connus. Notons que cela sous-entend de choisir un *a priori* sur les moments initiaux d'ordre 1 et 2 des paramètres, c'est-à-dire $\hat{\theta}_{0|0}$ et $P_{0|0}$. Il est ainsi possible d'injecter de la connaissance *a priori*, si disponible. La première étape, dite

de *prédiction*, consiste à calculer les prédictions $\hat{\theta}_{i|i-1}$ et $P_{i|i-1}$ qui sont nécessaires pour les mises à jour (3.7) et (3.17). Rappelons que pour un modèle de marche aléatoire la prédiction du vecteur de paramètres est égale à son estimation précédente (3.9) : $\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$. La covariance prédite peut également être calculée analytiquement en utilisant l'équation d'évolution et l'indépendance du bruit d'évolution :

$$\begin{aligned} P_{i|i-1} &= \text{cov}\left(\tilde{\theta}_{i|i-1}|r_{1:i-1}\right) \\ &= \text{cov}\left(\tilde{\theta}_{i-1|i-1} + v_i|r_{1:i-1}\right) \\ &= P_{i-1|i-1} + P_{v_i} \end{aligned} \tag{3.18}$$

Cette phase de prédiction consiste donc à ajouter de l'incertitude aux paramètres, cette incertitude étant quantifiée par la matrice de variance du bruit d'évolution.

La seconde étape consiste à calculer quelques statistiques d'intérêt. C'est principalement cette partie qui sera spécialisée par la suite. La première statistique à calculer est la prédiction de la récompense $\hat{r}_{i|i-1}$ (3.10). La seconde est la covariance entre l'erreur sur les paramètres et l'innovation, que l'on peut simplifier étant donnée la forme de l'équation d'observation et le bruit d'observation étant blanc et indépendant :

$$\begin{aligned} P_{\theta r_i} &= E\left[(\theta_i - \hat{\theta}_{i|i-1})(r_i - \hat{r}_{i|i-1})|r_{1:i-1}\right] \\ &= E\left[(\theta_i - \hat{\theta}_{i|i-1})(g_{t_i}(\theta_i) + n_i - \hat{r}_{i|i-1})|r_{1:i-1}\right] \\ &= E\left[(\theta_i - \hat{\theta}_{i|i-1})(g_{t_i}(\theta_i) - \hat{r}_{i|i-1})|r_{1:i-1}\right] \end{aligned} \tag{3.19}$$

Enfin, la dernière statistique à calculer est la variance de l'innovation, qui peut être écrite (en utilisant à nouveau les caractéristiques du bruit d'observation) :

$$\begin{aligned} P_{r_i} &= E\left[(r_i - \hat{r}_{i|i-1})^2|r_{1:i-1}\right] \\ &= E\left[(g_{t_i}(\theta_i) - \hat{r}_{i|i-1} + n_i)^2|r_{1:i-1}\right] \\ &= E\left[(g_{t_i}(\theta_i) - \hat{r}_{i|i-1})^2|r_{1:i-1}\right] + P_{n_i} \end{aligned} \tag{3.20}$$

La dernière étape de l'algorithme est la phase de *correction* qui consiste à calculer le gain (3.16), ce qui est aisé si les statistiques d'intérêt sont connues, puis à mettre à jour le vecteur de paramètres (3.7) ainsi que la matrice de covariance associée (3.17). L'approche générale proposée est résumée dans l'algorithme 3.1.

KTD est un algorithme du second ordre, dans la mesure où il maintient une information de moyenne et de variance sur les paramètres. De la même façon que pour les méthodes de gradient des approches de type Newton-Raphson, qui utilisent les dérivées d'ordre un et deux, sont plus efficaces (bien que plus coûteux) que les approches du type descente de gradient, cela laisse espérer que KTD (ou plutôt ses implantations pratiques) soit efficace en termes d'échantillons, c'est-à-dire qu'il traque rapidement la fonction de valeur ou de qualité. Cela sera expérimenté dans la section 3.8.

3.3 Cas linéaire

Nous allons dans un premier temps nous intéresser au cas linéaire : d'une part la paramétrisation est linéaire, d'autre part nous nous restreignons aux équations d'évaluation

Algorithme 3.1 : KTD général

Initialisation : *a priori* $\hat{\theta}_{0|0}$ et $P_{0|0}$;

pour $i \leftarrow 1, 2, \ldots$ **faire**

Observer la transition t_i ainsi que la récompense associée r_i ;

Phase de prédiction;

$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;

$P_{i|i-1} = P_{i-1|i-1} + P_{v_i}$;

Calcul des statistiques d'intérêt;

$\hat{r}_{i|i-1} = E[g_{t_i}(\theta_i)|r_{1:i-1}]$;

$P_{\theta r_i} = E\left[(\theta_i - \hat{\theta}_{i|i-1})(g_{t_i}(\theta_i) - \hat{r}_{i|i-1})|r_{1:i-1}\right]$;

$P_{r_i} = E\left[(g_{t_i}(\theta_i) - \hat{r}_{i|i-1})^2|r_{1:i-1}\right] + P_{n_i}$;

Phase de correction;

$K_i = P_{\theta r_i} P_{r_i}^{-1}$;

$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i\left(r_i - \hat{r}_{i|i-1}\right)$;

$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

de Bellman, l'équation d'optimalité ne pouvant être considérée à cause de l'opérateur max (qui rend l'équation d'observation non-linéaire). Dans ce cas bien particulier, il est possible de déterminer de façon analytique les équations de KTD (qui sont en fait les équations de Kalman, sous une forme très générale). Comme les problèmes d'estimation de la valeur et de la qualité sont très proches, nous nous restreindrons à l'évaluation de la fonction de valeur, l'extension à l'évaluation de la Q-fonction étant évidente.

Dans cette section la paramétrisation linéaire de l'équation (2.35) est adoptée, que l'on rappelle :

$$\hat{V}_\theta(s) = \phi(s)^T\theta \tag{3.21}$$

La formulation espace d'état (3.4) peut donc se réécrire :

$$\begin{cases} \theta_i = \theta_{i-1} + v_i \\ r_i = \left(\phi(s_i) - \gamma\phi(s_{i+1})\right)^T \theta_i + n_i \end{cases} \tag{3.22}$$

Pour raccourcir les équations, nous définissons le vecteur H_i :

$$H_i = \phi(s_i) - \gamma\phi(s_{i+1}) \tag{3.23}$$

Comme l'équation d'observation est linéaire, les statistiques d'intérêt peuvent être calculées analytiquement. La prédiction de la récompense est donnée par :

$$\begin{aligned} \hat{r}_{i|i-1} &= E\left[g_{t_i}(\theta_i)|r_{1:i-1}\right] \\ &= E\left[H_i^T\theta_i|r_{1:i-1}\right] \\ &= H_i^T E\left[\theta_i|r_{1:i-1}\right] \\ &= H_i^T\hat{\theta}_{i|i-1} \end{aligned} \tag{3.24}$$

La covariance entre l'erreur sur les paramètres et l'innovation peut également être calculée analytiquement :

$$\begin{aligned} P_{\theta r_i} &= E\left[\tilde{\theta}_{i|i-1}\left(g_{t_i}(\theta_i) - \hat{r}_{i|i-1}\right)|r_{1:i-1}\right] \\ &= E\left[\tilde{\theta}_{i|i-1}H_i^T\tilde{\theta}_{i|i-1}|r_{1:i-1}\right] \\ &= E\left[\tilde{\theta}_{i|i-1}\tilde{\theta}_{i|i-1}^T H_i|r_{1:i-1}\right] \\ &= E\left[\tilde{\theta}_{i|i-1}\tilde{\theta}_{i|i-1}^T|r_{1:i-1}\right]H_i \\ &= P_{i|i-1}H_i \end{aligned} \tag{3.25}$$

Enfin, la variance de l'innovation est également calculable :

$$\begin{aligned} P_{r_i} &= E\left[\left(g_{t_i}(\theta_i) - \hat{r}_{i|i-1}\right)^2|r_{1:i-1}\right] + P_{n_i} \\ &= E\left[\left(H_i^T\tilde{\theta}_{i|i-1}\right)^2|r_{1:i-1}\right] + P_{n_i} \\ &= E\left[\left(H_i^T\tilde{\theta}_{i|i-1}\right)\left(\tilde{\theta}_{i|i-1}^T H_i\right)|r_{1:i-1}\right] + P_{n_i} \\ &= H_i^T P_{i|i-1}H_i + P_{n_i} \end{aligned} \tag{3.26}$$

Le gain peut ainsi être défini de façon algébrique et récursive (la récurrence apparaissant sur la matrice de variance $P_{i|i-1}$) :

$$K_i = \frac{P_{i|i-1}H_i}{H_i^T P_{i|i-1}H_i + P_{n_i}} \tag{3.27}$$

Ce gain partage des similarités avec le gain de l'algorithme LSTD de [19], donné équation (2.61), ce qui n'est pas une surprise. En effet, LSTD est basé sur une minimisation par moindres carrés (cependant avec l'introduction du concept de variable instrumentale pour prendre en compte le cas des transitions stochastiques) et le filtre de Kalman peut être vu comme une généralisation de cette approche (sans variables instrumentales). En imposant un bruit d'évolution nul (*id est* de variance nulle), le gain obtenu est exactement celui de l'algorithme GPTD [28], présenté section 2.3.2 et dont le gain est donné équation (2.88). Les deux points de vue adoptés pour GPTD et KTD sont relativement différents, cependant ce recouvrement des algorithmes n'est pas vraiment une surprise. Dans le cas linéaire gaussien, le filtrage de Kalman est la solution optimale au problème du filtrage bayésien et la façon dont sont utilisés les processus gaussiens dans [28] est une forme d'inférence bayésienne. Nous discutons plus avant les points communs et différences de ces deux approches dans la section 3.7. La spécialisation de KTD pour l'évaluation de la fonction de valeur linéairement paramétrée, que nous nommons KTD-V avec paramétrisation linéaire, est résumée dans l'algorithme 3.2. L'extension à l'évaluation de la fonction de qualité est simple.

Nous avons ainsi présenté un algorithme KTD pratique dans le cas d'une paramétrisation linéaire, pour l'évaluation de la fonction de valeur. Dans le cas d'une paramétrisation non-linéaire et/ou pour l'équation d'optimalité de Bellman, les statistiques d'intérêt ne peuvent plus être calculées analytiquement. Un schéma d'approximation est donc nécessaire. De plus, il faudrait que ce dernier ne nécessite pas la mise en œuvre de gradients,

Algorithme 3.2 : KTD-V : paramétrisation linéaire

Initialisation : *a priori* $\hat{\theta}_{0|0}$ et $P_{0|0}$;

pour $i \leftarrow 1, 2, \ldots$ **faire**

Observer la transition (s_i, s_{i+1}) ainsi que la récompense associée r_i ;

Phase de prédiction;

$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;

$P_{i|i-1} = P_{i-1|i-1} + P_{v_i}$;

Calcul des statistiques d'intérêt;

$\hat{r}_{i|i-1} = H_i^T \hat{\theta}_{i|i-1}$;

$P_{\theta r_i} = P_{i|i-1} H_i$;

$P_{r_i} = H_i^T P_{i|i-1} H_i + P_{n_i}$;

/* où $H_i = \phi(s_i) - \gamma\phi(s_{i+1})$ */

Phase de correction;

$K_i = P_{\theta r_i} P_{r_i}^{-1}$;

$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i \left(r_i - \hat{r}_{i|i-1}\right)$;

$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

car sinon il ne permettrait pas de prendre en compte l'opérateur max. Basiquement, ce problème du calcul des statistiques d'intérêt peut être exprimé de la façon suivante : étant donné la moyenne et la covariance d'une variable aléatoire, comment calculer la moyenne et la covariance d'une transformation non-linéaire (voir éventuellement non dérivable) de cette variable aléatoire ? Nous présentons dans la prochaine section la transformation non-parfumée [59, 58, 57], qui est un schéma d'approximation permettant de faire face à ce genre de problème, avant de l'utiliser pour obtenir de nouveaux algorithmes.

3.4 La transformation non-parfumée

Laissons pour l'instant de côté l'apprentissage par renforcement et le filtrage de Kalman. Soit X un vecteur aléatoire et Y une fonction de X. Le problème posé est de calculer la moyenne et la variance de Y en connaissant celles de X ainsi que la fonctionnelle les liant. Si cette dernière est linéaire, la relation entre X et Y peut s'écrire $Y = AX$ où A est une matrice de dimension *ad hoc*. Dans ce cas, moyenne et covariance peuvent être calculées analytiquement :

$$E[Y] = AE[X] \text{ et } E[YY^T] = AE[XX^T]A^T \tag{3.28}$$

Ce résultat a été utilisé pour obtenir l'algorithme KTD-V de la section 3.3.

Si la transformation est non-linéaire, elle peut génériquement se mettre sous la forme suivante :

$$Y = f(X) \tag{3.29}$$

Une première solution est d'approximer la fonctionnelle même, c'est-à-dire de la linéariser autour de la moyenne du vecteur aléatoire X. Cela mène aux approximations suivantes pour la moyenne et la covariance de Y :

$$E[Y] \approx f\left(E[X]\right) \text{ et } E[YY^T] \approx \left(\nabla f\left(E[X]\right)\right) E[XX^T] \left(\nabla f\left(E[X]\right)\right)^T \tag{3.30}$$

Ceci est la base du filtrage de Kalman étendu (EKF pour *Extended Kalman Filtering*, voir [99] par exemple), qui a été intensivement étudié et utilisé dans les décennies passées. Cependant, cette approche présente quelques sévères limitations. Premièrement, elle ne permet pas de prendre en compte des non-linéarités non dérivables et ne peut donc pas prendre en compte l'équation d'optimalité de Bellman à cause de l'opérateur max. Cette approche nécessite également d'évaluer le gradient de f, ce qui peut être compliqué (par exemple si f représente un réseau de neurones, il faut rétropropager le gradient). Enfin, cela suppose que f est localement linéarisable, ce qui n'est malheureusement pas toujours le cas et peut mener à de mauvaises approximations, comme illustré par [57].

L'idée de base de la transformation non-parfumée est qu'il est plus judicieux d'approximer un vecteur aléatoire arbitraire qu'une fonction non-linéaire arbitraire. Son principe est d'échantillonner de façon *déterministe* un ensemble de "sigma-points", définis ci-après, à partir de la moyenne et de la covariance de X. Les images de ces sigma-points par l'application f sont ensuite calculées et elles sont utilisées pour calculer les statistiques d'intérêt. Ce schéma d'approximation ressemble aux méthodes de Monte-Carlo. Cependant l'échantillonnage est ici déterministe, nécessite la génération de moins d'échantillons, ne nécessite pas de connaître la loi de X, ce tout en permettant une précision donnée [57].

Nous décrivons à présent l'une des premières formes de transformation non-parfumée. D'autres variantes ont été introduites depuis, mais le principe de base est le même. Soit n la dimension de X. Un ensemble de $2n+1$ sigma-points et poids associés est calculé comme suit :

$$\begin{cases} x^{(0)} = \bar{X} & w_0 = \frac{\kappa}{n+\kappa}, \quad j = 0 \\ x^{(j)} = \bar{X} + \left(\sqrt{(n+\kappa)P_X}\right)_j & w_j = \frac{1}{2(n+\kappa)}, \quad 1 \leq j \leq n \\ x^{(j)} = \bar{X} - \left(\sqrt{(n+\kappa)P_X}\right)_{n-j} & w_j = \frac{1}{2(n+\kappa)}, \quad n+1 \leq j \leq 2n \end{cases} \tag{3.31}$$

où $\bar{X}$ est la moyenne de X, P_X est sa matrice de variance, κ est un coefficient d'échelle permettant de contrôler la précision de la transformation non-parfumée [57] et $(\sqrt{(n+\kappa)P_X})_j$ est la $j^{\text{ème}}$ colonne de la décomposition de Cholesky de la matrice $(n+\kappa)P_X$. L'image de chacun de ces points par f est ensuite calculée : $y^{(j)} = f(x^{(j)})$, $0 \leq j \leq 2n$. L'ensemble des sigma-points et de leurs images peut alors être utilisé pour estimer les moments d'ordre un et deux de Y et même P_{XY}, la covariance entre X et Y :

$$\bar{Y} \approx \bar{y} = \sum_{j=0}^{2n} w_j y^{(j)} \tag{3.32}$$

$$P_Y \approx \sum_{j=0}^{2n} w_j \left(y^{(j)} - \bar{y}\right)\left(y^{(j)} - \bar{y}\right)^T \tag{3.33}$$

$$P_{XY} \approx \sum_{j=0}^{2n} w_j \left(x^{(j)} - \bar{X}\right)\left(y^{(j)} - \bar{y}\right)^T \tag{3.34}$$

Nous illustrons la transformation non-parfumée sur la figure 3.1, adaptée de [116]. Sur la partie gauche est représentée la transformation non-linéaire d'une variable aléatoire gaussienne, ainsi que les moyennes et covariances exactes. Les distributions sont représentées par un ensemble de points échantillonnés pour X et par leurs images par la transformation non-linéaire pour Y. Sur la figure du milieu sont illustrées les approximations de la moyenne et de la covariance obtenues en linéarisant la fonctionnelle. Sur la figure de droite

c'est l'approximation obtenue en utilisant la transformation non-parfumée qui est montrée, ainsi que les sigma-points et leurs images.

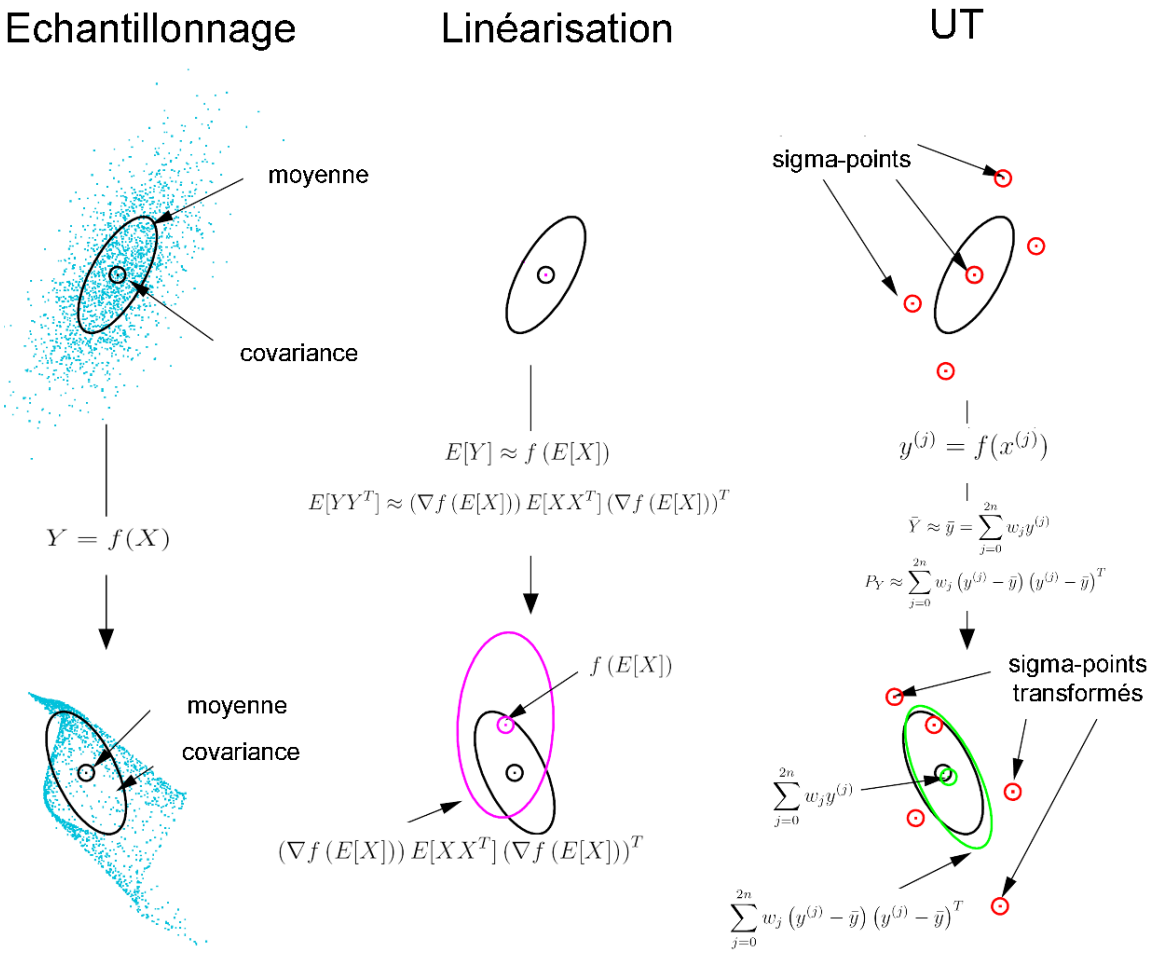


FIG. 3.1 – Illustration de la transformation non-parfumée.

Il est à noter que si la transformation est linéaire, la transformation non-parfumée n'est plus une approximation et les moments calculés sont exacts. Cette transformation non-parfumée ayant été présentée, nous allons l'intégrer à KTD pour obtenir une nouvelle famille d'algorithmes.

3.5 Spécialisations

Dans cette section nous spécialisons l'algorithme général de KTD, selon que l'on considère la fonction de valeur ou de qualité, l'équation d'évaluation ou d'optimisation de Bellman, cela quelle que soit la paramétrisation. Notons que, comme dans le cas d'une transformation linéaire, la transformation non-parfumée n'est plus une approximation (c'est-à-dire que les moments ne sont plus approchés, ils sont exacts), les résultats et algorithmes que nous présentons restent valable pour le cas présenté section 3.3, ils en sont une autre représentation.

3.5.1 KTD-V

Dans cette section nous nous intéressons à l'évaluation de la fonction de valeur pour une paramétrisation générique $\hat{V}_\theta$: cela peut être un réseau de neurones [15], une représentation par noyau semi-paramétrique [38], ou toute autre représentation d'intérêt, du moment qu'elle puisse être décrite par un ensemble fini de p paramètres. La formulation générale espace d'état de l'équation (3.4) s'écrit dans ce cas :

$$\begin{cases} \theta_i = \theta_{i-1} + v_i \\ r_i = \hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) + n_i \end{cases} \tag{3.35}$$

Le problème est toujours de calculer les statistiques d'intérêt, ce qui est fait ici en utilisant la transformation non-parfumée. Au temps i, les estimations $\hat{\theta}_{i-1|i-1}$ et $P_{i-1|i-1}$ sont connues et la transition $t_i = (s_i, s_{i+1})$ ainsi que la récompense associée r_i sont observées. Les estimations sont utilisées pour calculer les prédictions $\hat{\theta}_{i|i-1}$ et $P_{i|i-1}$. L'ensemble des sigma-points est calculé à partir de ces prédictions, ainsi que les poids associés, comme décrit dans la section 3.4 :

$$\Theta_{i|i-1} = \left\{ \hat{\theta}^{(j)}_{i|i-1},\ 0 \leq j \leq 2p \right\} \tag{3.36}$$

$$\mathcal{W} = \{ w_j,\ 0 \leq j \leq 2p \} \tag{3.37}$$

Ensuite les images de ces sigma-points sont calculées en utilisant l'équation d'observation du modèle espace d'état (3.35), qui est lié à l'équation d'évaluation de Bellman pour la fonction de valeur :

$$\mathcal{R}_{i|i-1} = \left\{ \hat{r}^{(j)}_{i|i-1} = \hat{V}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i) - \gamma \hat{V}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}),\ 0 \leq j \leq 2p \right\} \tag{3.38}$$

Enfin, les sigma-points et leurs images étant calculés, les statistiques d'intérêt peuvent être approchées par :

$$\hat{r}_{i|i-1} \approx \sum_{j=0}^{2p} w_j \hat{r}^{(j)}_{i|i-1} \tag{3.39}$$

$$P_{r_i} \approx \sum_{j=0}^{2p} w_j \left(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1} \right)^2 + P_{n_i} \tag{3.40}$$

$$P_{\theta r_i} \approx \sum_{j=0}^{2p} w_j \left(\hat{\theta}^{(j)}_{i|i-1} - \hat{\theta}_{i|i-1} \right) \left(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1} \right) \tag{3.41}$$

L'algorithme résultant est équivalent à l'algorithme 3.2 dans le cas d'une paramétrisation linéaire. L'algorithme 3.3 résume l'approche KTD-V générale.

3.5.2 KTD-SARSA

Cette section traite le problème de l'évaluation de la fonction de qualité pour une politique donnée. L'algorithme associé est appelé KTD-SARSA, ce qui peut induire en erreur. En effet, de façon générale, le terme SARSA est souvent associé à l'évaluation de la Q-fonction dans le cadre d'une itération optimiste de la politique (politique ϵ-gloutonne

Algorithme 3.3 : KTD-V, KTD-SARSA et KTD-Q

Initialisation : *a priori* $\hat{\theta}_{0|0}$ et $P_{0|0}$;

pour $i \leftarrow 1, 2, \ldots$ **faire**

Observer la transition $t_i = \begin{cases} (s_i, s_{i+1}) \text{ (KTD-V)} \\ (s_i, a_i, s_{i+1}, a_{i+1}) \text{ (KTD-SARSA)} \\ (s_i, a_i, s_{i+1}) \text{ (KTD-Q)} \end{cases}$ ainsi que la récompense r_i ;

Phase de prédiction;

$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;

$P_{i|i-1} = P_{i-1|i-1} + P_{v_i}$;

Calcul des sigma-points ;

$\Theta_{i|i-1} = \left\{ \hat{\theta}^{(j)}_{i|i-1}, \quad 0 \leq j \leq 2p \right\}$ (en utilisant $\hat{\theta}_{i|i-1}$ et $P_{i|i-1}$);

$\mathcal{W} = \{ w_j, \quad 0 \leq j \leq 2p \quad \}$;

$\mathcal{R}_{i|i-1} =$

$$\begin{cases} \left\{ \hat{r}^{(j)}_{i|i-1} = \hat{V}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i) - \gamma \hat{V}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}),\ 0 \leq j \leq 2p \right\} \text{ (KTD-V)} \\ \left\{ \hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i) - \gamma \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, a_{i+1}),\ 0 \leq j \leq 2p \right\} \text{ (KTD-SARSA)} \\ \left\{ \hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i) - \gamma \max_{b \in A} \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, b),\ 0 \leq j \leq 2p \right\} \text{ (KTD-Q)} \end{cases} ;$$

Calcul des statistiques d'intérêt;

$\hat{r}_{i|i-1} = \sum_{j=0}^{2p} w_j \hat{r}^{(j)}_{i|i-1}$;

$P_{\theta r_i} = \sum_{j=0}^{2p} w_j (\hat{\theta}^{(j)}_{i|i-1} - \hat{\theta}_{i|i-1})(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})$;

$P_{r_i} = \sum_{j=0}^{2p} w_j \left(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1} \right)^2 + P_{n_i}$;

Phase de correction;

$K_i = P_{\theta r_i} P_{r_i}^{-1}$;

$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i \left(r_i - \hat{r}_{i|i-1} \right)$;

$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

notamment). Ici nous nous focalisons sur l'aspect évaluation de la fonction de qualité, en laissant le contrôle de côté (ce problème étant tout de même considéré par la suite). Pour une paramétrisation générale $\hat{Q}_\theta$, en considérant l'équation d'évaluation de Bellman pour la fonction de qualité, le modèle espace d'état (3.4) s'écrit :

$$\begin{cases} \theta_i = \theta_{i-1} + v_i \\ r_i = \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \hat{Q}_{\theta_i}(s_{i+1}, a_{i+1}) + n_i \end{cases} \tag{3.42}$$

Pour une politique fixée, l'évaluation de la Q-fonction sur la chaîne de Markov valuée induite par l'espace état-action est similaire au problème de l'évaluation de la fonction de valeur sur la chaîne de Markov valuée induite par l'espace d'état. Il est alors assez évident d'étendre KTD-V à l'évaluation de la Q-fonction. Au temps i une transition $t_i = (s_i, a_i, s_{i+1}, a_{i+1})$ est observée ainsi que la récompense associée r_i. Comme précédemment les prédictions, calculées à partir des estimations précédentes, sont utilisées pour déterminer les sigma-points ainsi que les poids associés. Les images de ces sigma-points sont ensuite calculées en utilisant l'équation d'observation :

$$\mathcal{R}_{i|i-1} = \left\{ \hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i) - \gamma \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, a_{i+1}),\ 0 \leq j \leq 2p \right\} \tag{3.43}$$

Les sigma-points et leurs images servent ensuite à approcher les statistiques d'intérêt, comme pour KTD-V, les équations (3.39-3.41) restant valables. KTD-SARSA est aussi résumé dans l'algorithme 3.3.

3.5.3 KTD-Q

Cette section s'intéresse à l'optimisation directe de la Q-fonction, c'est-à-dire à trouver une solution approchée de l'équation d'optimalité de Bellman. Une paramétrisation générale $\hat{Q}_\theta$ est adoptée. Le modèle espace d'état (3.4) est alors spécialisé comme suit :

$$\begin{cases} \theta_i = \theta_{i-1} + v_i \\ r_i = \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \max_{b \in A} \hat{Q}_{\theta_i}(s_{i+1}, b) + n_i \end{cases} \tag{3.44}$$

Ici la distinction entre paramétrisation linéaire ou non n'a plus lieu d'être. En effet l'opérateur max, inhérent à l'équation d'optimalité de Bellman, rend l'équation d'observation de (3.44) non-linéaire. Cet opérateur est difficile à prendre en compte, particulièrement à cause de sa non dérivabilité.

Heureusement, comme elle approxime le vecteur aléatoire plutôt que la fonction, la transformation non-parfumée ne requiert pas de dérivation. Etant donnés l'algorithme KTD général présenté section 3.2.3 et la transformation non-parfumée décrite dans la section 3.4, il est possible d'obtenir KTD-Q, la spécialisation de KTD pour l'optimisation directe de la Q-fonction. Notons qu'une telle spécialisation n'est *a priori* pas possible pour LSTD [19] ou GPTD [28], l'hypothèse de linéarité étant fortement ancrée dans ces travaux. Même pour les algorithmes résiduels [6], l'optimisation directe de la fonction de qualité est toute théorique, en raisons de la nécessité de calculer le gradient de l'opérateur max. Pour revenir à KTD-Q, une première étape est de calculer les sigma-points associés au vecteur aléatoire de paramètres, comme dans les équations (3.36-3.37), après avoir observé la transition $t_i = (s_i, a_i, s_{i+1})$. Ensuite, l'image de ces sigma-points à travers l'équation d'observation du

modèle espace d'état (3.44), qui contient l'opérateur max, est calculée :

$$\mathcal{R}_{i|i-1} = \left\{ \hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i) - \gamma \max_{b \in A} \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, b),\ 0 \leq j \leq 2p \right\} \tag{3.45}$$

Enfin, les sigma-points et leurs images sont utilisés pour calculer les statistiques d'intérêt, comme dans les équations (3.39-3.41). Autant que nous le sachions, KTD-Q est le premier algorithme du second ordre qui soit du type itération de la valeur (c'est-à-dire qui cherche directement à résoudre l'équation d'optimalité de Bellman). L'approche KTD-Q proposée est aussi résumée dans l'algorithme 3.3.

3.5.4 Illustration des algorithmes

Dans cette section, nous nous proposons d'illustrer d'une part l'utilisation de la transformation non-parfumée permettant d'obtenir la prédiction des récompenses ainsi que les statistiques d'intérêt à partir des moments d'ordre 1 et 2 du vecteur aléatoire de paramètres, d'autre part le principe général de KTD à l'aide d'un schéma bloc.

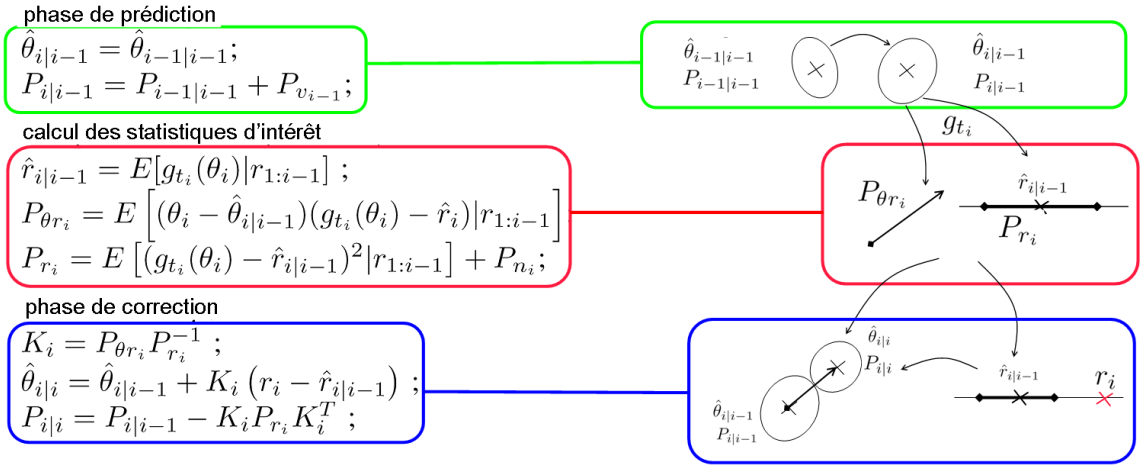


FIG. 3.2 – Schéma bloc de l'algorithme KTD.

La figure 3.2 illustre sous forme de schéma bloc une itération de KTD. Au temps i, les estimations obtenues au temps précédent sont disponibles et une transition est observée. La première étape est la phase de prédiction. Nous représentons un vecteur de paramètres à 2 dimensions et cette phase consiste à modifier la variance des paramètres en fonction du bruit d'évolution (les paramètres moyens ne sont pas modifiés, nous le rappelons). La seconde étape consiste à calculer les statistiques d'intérêt à partir des moments d'ordre 1 et 2 de la prédiction des paramètres et de la fonction g_{t_i}, qui dépend de l'algorithme considéré et de la transition observée. En pratique, ces statistiques sont déterminées en utilisant la transformation non-parfumée, comme illustrée figure 3.3. Nous y reviendrons. La prédiction de la récompenses $\hat{r}_{i|i-1}$ et la variance sur l'innovation P_{r_i} sont toujours des scalaires. Dans le cas illustré la corrélation entre les paramètres et les récompenses $P_{\theta r_i}$ est un vecteur à 2 dimensions. La dernière étape est la phase de correction, qui consiste à corriger les paramètres moyens dans la direction donnée par le gain de Kalman (à l'amplitude près, la direction donnée par $P_{\theta r_i}$), proportionnellement à l'erreur faite entre la récompense observée

et la récompense prédite (cette erreur étant une forme d'erreur de différence temporelle). La matrice de variance est corrigée de façon similaire. Ceci clôt la mise à jour effectuée par KTD.

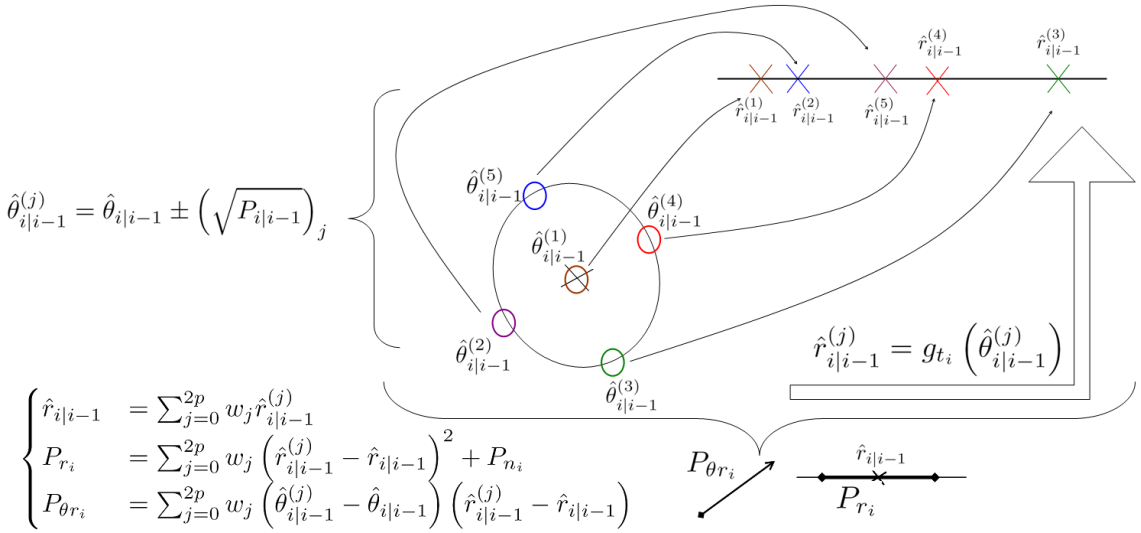


FIG. 3.3 – Utilisation de la transformation non-parfumée dans KTD.

Nous illustrons plus particulièrement sur la figure 3.3 l'utilisation de la transformation non parfumée pour obtenir les statistiques d'intérêt. Nous représentons encore les paramètres par un vecteur de dimension 2. La décomposition de Cholesky est utilisée pour calculer l'ensemble des sigma-points (5 dans ce cas) à partir des prédictions des moments d'ordre 1 et 2 des paramètres. Chaque sigma-point correspond à une paramétrisation particulière de la fonction de valeur (ou de qualité). L'image de chacun de ces vecteurs de paramètres par g_{t_i} est ensuite calculée, chaque image étant la prédiction d'une récompense basée sur une représentation particulière de la fonction de valeur. Enfin, les sigma-points et leurs images sont utilisés pour calculer les statistiques d'intérêt, à savoir le vecteur $P_{\theta r_i}$ et les scalaires P_{r_i} et $\hat{r}_{i|i-1}$.

3.5.5 Complexité des algorithmes

Rappelons que p désigne le nombre de paramètres. La transformation non-parfumée implique de calculer une décomposition de Cholesky qui a une complexité computationnelle en $O(p^3)$. Cependant, pour les algorithmes considérés, la structure de mise à jour particulière de la matrice de covariance $P_{i|i-1}$, qui est de rang un, permet de considérer un algorithme spécifique de mise à jour de la décomposition de Cholesky dont la complexité est en $O(p^2)$. Les détails de cette approche "racine carrée" sont donnés dans [119, 116]. Les différents algorithmes impliquent d'évaluer $2p+1$ fois la fonction g_{t_i} à chaque pas de temps. Pour KTD-V et KTD-SARSA et une paramétrisation générale, chaque évaluation est en $O(p)$. Pour KTD-Q, le maximum selon les actions doit être calculé. Notons $\mathcal{A}$ le nombre d'actions si l'espace correspondant est fini, la complexité calculatoire de l'algorithme utilisé pour trouver ce maximum sinon (par exemple le nombre d'échantillons tirés pour Monte-Carlo). Ainsi chaque évaluation est bornée par $O(p\mathcal{A})$. Le reste des opérations

est de l'algèbre linéaire basique, de complexité au plus $O(p^2)$. Ainsi, la complexité computationnelle (par itération) de KTD-V et KTD-SARSA est $O(p^2)$, celle de KTD-Q est $O(\mathcal{A}p^2)$. L'ensemble des algorithmes requiert de stocker le vecteur de paramètres ainsi que la matrice de covariance associée, leur complexité en mémoire est donc $O(p^2)$. Notons que dans le cas de l'évaluation de la fonction de valeur ou de qualité, pour une paramétrisation linéaire (voir l'algorithme 3.2), seules des opérations algébriques sont impliquées, la complexité tant calculatoire qu'en mémoire est également en $O(p^2)$. Dans ce cas particulier, il n'y a donc pas de raison *a priori* de préférer une formulation à l'autre. Cette complexité est plus grande que celle d'algorithmes d'apprentissage par renforcement qui sont linéaires en le nombre de paramètres, comme les algorithmes directs ou résiduels, et équivalente à celle d'algorithmes quadratiques comme LSTD ou encore GPTD. Cependant, KTD ne fait pas d'hypothèse de la linéarité de la représentation de la fonction de valeur, ce qui permet de considérer des approximateurs beaucoup plus compacts (mais non-linéaires). De plus, ce sont des algorithmes du second ordre, donc efficaces en termes d'échantillons. Ainsi la complexité quadratique peut être un problème, mais elle a des contreparties importantes.

3.6 Analyse de convergence

Nous proposons ici une analyse de convergence de l'algorithme KTD (c'est-à-dire indistinctement KTD-V, KTD-SARSA et KTD-Q avec approximation des statistiques d'intérêt par la transformation non-parfumée). Elle conduit à un résultat similaire à celui des algorithmes résiduels [6] présentés section 2.3.2, c'est-à-dire la minimisation d'un résidu de Bellman quadratique.

Théorème 3.1 (Convergence de KTD). *Supposons que les distributions a posteriori et des bruits soit gaussiennes et que l'a priori soit également gaussien, de moyenne θ_0 et de variance P_0. Alors l'estimateur de KTD minimise un résidu quadratique empirique régularisé pondéré de Bellman :*

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmin}} \left(\sum_{j=1}^{i} \frac{1}{P_{n_j}} \left(r_j - g_{t_j}(\theta) \right)^2 + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \right) \tag{3.46}$$

Démonstration. La première chose à noter est que KTD est en fait une forme spécifique de filtre de Kalman à sigma-points [117] (SPKF pour *Sigma-Point Kalman Filter*). D'après [116, chapitre 4.5.], sous les hypothèses de distributions *a posteriori* et des bruits gaussiennes, l'estimateur d'un SPKF (et donc celui de KTD) est l'estimateur du maximum *a posteriori* (MAP) :

$$\hat{\theta}_{i|i} = \hat{\theta}_i^{\mathrm{MAP}} = \underset{\theta}{\operatorname{argmax}}\, p(\theta | r_{1:i}) \tag{3.47}$$

En appliquant la règle de Bayes, la distribution *a posteriori* $p(\theta|r_{1:i})$ peut être exprimée comme le produit (normalisé) de la vraisemblance $p(r_{1:i}|\theta)$ par la distribution *a priori* $p(\theta)$:

$$p(\theta | r_{1:i}) = \frac{p(r_{1:i}|\theta) p(\theta)}{p(r_{1:i})} \tag{3.48}$$

Le facteur de normalisation $p(r_{1:i})$ ne dépend pas des paramètres, le maximum *a posteriori* se réduit donc au produit de la vraisemblance et de l'*a priori* :

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmax}}\, p(r_{1:i}|\theta) p(\theta) \tag{3.49}$$

Nous rappelons que dans le modèle de KTD le bruit d'observation est supposé blanc, donc la vraisemblance jointe est le produit des vraisemblances locales :

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmax}}\, p(r_{1:i}|\theta)p(\theta) = \underset{\theta}{\operatorname{argmax}}\, \Pi_{j=1}^{i} p(r_j|\theta)p(\theta) \tag{3.50}$$

Comme de plus le bruit et l'*a priori* sont supposés gaussiens, nous avons :

$$r_j|\theta \sim \mathcal{N}\left(g_{t_j}(\theta), P_{n_j}\right) \text{ et } \theta \sim \mathcal{N}\left(\theta_0, P_0\right) \tag{3.51}$$

D'autre part, maximiser un produit de densités de probabilités est équivalent à minimiser la somme des opposés de leurs logarithmes :

$$\hat{\theta}_{i|i} = -\underset{\theta}{\operatorname{argmin}} \left(\sum_{j=1}^{i} \ln(p(r_j|\theta)) + \ln(p(\theta)) \right) \tag{3.52}$$

Etant donné que les distributions concernées sont gaussiennes, elles ont pour distributions :

$$p(r_j|\theta) = \frac{1}{\sqrt{2\pi P_{n_j}}} \exp\left(-\frac{1}{2} \frac{(r_j - g_{t_j}(\theta))^2}{P_{n_j}} \right) \tag{3.53}$$

$$\text{et } p(\theta) = \frac{1}{(2\pi)^{\frac{p}{2}} |P_0|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \left(\theta - \theta_0\right)^T P_0^{-1} \left(\theta - \theta_0\right) \right) \tag{3.54}$$

Nous avons donc :

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmin}} \left(\sum_{j=1}^{i} \frac{1}{P_{n_j}} \left(r_j - g_{t_j}(\theta)\right)^2 + \left(\theta - \theta_0\right)^T P_0^{-1} \left(\theta - \theta_0\right) \right) \tag{3.55}$$

Ceci prouve le résultat. □

Plusieurs remarques d'importance concernant ce résultat peuvent être mentionnées. Premièrement, la forme de la fonction coût minimisée (3.46) permet d'appréhender intuitivement un problème que vont poser les transitions stochastiques. En effet, ce coût est proche[4] de celui des algorithmes résiduels (bien que la façon de le minimiser soit assez fondamentalement différente) et nous avons vu dans la section 2.3.2 que cette fonction coût utilisée dans un contexte stochastique est biaisée, comme le montre l'équation (2.52). On peut donc s'attendre à ce que KTD fournisse également un estimateur biaisé dans le cas des transitions stochastiques, ce qui sera montré expérimentalement dans la section 3.8. Nous montrons d'ailleurs dans le chapitre 4 que c'est l'hypothèse que le bruit d'observation est blanc, valable dans le cas de transitions déterministes, qui est alors trop forte et nous y proposerons une solution à ce problème. Deuxièmement, la variance du bruit d'observation choisi P_{n_i} permet de pondérer les échantillons. La variance du bruit d'évolution n'apparaît pas directement dans la fonction coût minimisée, cependant elle influence (de façon empirique) la convergence de l'algorithme ainsi que sa capacité à traquer les non-stationnarités. Par exemple, elle peut aider à éviter les minima locaux, son effet pouvant être compris comme une forme de recuit simulé. Troisièmement, ce résultat montre que l'*a priori* se traduit par un terme de régularisation, classique en apprentissage supervisé. Ceci peut fournir

[4] Au terme de régularisation près, les coûts sont identiques si la variance du bruit d'observation est choisie constante.

des pistes intéressantes et intuitives quant au choix de l'initialisation de l'algorithme. Enfin, il peut être montré (voir [116, chapitre 4.5.] à nouveau) qu'une mise à jour de SPKF (et donc de KTD) est en fait une forme en ligne d'une méthode modifiée de Gauss-Newton, qui est en fait une variante de descente de gradient naturel. Dans ce cas, la matrice d'information de Fisher est l'inverse de la matrice de variance des paramètres, c'est-à-dire $P_{i|i}^{-1}$. Les approches de gradient naturel ont été prouvées efficaces pour la recherche directe de politiques [63] ainsi que pour des architectures acteur-critique [82], ce qui laisse espérer de bons résultats expérimentaux pour KTD. Cela est vérifié dans la section 3.8. Autant que nous le sachions, KTD est le premier algorithme en apprentissage par renforcement qui traite l'approximation de la fonction de valeur (ou de qualité), dans un sens critique pur, qui implique un gradient naturel.

3.7 Travaux similaires et discussion

Comme nous avons pu le remarquer précédemment, des approches apparentées ont été proposées précédemment dans la littérature. Nous avons déjà dit que l'algorithme proposé partage des similarités avec LSTD [19], présenté section 2.3.2. De façon générale, le filtrage de Kalman peut être vu comme une généralisation des approches de type moindres carrés. En effet, si le modèle espace d'état est linéaire, le bruit d'évolution nul et le bruit d'observation de variance unitaire, Kalman fournit la solution des moindres carrés. Pour les MDP déterministes (pour lesquels l'hypothèse de bruit blanc est vérifiée), KTD peut donc être vu comme une généralisation de LSTD selon plusieurs axes : prise en compte de la non-stationnarité (bruit d'évolution), liberté sur la pondération des erreurs instantanées (bruit d'observation) et non-linéarité (tant pour la paramétrisation que pour la prise en compte de l'opérateur d'optimalité de Bellman). Cependant KTD ne prend pas en compte le concept de variable instrumentale [110] (leur inclusion dans le contexte du filtrage de Kalman est loin d'être évidente), ces remarques ne sont valables que pour des environnements dont la dynamique est déterministe. Nous avons également déjà vu que l'algorithme proposé partage des similarités avec l'algorithme GPTD [28], aussi présenté section 2.3.2. Dans le cas d'une paramétrisation linéaire et d'un bruit de process nul, KTD-V est le même algorithme que GPTD. En fait, les deux cadres de travail, KTD et GPTD [27] (et variantes), bien que dérivés de points de vue sensiblement différents (modèle génératif de la récompense par les valeurs et processus gaussiens pour GPTD, minimisation de l'espérance de l'erreur quadratique sur les paramètres conditionnée aux observations pour KTD), présentent de nombreux points communs (nous verrons dans la suite du manuscrit que d'autres variantes de KTD sont équivalentes à d'autres variantes de GPTD). Basiquement, les différences principales entre ces deux approches sont les suivantes : GPTD permet une représentation non-paramétrique de la fonction de valeur, mais ne permet pas de prendre en compte un bruit d'évolution et ne peut considérer que des modèles linéaires (pas de possibilité *a priori* d'avoir un équivalent à KTD-Q notamment), alors que KTD prend en compte les non-linéarités et peut bénéficier de la définition d'un bruit d'évolution, mais est restreint aux représentations paramétriques de la fonction de valeur. Notons tout de même que cet aspect non-paramétrique de GPTD peut être adapté à KTD, comme nous le montrons dans la section 8.3.2. Dans [21] est proposée une forme de filtre de Kalman conçu pour approcher le point fixe de l'opérateur de Bellman (ou de façon équivalent approcher la solution de l'équation de Bellman) dans le cas d'une paramétrisation linéaire de la valeur. Cette approche peut approximativement être vue comme une version "bootstrappée" de KTD-V

(le terme *bootstrap* est utilisé ici avec le même sens que [105], c'est-à-dire considérer que la valeur de l'état vers lequel transite le système est connue et utiliser une valeur approchée à la place, comme expliqué section 2.3.2 pour les algorithmes dits directs). A la place de l'équation d'observation (3.22), l'équation d'observation suivante est utilisée :

$$r_i + \gamma\phi(s_{i+1})^T\hat{\theta}_{i-1|i-1} = \phi(s_i)^T\theta_i + n_i \tag{3.56}$$

Autrement dit, la récompense n'est pas considérée comme l'observation, mais une approximation de la fonction de valeur est utilisée pour calculer une pseudo-observation (ou observation "bootstrappée") et la mise à jour du filtre est faite de façon à corriger l'erreur entre la fonction de valeur de l'état courant et la pseudo-observation. Dans [84] est utilisé un banc de filtres de Kalman pour apprendre les paramètres d'une représentation linéaire par morceaux de la fonction de valeur (un filtre de Kalman par morceau linéaire). Cette méthode peut grossièrement être vue comme un cas particulier de KTD, cependant des différences existent : un banc de filtres est utilisé plutôt qu'un seul (ce qui revient à imposer une décorrélation des paramètres correspondants à chaque morceau linéaire) et la paramétrisation est linéaire, fait exploité pour développer des spécificités de l'algorithme (notamment concernant la mise à jour des paramètres).

Le cadre de travail proposé présente les aspects que nous avons souhaités. Premièrement, il ne suppose pas la stationnarité, grâce au bruit d'évolution. Il traque la solution plutôt qu'il ne converge vers elle. Une application immédiate est la prise en compte d'environnements non-stationnaires. D'autres raisons pour préférer traquer la solution, même dans un environnement stationnaire, sont données par [106]. Un autre aspect intéressant est le cas du contrôle, comme discuté dans la section 2.2.3. Par exemple, l'algorithme LSTD est connu pour mal se comporter lorsqu'il est combiné avec un schéma d'itération de la politique optimiste (politique ϵ-gloutonne par exemple, voir [84]), à cause des non-stationnarités induites par ce schéma spécifique d'apprentissage et de contrôle. Dans la même idée, TD est préféré à LSTD dans [13] comme critique dans l'approche acteur-critique qui y est proposée, pour le même problème, alors que TD est moins efficace en terme d'échantillons. Le filtrage de Kalman et donc les algorithmes proposés sont robustes aux problèmes de non-stationnarité. Nous proposons d'ailleurs dans le chapitre 7 un algorithme acteur-critique basé sur KTD. Deuxièmement, comme le vecteur de paramètres est modélisé par une variable aléatoire, il est possible de calculer une information d'incertitude sur la valeur des états, comme nous le décrirons dans le chapitre 6. Cette information d'incertitude pourrait être utile pour traiter le problème général du dilemme entre exploration et exploitation et nous proposons un certain nombre d'approches dans le chapitre 6. Les algorithmes de la famille KTD sont du second ordre et peuvent être compris comme des formes de descente de gradient naturel, ce qui devrait leur assurer une bonne vitesse de convergence. Remarquons qu'une difficulté éventuelle peut être le choix des différents paramètres. Cependant, d'une part la nécessité de choisir certains paramètres n'est pas propre à notre approche (taux d'apprentissage pour TD, *a priori* pour LSTD, *et caetera*) et pas forcément plus compliqué dans notre cas, d'autre part KTD peut grandement bénéficier de la vaste littérature sur le filtrage adaptatif, domaine étudié depuis plusieurs décennies. Nous en utilisons d'ailleurs une forme dans la section expérimentale. L'ensemble de ces aspects est illustré dans la section suivante.

3.8 Expérimentations

Dans cette section est proposé un ensemble de tests de référence classiques en apprentissage par renforcement de façon à comparer KTD à différents algorithmes de l'état de l'art et à illustrer les différents aspects de cette approche. Nous en donnons d'abord un aperçu :

chaîne de Boyan : c'est une simple chaîne de Markov valuée sur laquelle l'objectif est d'apprendre la fonction de valeur. Nous illustrons dans un premier temps le biais intuité par le résultat du théorème 3.1 sur la chaîne originale, dont les transitions sont stochastiques. Nous la modifions ensuite pour la rendre déterministe et non-stationnaire, de façon à illustrer cet aspect de KTD et le comparer à l'état de l'art ;

pendule inversé : ce problème, qui consiste à maintenir un pendule inversé dans une position verticale en appliquant des forces sur la chariot dans lequel il repose, vise à illustrer l'algorithme KTD-Q, qui y apprend une politique quasi-optimale en observant une politique totalement aléatoire ;

***mountain car* :** cette tâche, qui consiste à sortir une voiture sous-puissante d'une cuvette, vise à illustrer KTD-SARSA combiné à un schéma d'itération optimiste de la politique, nommément une politique ϵ-gloutonne, pour montrer que KTD est robuste aux non-stationnarités induites par la dualité entre apprentissage et contrôle.

Les autres algorithmes considérés sont TD, SARSA, Q-learning et LSTD (et ponctuellement GPTD). Nous considérons peu leurs extensions aux traces d'éligibilité, dans la mesure où LSTD produit de meilleurs résultats que TD(λ) et où varier la valeur du facteur d'éligibilité a peu d'incidence sur les performances de LSTD(λ), comme noté par [18]. Cependant, les traces d'éligibilité peuvent tout de même jouer sur le comportement non-asymptotique et nous proposons une comparaison de KTD à LSTD(λ) sur la chaîne de Boyan.

3.8.1 Chaîne de Boyan

La première expérimentation est la chaîne Boyan [18], illustrée figure 3.4. Le but est d'une part d'illustrer le biais intuité par le résultat du théorème 3.1 (les raisons de ce biais et sa forme propre à KTD seront étudiées plus avant dans le prochain chapitre) et d'autre part de montrer l'efficacité de KTD en terme d'échantillons et sa capacité à prendre en compte un environnement non-stationnaire sur une version déterministe du problème.

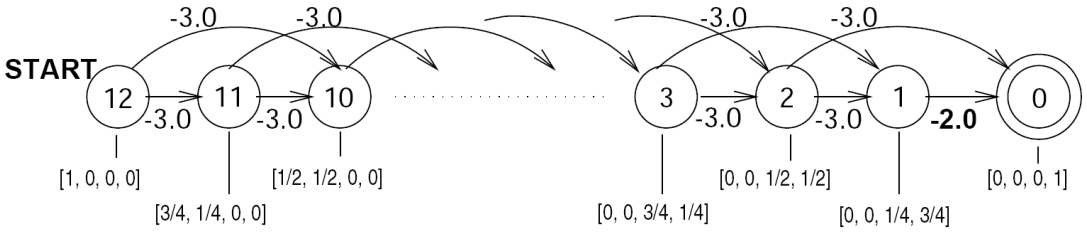


FIG. 3.4 – Chaîne de Boyan illustrée [17].

Cas stochastique

La chaîne de Boyan est une chaîne de Markov valuée à 13 états dont l'état s^0 est absorbant, s^1 transite vers s^0 avec une probabilité de 1 et une récompense de -2 et s^i transite vers s^{i-1} ou s^{i-2}, $2 \leq i \leq 12$, pour chacun avec une probabilité de 0.5 et une récompense de -3. Pour cette expérience, KTD-V est comparé à TD ainsi qu'à LSTD. La paramétrisation est linéaire et les vecteurs de base $\phi(s)$ pour les états s^{12}, s^8, s^4 et s^0 sont respectivement $[1,0,0,0]^T$, $[0,1,0,0]^T$, $[0,0,1,0]^T$ et $[0,0,0,1]^T$. Pour les autres états, les vecteurs de base sont obtenus par interpolation linéaire. L'approximation de la fonction de valeur est donc $\hat{V}_\theta(s) = \theta^T\phi(s)$. La fonction de valeur optimale est linéaire en ces bases et le vecteur de paramètres optimal correspondant est $\theta^* = [-24,-16,-8,0]^T$. La performance est mesurée avec la distance euclidienne $\|\theta - \theta^*\|$ entre les vecteurs de paramètres courant et optimal.

Le coefficient d'actualisation γ est fixé à 1 pour cette tâche épisodique. Pour TD, le taux d'apprentissage est fixé à $\alpha_i = 0.1$. Pour LSTD et KTD-V l'*a priori* est fixé à $P_{0|0} = I$, où I est la matrice identité. Choisir le même *a priori* pour LSTD et KTD est équitable, étant donné le lien qu'il existe entre les deux approches et que nous venons de discuter. Pour KTD-V, le bruit d'observation est fixé à $P_{n_i} = 10^{-3}$ et le bruit de process à $P_{v_i} = 0I$. Choisir ces paramètres requiert un peu de pratique, mais pas forcément plus que le choix d'un taux d'apprentissage pour d'autres algorithmes. La variance du bruit d'observation P_{n_i} traduit la confiance que l'on a en la capacité de la paramétrisation à représenter la fonction de valeur recherchée. Dans ce cas, nous savons que la fonction de valeur est linéaire en les bases choisies, nous choisissons donc une variance faible. Si la variance est trop forte, les mises à jour auront une faible amplitude et la convergence sera lente. Si elle est trop faible, ces mêmes amplitudes seront trop fortes, notamment si la fonction de valeur ne peut pas être parfaitement représentée par la paramétrisation choisie et l'apprentissage sera instable. Le choix de la variance du bruit d'évolution va conditionner la capacité d'adaptation de l'algorithme, autrement dit sa faculté à traquer la solution plutôt qu'à converger vers elle. S'il est choisi trop important, le processus d'adaptation sera très rapide, mais l'apprentissage aura tendance à être instable. Si ce bruit est trop faible, l'adaptation sera lente. Ici nous évaluons une politique fixée, il n'y a pas de raison à traquer la solution, nous choisissons donc une variance nulle. Pour toutes les méthodes considérées, le vecteur de paramètres est initialisé à zéro. Les résultats sont présentés figure 3.5.

LSTD converge plus rapidement que TD, comme attendu, KTD-V converge encore plus vite. Cependant ce dernier algorithme ne converge pas vers le vecteur de paramètres optimal, ce qui s'explique par le fait que la fonction de coût minimisée est biaisée. Cette expérience a été menée pour montrer le problème causé par les transitions stochastiques et nous nous concentrons à présent sur des MDP déterministes.

Cas déterministe et non-stationnaire

La chaîne de Boyan est rendue déterministe en posant la probabilité de transiter de s^i à s^{i-1} à 1. KTD-V est à nouveau comparé à LSTD et TD. De plus, pour simuler un changement dans le MDP et donc la non-stationnarité, le signe de la récompense est inversé à partir du $100^{\text{ème}}$ épisode. La fonction de valeur optimale est toujours linéaire en les fonctions de base et le vecteur de paramètres optimal est $\theta^*_{(-)} = [-35,-23,-11,0]^T$ avant le changement de récompense et $\theta^*_{(+)} = -\theta^*_{(-)}$ après.

L'environnement étant maintenant non-stationnaire, nous ajoutons une comparaison à

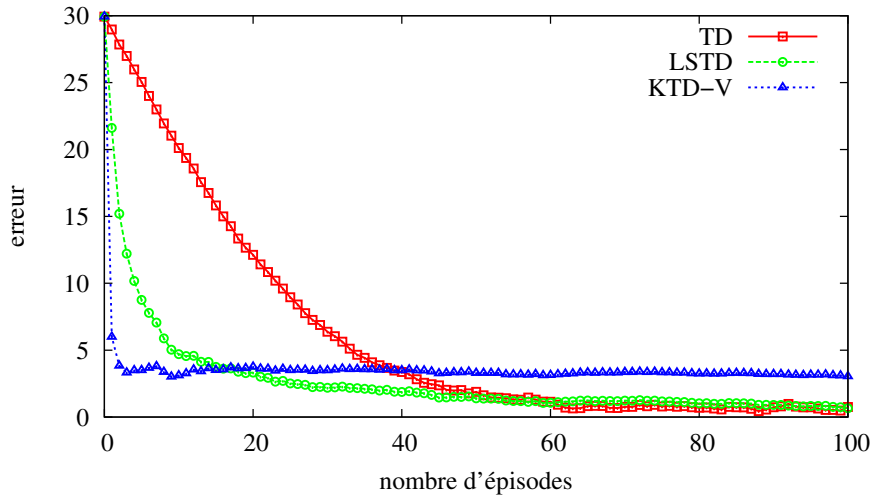


FIG. 3.5 – Chaîne de Boyan, cas stochastique.

GPTD (ce qui était inutile dans l'expérience précédente, étant donné qu'à bruit de process nul et paramétrisation linéaire, les deux algorithmes sont identiques). Les paramètres des différents algorithmes déjà considérés sont les mêmes, sauf le bruit de process qui est maintenant fixé à $P_{v_i} = 10^{-3}I$. Pour GPTD, les mêmes paramètres que pour KTD-V sont utilisés, sauf le bruit de process qui n'en fait pas partie. Les résultats sont présentés sur la figure 3.6.

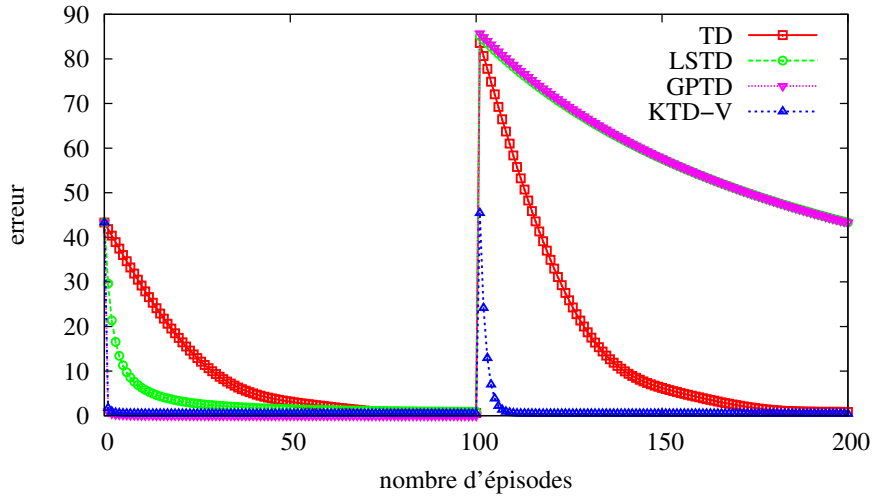


FIG. 3.6 – Chaîne de Boyan, cas déterministe et non-stationnaire.

A nouveau KTD-V converge plus rapidement que LSTD et TD, cependant maintenant vers le vecteur de paramètres optimal, l'environnement étant déterministe. Après le changement de récompense, LSTD est très lent à converger, à cause de la non-stationnarité induite. TD s'adapte plus rapidement, le taux d'apprentissage étant constant. KTD-V s'adapte quant à lui très rapidement. Les comportements de KTD-V et GPTD sont équiva-

lents dans la première phase, mais après le changement de récompense la courbe de GPTD se confond avec celle de LSTD, l'algorithme ne permet pas de traquer la solution. Ainsi, si KTD-V est biaisé dans le cas stochastique, il converge plus vite et s'adapte plus rapidement que TD et LSTD. Cette capacité à prendre en compte les non-stationnarités est importante pour suivre la dynamique de la fonction de valeur. Même si l'environnement est stationnaire, cela peut être utile dans un contexte de contrôle, comme illustré dans la section 3.8.3.

Cas déterministe et stationnaire

L'objectif de cette section est de comparer KTD à LSTD(λ) sur la chaîne de Boyan rendue déterministe. Pour les détails concernant LSTD(λ), le lecteur peut se référer à [18]. Cependant, l'algorithme qui y est présenté n'est pas en ligne et nous utilisons sa version récursive (qui s'obtient de manière très similaire à ce que nous avons présenté, en utilisant le lemme d'inversion matricielle), présentée par exemple par [27, chapitre 1.4, algorithme 9]. En effet, dans la mesure où les traces d'éligibilité permettent d'améliorer la vitesse de convergence ainsi que le comportement asymptotique de LSTD, il est plus équitable de les considérer également dans nos comparaisons à KTD. Pour les différents algorithmes nous considérons les mêmes paramètres que dans la section sur le cas stochastique et l'environnement est stationnaire. Nous présentons les résultats en échelle semi-logarithmique sur la figure 3.7.

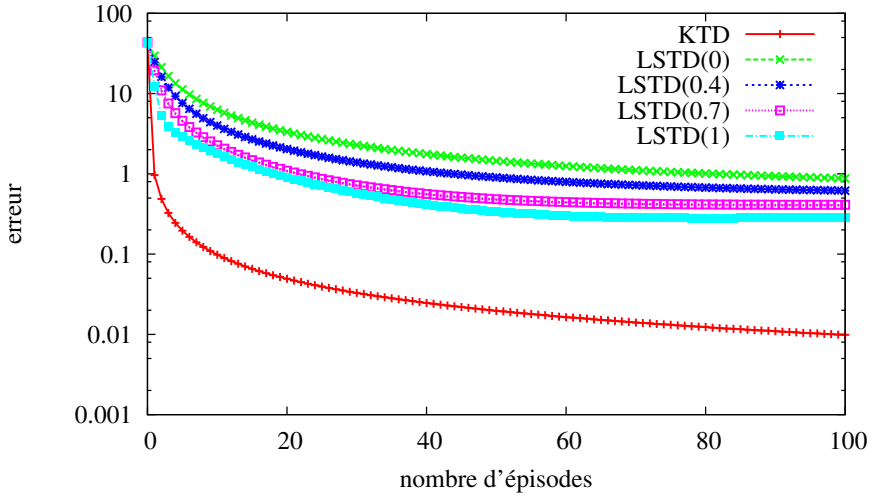


FIG. 3.7 – Chaîne de Boyan, comparaisons à LSTD(λ).

Augmenter la valeur de λ permet effectivement d'améliorer la vitesse de convergence ainsi que le comportement asymptotique, cependant dans aucun cas les performances de KTD ne sont atteintes. Nous considérerons donc généralement LSTD(0), sauf mention du contraire.

3.8.2 Pendule inversé

La seconde expérience est le pendule inversé tel que décrit par [73] et illustré figure 3.8. Le but est ici de comparer deux algorithmes de type itération de la valeur, à savoir KTD-Q

et Q-learning avec approximation de la Q-fonction, qui ont tous deux pour objectif d'estimer directement la fonction de qualité optimale. Autant que nous le sachions, KTD-Q est le seul algorithme d'ordre deux qui suive un schéma d'itération de la valeur (la difficulté majeure étant l'opérateur max), ce qui explique sa comparaison à un algorithme d'ordre 1.

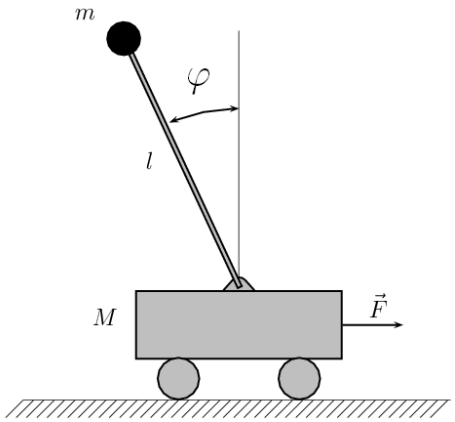


FIG. 3.8 – Pendule inversé illustré.

Cette tâche nécessite de balancer un pendule de longueur et masse inconnues de façon à ce qu'il reste vertical en applicant des forces au chariot sur lequel il est fixé. Trois actions sont possibles : pousser à gauche (-1), pousser à droite ($+1$), ou ne rien faire (0). L'état du système est donné par la position angulaire φ et la vitesse angulaire $\dot{\varphi}$. Les transitions déterministes sont calculées grâce à la dynamique du système physique :

$$\ddot{\varphi} = \frac{g\sin(\varphi) - \beta ml\dot{\varphi}^2\sin(2\varphi)/2 - 50\beta\cos(\varphi)a}{4l/3 - \beta ml\cos^2(\varphi)} \tag{3.57}$$

où g est la constante de gravitation, m et l sont la masse et la longueur du pendule, M la masse du chariot et $\beta = \frac{1}{m+M}$. Une récompense nulle est donnée tant que la position angulaire appartient à l'intervalle $[-\frac{\pi}{2}, \frac{\pi}{2}]$. Sinon, l'épisode se termine et une récompense de -1 est donnée. La paramétrisation est donnée par un terme constant et un ensemble de 9 noyaux gaussiens équi-répartis (centrés en $\{-\frac{\pi}{4}, 0, \frac{\pi}{4}\} \times \{-1, 0, 1\}$ et d'écart-type 1), cela pour chaque action. Il y a donc 30 fonctions de base. Le facteur d'actualisation γ est fixé à 0.95.

Nous comparons donc la capacité des deux algorithmes à apprendre la politique optimale. Pour Q-learning, le taux d'apprentissage est fixé à $\alpha_i = \alpha_0 \frac{n_0+1}{n_0+i}$ où $\alpha_0 = 0.5$ et $n_0 = 200$, en accord avec [73]. Pour KTD-Q, les paramètres sont $P_{0|0} = 10I$, $P_{n_i} = 1$ et $P_{v_i} = 0I$. Les vecteurs de paramètres sont initialisés à zéro. La politique suivie par l'agent est aléatoire (équi-probabilité des actions) et les deux algorithmes apprennent à partir des mêmes trajectoires. L'agent est initialisé dans un état aléatoire proche de l'équilibre $(0, 0)$. La longueur moyenne d'un tel épisode aléatoire est d'environ 10 pas. Les résultats sont présentés figure 3.9.

Pour chaque essai, l'apprentissage est fait sur 1000 épisodes. Tous les 50 épisodes, il est gelé et la politique courante est testée. Pour cela, l'agent est initialisé dans un état aléatoire proche de l'équilibre et la politique gloutonne est suivie. Chaque test est répété 100 fois et moyenné (l'évaluation de la performance de la politique gloutonne courante étant initialisée

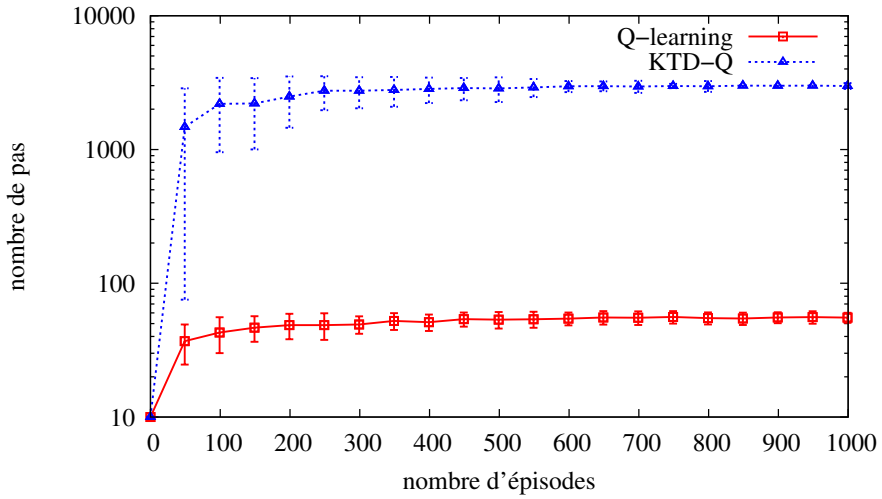


FIG. 3.9 – Pendule inversé, apprentissage de la politique optimale.

dans un état aléatoire). La mesure de performance est le nombre de pas d'un épisode. Le nombre maximum de pas autorisé est de 3000, ce qui correspond à maintenir le pendule pendant 5 minutes. Les résultats de la figure 3.9 sont moyennés sur 100 essais (chaque essai correspondant à 1000 épisodes, la politique étant testée tous les 50 épisodes) et présentés en échelle semi-logarithmique.

Asymptotiquement, KTD-Q apprend la politique optimale (c'est-à-dire maintenir le pendule pendant le nombre maximum de pas autorisés) et de bonnes politiques sont apprises après seulement quelques dizaines d'épisodes (en terme de simulation, moins de deux minutes d'interactions avec le pendule). Avec le même nombre d'épisodes et la même paramétrisation, Q-learning échoue à apprendre une politique qui permette de maintenir le pendule pendant plus de quelques secondes, ce qui est en accord avec les résultats présentés par [73]. Notons que l'algorithme LSPI [73] (*Least-Squares Policy Iteration*) permet d'obtenir des résultats similaires à KTD-Q en terme de vitesse de convergence et de performance, cependant cette approche est hors-ligne. Elle n'est donc pas considérée ici, l'objectif étant de comparer des algorithmes du type itération de la valeur (donc liés à l'équation d'optimalité de Bellman).

3.8.3 *Mountain car*

La dernière expérience de cette section est le *mountain car* telle que décrite par [105] et illustrée figure 3.10. L'objectif ici est d'illustrer le comportement des algorithmes dans le cadre d'un schéma d'itération de la politique optimiste : apprendre en contrôlant induit des dynamiques de la valeur non-stationnaires. Cette tâche consiste à conduire un véhicule sous-puissant en haut d'une route de montagne raide, la gravité étant plus forte que le moteur de la voiture. L'état est donné par la position et la vitesse $(x, \dot{x}) \in [-1.2, 0.5] \times [-0.07, 0.07]$. Les trois actions possibles sont aller à gauche (-1), à droite $(+1)$ ou ne rien faire (0). La dynamique du système est donnée par :

$$\begin{cases} \dot{x}_{i+1} = \text{bound}\left[\dot{x}_i + 10^{-3}(a_i - 2.5\cos(3x_i))\right] \\ x_{i+1} = \text{bound}\left[x_i + \dot{x}_{i+1}\right] \end{cases} \tag{3.58}$$

où l'opérateur bound force les bornes de la position et de la vitesse. Quand la position atteint la borne inférieure, la vitesse est mise à zéro. Quand elle atteint la borne supérieure, l'épisode se termine avec une récompense nulle. La récompense est de -1 le reste du temps. Le facteur d'actualisation est fixé à 0.1. L'état est normalisé et la paramétrisation est composée d'un terme constant et d'un ensemble de 9 noyaux gaussiens équi-répartis (centrés en $\{0, 0.5, 1\} \times \{0, 0.5, 1\}$ et d'écart-type 0.1), cela pour chaque action. Il y a donc 30 fonctions de base.

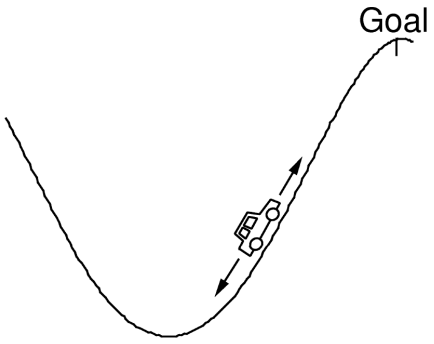


FIG. 3.10 – *Mountain car* illustré [105].

Cette expérience compare SARSA avec approximation de la fonction de qualité, LSTD et KTD-SARSA, dans un contexte d'itération optimiste de la politique. La politique suivie est ϵ-gloutonne, avec $\epsilon = 0.1$. Pour SARSA, le taux d'apprentissage est fixé à $\alpha_i = 0.1$. Pour LSTD l'*a priori* est fixé à $10I$. Pour KTD-SARSA, le même *a priori* est utilisé et la variance du bruit d'observation est fixée à $P_{n_i} = 1$. Pour le bruit d'évolution nous utilisons une méthode de filtrage adaptatif qui consiste à prendre comme variance du bruit d'évolution une fraction de la variance sur les paramètres, c'est-à-dire :

$$P_{v_i} = \eta P_{\theta_{i-1|i-1}} \tag{3.59}$$

où $\eta \ll 1$ est une constante positive, choisie ici égale à 10^{-5}. Ce schéma est en fait inspiré par les moindres carrés adaptatifs, voir [116] par exemple. Pour tous les algorithmes le vecteur de paramètres est initialisé à zéro. Chaque épisode commence dans un état aléatoire (tirage uniforme sur le domaine). Un maximum de 1500 pas est autorisé.

Pour chaque essai, l'apprentissage se fait sur 200 épisodes, la figure 3.11 montre la longueur de chaque épisode moyennée sur 300 essais. KTD-SARSA converge plus rapidement et vers une meilleure politique que LSTD, qui se comporte mieux que SARSA avec approximation de la Q-fonction. De meilleurs résultats ont peut être été rapportés dans la littérature pour une paramétrisation de type *tile-coding*, cependant la paramétrisation choisie ici est plus brute et implique bien moins de paramètres. De plus, même avec une paramétrisation de type *tile-coding* et des paramètres optimisés, il est rapporté par [105, chapitre 8.4] qu'il faut une centaine d'épisodes pour atteindre une politique optimale, ce qui est environ un ordre de magnitude de plus que KTD. Le fait de suivre une politique ϵ-gloutonne implique la non-stationnarité de la Q-fonction apprise, ce qui explique que LSTD échoue à trouver une politique quasi-optimale (le même type de résultat est obtenu par [84]). LSTD a été étendu par [73] à LSPI (*Least-Squares Policy Iteration*), qui permet de chercher une politique optimale plus efficacement. Cependant, cet algorithme est

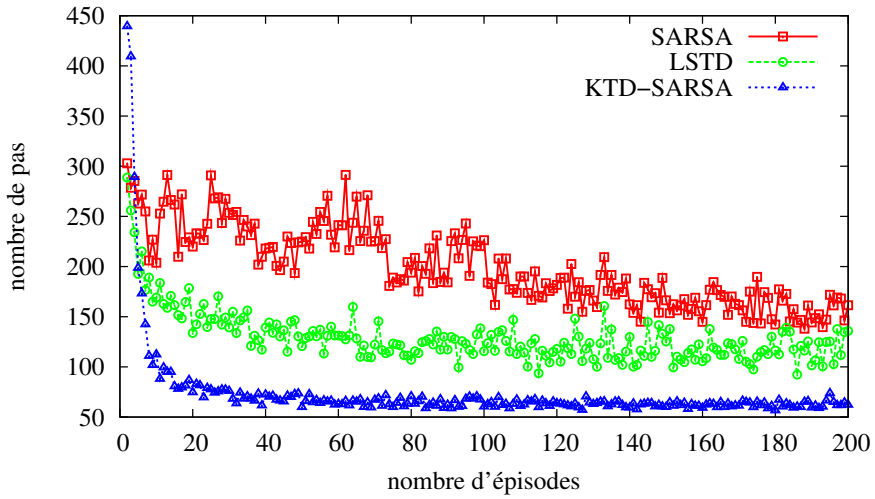


FIG. 3.11 – *Mountain car.*

hors-ligne et n'implique pas d'apprendre tout en contrôlant, c'est pourquoi il n'est pas comparé ici. KTD-SARSA obtient des politiques optimales très rapidement, après seulement quelques dizaines d'épisodes. L'apprentissage est également plus stable avec l'algorithme proposé (variance plus faible).

Chapitre 4

Biais de l'estimateur et modèle de bruit coloré

Le cadre de travail des différences temporelles de Kalman que nous avons proposé fait l'hypothèse d'un bruit d'observation blanc, ce que nous avons annoncé section 3.6 ne se vérifie que dans le cas de MDP déterministes. Le problème posé par la stochasticité des transitions a été mis en avant à partir du résultat du théorème 3.1, obtenu sous cette hypothèse particulière et illustré dans la section 3.8 sur une simple chaîne de Markov valuée. De façon fondamentale, c'est cette hypothèse de bruit d'observation blanc qui est caduque si les transitions sont stochastiques, alors que c'est une propriété importante pour l'obtention des équations de Kalman. Si l'on écrit l'équation d'évaluation de Bellman pour la fonction de valeur dans le cas de transitions stochastiques sous la forme de l'équation d'observation de KTD, le bruit d'observation n'est pas seulement un bruit de modélisation, il contient la stochasticité du MDP :

$$r_i = \hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) + n_i \text{ avec}$$
$$n_i = \gamma \left(E_{s'|s_i,\pi(s_i)}[\hat{V}_{\theta_i}(s')] - \hat{V}_{\theta_i}(s_i) \right) + r_i - E_{s'|s_i,\pi(s_i)}[R(s_i, \pi(s_i), s')] \tag{4.1}$$

Il est clair en considérant l'équation (4.1) que l'hypothèse de blancheur du bruit d'observation est trop forte, ce qui rend l'estimateur de KTD biaisé dans des environnements stochastiques. Ce problème est similaire à celui d'erreur dans les variables introduit dans le cadre de la présentation de l'algorithme LSTD section 2.3.2, voir plus particulièrement l'équation (2.58). Pour répondre à cela LSTD utilise le concept de variable instrumentale. Cependant, ce dernier ne peut *a priori* pas être pris en compte dans les approches proposées. A notre connaissance, il n'y a que [68] qui traite l'usage de variables instrumentales dans le contexte du filtrage de Kalman. Toutefois, la linéarité du problème est une hypothèse très fortement ancrée dans les techniques développées ; nous ne creusons pas cette approche. Pour prendre en compte le cas des transitions stochastiques dans le cadre de KTD, nous colorons le bruit.

Nous montrons dans un premier temps que la fonction de coût minimisée par KTD est biaisée lorsque les transitions sont stochastiques. La forme de ce biais est similaire à celle résultante de la minimisation du résidu quadratique de Bellman (2.52), bien que plus complexe. Nous présentons ensuite un modèle de bruit coloré, introduit initialement par [30]. Ce bruit est un bruit à moyenne mobile, sa prise en compte dans le cadre du filtrage de Kalman n'est pas usuelle (à notre connaissance ce problème n'a jamais été traité

dans la littérature). Nous proposons donc une méthode permettant de le faire. Cela nous permet de dériver un certain nombre d'algorithmes qui s'appliquent aux environnements stochastiques. Nous nommons ce cadre de travail XKTD (*eXtended Kalman Temporal Differences*). Nous présentons également une version pondérée de XKTD et proposons une analyse de convergence qui montre que comme l'algorithme LSTD(1), XKTD minimise un coût quadratique associant aux valeurs un cumul pondéré de récompenses qui pourrait être obtenu par Monte Carlo. Une série d'expériences conclut le chapitre.

4.1 Stochasticité des transitions et biais

Sous l'hypothèse de distributions gaussiennes, le théorème 3.1 montre que KTD minimise un résidu quadratique de Bellman, ce qui en fait un estimateur biaisé dans le cas de transitions stochastiques, en vertu de l'équation (2.52). Il est possible d'obtenir un résultat plus général, en considérant la fonction de coût (3.5) directement minimisée par KTD.

Théorème 4.1. *Supposons que la fonction de récompense ne dépend que du couple état-action courant et pas de l'état vers lequel transite le système. Lorsqu'elle est utilisée avec un processus décisionnel de Markov stochastique, la fonction coût (3.5) est biaisée, ce biais étant donné par :*

$$\begin{aligned} &\|K_i\|^2 E\left[\mathrm{cov}_{s'|s_i,a_i}\left(r_i - g_{t_i}(\theta)\right)|r_{1:i-1}\right] \\ &= \begin{cases} \|K_i\|^2 E\left[\mathrm{cov}_{s'|s_i,\pi(s_i)}\left(r_i + \gamma V_\theta(s')\right)|r_{1:i-1}\right] \\ \|K_i\|^2 E\left[\mathrm{cov}_{s'|s_i,\pi(s_i)}\left(r_i + \gamma Q_\theta(s',\pi(s'))\right)|r_{1:i-1}\right] \\ \|K_i\|^2 E\left[\mathrm{cov}_{s'|s_i,a_i}\left(r_i + \gamma \max_{a\in A} Q_\theta(s',a)\right)|r_{1:i-1}\right] \end{cases} \end{aligned} \tag{4.2}$$

Il est clair que ce biais est nul pour des transitions déterministes.

Démonstration. L'hypothèse que la fonction de récompense ne dépend pas de l'état vers lequel transite le système est faite pour simplifier la démonstration, dans la mesure où la fonction de coût est conditionnée aux récompenses observées dans le passé. Cependant, l'idée générale reste la même. Sous les hypothèses données, le modèle espace d'état qu'il faudrait considérer pour un MDP stochastique est le suivant :

$$\begin{cases} \theta_i = \theta_{i-1} + v_i \\ r_i = E_{s'|s_i,a_i}\left[g_{t_i}(\theta_i)\right] + n_i \end{cases} \tag{4.3}$$

où t_i définit maintenant une transition aléatoire : $t_i = (s_i, a_i, s')$. Notons que l'équation d'observation (moins le bruit) est l'équation de Bellman dans le cas de transitions stochastiques. La différence avec le modèle espace d'état (3.4) est que les transitions ne sont plus échantillonnées mais moyennées. Dans ce cas le bruit d'observation est uniquement un bruit de modélisation, il peut donc être supposé blanc. La fonction de coût associée est :

$$\begin{aligned} \mathcal{J}_i(\theta) &= \mathrm{trace}\left(\mathcal{P}_{i|i}\right) \\ &= \mathrm{trace}\left(\mathcal{P}_{i|i-1} - \mathcal{P}_{\theta r_i} K_i^T - K_i \mathcal{P}_{\theta r_i}^T - K_i \mathcal{P}_{r_i} K_i^T\right) \end{aligned} \tag{4.4}$$

où les notations calligraphiées représentent la même chose pour le modèle espace d'état (4.3)

que les notations (3.12) pour le modèle espace d'état (3.4), *exempli gratia* :

$$\mathcal{P}_{\theta r_i} = E\left[\tilde{\theta}_{i|i-1}\tilde{\mathfrak{r}}_i | r_{1:i-1}\right] \tag{4.5}$$

$$\begin{aligned}\text{avec } \tilde{\mathfrak{r}}_i &= r_i - \hat{\mathfrak{r}}_{i|i-1} \\ &= r_i - E\left[E_{s'|s_i,a_i}\left[g_{t_i}(\theta_i)\right]|r_{1:i-1}\right]\end{aligned} \tag{4.6}$$

Notons que la prédiction de la récompense est non biaisée, donc l'innovation également (linéarité en les termes dépendant de s') :

$$E_{s'|s_i,a_i}\left[\hat{r}_{i|i-1}\right] = \hat{\mathfrak{r}}_{i|i-1} \tag{4.7}$$

$$E_{s'|s_i,a_i}\left[\tilde{r}_{i|i-1}\right] = \tilde{\mathfrak{r}}_{i|i-1} \tag{4.8}$$

Le terme $P_{i|i-1}$ ne dépend pas de l'état s' vers lequel transite le système et le terme $P_{\theta r_i}$ est linéaire en l'innovation (qui est sa seule dépendance à s'), ils sont donc tous deux non-biaisés :

$$E_{s'|s_i,a_i}\left[P_{i|i-1}\right] = \mathcal{P}_{i|i-1} \tag{4.9}$$

$$E_{s'|s_i,a_i}\left[P_{\theta r_i}\right] = \mathcal{P}_{\theta r_i} \tag{4.10}$$

Ce n'est pas le cas pour la variance de l'innovation (dépendance quadratique aux termes faisant intervenir s') :

$$\begin{aligned}E_{s'|s_i,a_i}\left[P_{r_i}\right] &= E_{s'|s_i,a_i}\left[E\left[\tilde{r}^2_{i|i-1}|r_{i-1}\right]\right] \\ &= E\left[E_{s'|s_i,a_i}\left[\tilde{r}^2_{i|i-1}\right]|r_{i-1}\right] \\ &= E\left[\tilde{\mathfrak{r}}^2_i|r_{1:i-1}\right] + E\left[E_{s'|s_i,a_i}\left[\tilde{r}^2_i\right] - \left(E_{s'|s_i,a_i}\left[\tilde{r}_i\right]\right)^2\right] \\ &= \mathcal{P}_{r_i} + E\left[\mathrm{cov}_{s'|s_i,ai}\left(\tilde{r}_i\right)|r_{1:i-1}\right]\end{aligned} \tag{4.11}$$

Ainsi le biais peut être calculé :

$$\begin{aligned}E_{s'|s_i,a_i}\left[J_i(\theta)\right] - \mathcal{J}_i(\theta) &= E_{s'|s_i,a_i}\left[\mathrm{trace}\left(K_i\left(P_{r_i} - \mathcal{P}_{r_i}\right)K_i^T\right)\right] \\ &= \mathrm{trace}(K_iK_i^T)E_{s'|s_i,a_i}\left[P_{r_i} - \mathcal{P}_{r_i}\right] \\ &= K_i^TK_i\left(E_{s'|s_i,a_i}\left[P_{r_i}\right] - \mathcal{P}_{r_i}\right) \\ &= \|K_i\|^2E\left[\mathrm{cov}_{s'|s_i,a_i}\left(r_i - g_{t_i}(\theta)\right)|r_{1:i-1}\right]\end{aligned} \tag{4.12}$$

Notons que ni $V_\theta(s_i)$ ni $Q_\theta(s_i)$ ne dépendent de l'état s' vers lequel transite le système, d'où le résultat tel qu'exprimé dans le théorème 4.1. □

Ce biais est donc similaire à celui obtenu par la minimisation d'un résidu quadratique de Bellman (2.52) et comme lui il tend à favoriser les fonctions de valeur régulières. Pour supprimer ce biais, une solution pourrait être d'introduire un filtre auxiliaire (ou d'étendre le filtre), de la même façon qu'une fonction auxiliaire est introduite par [2, 3]. Cependant, cette approche induit une fonction de coût qui n'est pas purement quadratique, ce qui rend son extension à Kalman particulièrement difficile, si même possible. L'extension de ces travaux n'est donc pas évidente. Une autre approche pourrait être d'estimer ce biais en ligne pour l'enlever, de façon similaire à ce qui est fait par [56] dans le cadre du filtrage par moindres

carrés. Cependant, le filtrage de Kalman est un cadre de travail plus complexe, spécialement lorsqu'il est combiné à la transformation non-parfumée. L'extension de ces travaux est donc également loin d'être facile. Une autre perspective intéressante est d'introduire un bruit d'observation coloré comme cela est fait par [30, 27] pour les algorithmes de la famille GPTD. C'est en adaptant cette dernière approche que nous étendons KTD.

4.2 Un modèle de bruit coloré

Ainsi le fait d'utiliser KTD sur un problème dont la dynamique est stochastique induit un biais dans la fonction de coût minimisée. Remarquons que la suppression du bais causé par le caractère stochastique des transitions dans les approches visant à minimiser un résidu quadratique de Bellman[1] est un problème récurrent et important de l'apprentissage par renforcement [6, 105, 73, 2, 3, 94].

Dans cette section, nous nous focalisons sur le problème d'évaluation de la fonction de valeur (ou de qualité). L'optimisation directe de la Q-fonction est discutée plus tard, en raison de son aspect *off-policy* qui pose problème dans le cadre du modèle d'extension de KTD proposé, comme nous le verrons. Dans un premier temps nous présentons d'où vient ce modèle de bruit coloré introduit à l'origine par [30, 27] : basiquement, cette coloration provient d'une réécriture de l'équation d'évaluation de Bellman, sous une forme plus stochastique. Le bruit obtenu est un bruit à moyenne mobile, ce qui est loin d'être usuel dans le cadre du filtrage de Kalman. Nous proposons donc une approche permettant de le prendre en compte dans ce cadre de travail. Ceci permet de proposer un nouvel algorithme général, XKTD (*eXtended Kalman Temporal Differences*), qui est spécialisé en XKTD-V et XKTD-SARSA en utilisant la transformation non parfumée.

4.2.1 Modélisation du bruit

Comme annoncé, nous nous intéressons au problème de l'évaluation pure. La politique π est donc fixée et le MDP se réduit en une chaîne de Markov valuée de probabilité de transition p^π et de récompense R^π définis par :

$$p^\pi(.|s) = p(.|s, \pi(s)) \text{ et } R^\pi(s, s') = R(s, \pi(s), s') \tag{4.13}$$

Comme le fait [27] pour introduire ce bruit coloré, nous définissons le retour pondéré D^π comme étant le processus aléatoire suivant :

$$D^\pi(s) = \sum_{i=0}^{\infty} \gamma^i R^\pi(S_i, S_{i+1}) | S_0 = s,\ S_{i+1} \sim p^\pi(.|S_i) \tag{4.14}$$

Il est assez évident que la fonction de valeur est par définition l'espérance (sur l'ensemble des trajectoires possibles) de ce processus aléatoire :

$$V^\pi(s) = E[D^\pi(s)] \tag{4.15}$$

[1] En raison des résultats du théorème 3.1, KTD peut être vu comme une telle méthode.

Le processus aléatoire D^π peut donc être décomposé en la somme de la fonction de valeur et d'un résidu centré que nous notons $\Delta V^\pi(s)$:

$$\begin{aligned} D^\pi(s) = & \underbrace{E[D^\pi(s)]}_{\text{fonction de valeur}} + \underbrace{D^\pi(s) - E[D^\pi(s)]}_{\text{résidu centré}} \\ & = V^\pi(s) + \Delta V^\pi(s) \end{aligned} \tag{4.16}$$

D'autre part, de la même façon que l'équation d'évaluation de Bellman définit une récurrence anti-causale sur la fonction de valeur, une relation similaire existe sur le processus aléatoire de retour pondéré :

$$\begin{aligned} D^\pi(s) &= \sum_{i=0}^{\infty} \gamma^i R^\pi(S_i, S_{i+1}) | S_0 = s,\ S_{i+1} \sim p^\pi(.|S_i) \\ &= R^\pi(s, S') + \gamma D^\pi(S'),\ S' \sim p^\pi(.|s) \end{aligned} \tag{4.17}$$

En injectant l'équation (4.16) dans l'équation (4.17), la récompense associée à une transition peut s'écrire comme une fonction des valeurs aux deux états consécutifs plus un terme de bruit :

$$\begin{aligned} R^\pi(s, S') &= D^\pi(s) - \gamma D^\pi(S'),\ S' \sim p^\pi(.|s) \\ &= V^\pi(s) + \Delta V^\pi(s) - \gamma \left(V^\pi(s') + \Delta V^\pi(s') \right),\ S' \sim p^\pi(.|s) \\ &= V^\pi(s) - \gamma V^\pi(S') + N(s, S'),\ S' \sim p^\pi(.|s) \end{aligned} \tag{4.18}$$

Dans cette dernière équation, le bruit N dépend des résidus consécutifs, son expression est donnée par

$$N(s, s') = \Delta V^\pi(s) - \gamma \Delta V^\pi(s') \tag{4.19}$$

Ce bruit ayant été défini, nous faisons comme [27] une hypothèse très forte (et même généralement fausse) sur le processus des résidus : nous le supposons blanc (c'est-à-dire notamment que les résidus sont indépendants entre eux). L'apprentissage se faisant à partir de trajectoires et les trajectoires associées aux résidus de deux états successifs ne différant que par un état, les résidus associés sont peu susceptibles d'être indépendants. Cependant, cette supposition permet tout de même de parvenir à une preuve de convergence, telle qu'exposée dans la suite de ce chapitre. Une interprétation intéressante de cette hypothèse de blancheur des résidus, dans le contexte des process gaussiens [30], peut être trouvée dans [27, chapitre 4.4.3.]. Il y est montré (et notre résultat est très similaire, bien que plus général) qu'utiliser ce bruit coloré revient à faire de la régression par moindres carrés pour la fonction de valeur, les cibles étant des échantillons de Monte Carlo du cumul pondéré des récompenses, obtenus à partir de la même trajectoire (deux échantillons consécutifs différent uniquement par la récompense d'un état). Cette hypothèse de blancheur des résidus revient donc à ne pas respecter l'hypothèse d'indépendance statistique des échantillons sous-jacente à Monte Carlo.

Maintenant que nous avons une nouvelle forme de bruit (4.19), il reste à l'intégrer d'une part au modèle espace d'état, puis à adapter l'algorithme de résolution de KTD. Rappelons l'équation d'observation de la formulation espace d'état (3.4) :

$$r_i = g_{t_i}(\theta_i) + n_i \tag{4.20}$$

Pour KTD, le bruit d'observation est supposé blanc, ce qui est nécessaire à l'obtention des équations de Kalman et donc à la dérivation de l'algorithme. Pour XKTD, le modèle

de bruit coloré[2] (4.19) est utilisé à la place. Les résidus étant centrés (4.16) et supposés indépendants, on peut modéliser les résidus d'une trajectoire par un bruit blanc centré de variance $\sigma_i^2 = \sigma^2(s_{i+1})$:

$$u_i = \Delta V^\pi(s_{i+1}) \sim (0, \sigma_i^2) \tag{4.21}$$

Ainsi, d'après cette modélisation et l'équation du bruit coloré (4.19), le bruit d'observation est la somme (pondérée) de deux bruits blancs :

$$n_i = -\gamma u_i + u_{i-1}, \quad u_i \sim (0, \sigma_i^2) \tag{4.22}$$

Notons que nous ne faisons pas d'hypothèse sur le caractère gaussien ou non du bruit blanc ; nous ne nous intéressons qu'à ses moments d'ordre un et deux (qui caractérisent totalement la distribution dans le cas gaussien), car c'est ce qui est utile dans le cadre de Kalman. Le bruit d'observation est donc un bruit à moyenne mobile[3] d'ordre 1 (ou bruit MA pour *moving average*). Etant donné que KTD fait l'hypothèse de blancheur du bruit d'observation, un problème est de l'étendre aux bruits à moyenne mobile.

4.2.2 Prise en compte d'un bruit à moyenne mobile dans un filtre de Kalman

Nous allons nous abstraire pour l'instant de l'apprentissage par renforcement et poser le problème général de la prise en compte d'un bruit d'observation à moyenne mobile dans le cadre du filtrage de Kalman. Autant que nous le sachions, ce problème n'a jamais été traité auparavant dans la littérature : si le cas d'un bruit auto-régressif [4] (ou bruit AR pour *autoregressive*) est très classique [53], nous n'avons connaissance que d'une publication traitant le cas d'un bruit d'évolution MA [100], technique qui n'est pas adaptable au bruit d'observation. L'approche que nous proposons consiste à transformer un bruit MA en bruit AR, afin de le prendre en compte assez classiquement. Nous discutons d'abord de la prise en compte d'un bruit AR dans le cadre du filtrage de Kalman. Soit un modèle espace-d'état assez[5] générique :

$$\begin{cases} x_i = x_{i-1} + v_i \\ y_i = h_i(x_i) + n_i \end{cases} \tag{4.25}$$

L'équation d'évolution est une marche aléatoire des états cachés x_i et l'équation d'observation lie les observations y_i aux états x_i via la fonctionnelle h_i, le bruit n_i étant ici supposé auto-régressif d'ordre 1. Ce bruit peut donc se mettre sous la forme suivante :

$$n_i = A_i n_{i-1} + u_i \tag{4.26}$$

[2]Par définition, un bruit coloré est un bruit qui n'est pas blanc.

[3]Un bruit y est dit à moyenne mobile d'ordre n si le bruit à l'instant i est une combinaison linéaire des n échantillons précédents d'un bruit blanc u :

$$y_i = \sum_{k=0}^{n} b_k u_{i-k} \tag{4.23}$$

[4]Un bruit y est dit auto-régressif d'ordre n si le bruit à l'instant i est une combinaison linéaire des n échantillons précédents et d'un bruit blanc u :

$$y_i = \sum_{k=1}^{n} a_k y_{i-k} + u_i \tag{4.24}$$

[5]Il pourrait être écrit de façon plus générique, mais cela n'apporterait rien à notre propos.

où A_i est une matrice de dimension *ad hoc* et u_i est un bruit blanc. Une technique classique pour prendre en compte ce type de bruit est d'étendre le vecteur d'état, ce qui donne le nouvel espace-d'état, qui est équivalent :

$$\begin{cases} \begin{pmatrix} x_i \\ n_i \end{pmatrix} = \begin{pmatrix} I & \mathbf{0} \\ \mathbf{0} & A_i \end{pmatrix} \begin{pmatrix} x_{i-1} \\ n_{i-1} \end{pmatrix} + \begin{pmatrix} v_i \\ u_i \end{pmatrix} \\ y_i = h_i(x_i) + n_i \end{cases} \tag{4.27}$$

Le bruit d'observation devient une composante du vecteur d'état, il est donc estimé, au même titre que l'état caché, au fur et à mesure que des observations sont effectuées.

Nous allons maintenant nous intéresser à un bruit MA. Etant donné le cas qui nous intéresse, nous allons nous cantonner aux bruits MA scalaires d'ordre 1. Cependant, les idées proposées ici s'étendent facilement à des cas plus généraux. L'idée principale de notre approche est d'exprimer ce bruit MA scalaire sous la forme équivalente d'un bruit AR vectoriel, de façon à l'intégrer au modèle espace-d'état comme vu précédemment. Supposons que ce bruit MA se mette sous la forme suivante :

$$n_i = \begin{pmatrix} a_i & b_i \end{pmatrix} \begin{pmatrix} u_i \\ u_{i-1} \end{pmatrix} = a_i u_i + b_i u_{i-1}, \quad u_i \sim (0, \sigma_i^2) \tag{4.28}$$

où u_i est blanc. Soit ω_i une variable aléatoire auxiliaire, dont le rôle va être de mémoriser le bruit blanc. Le bruit MA (4.28) est équivalent au bruit AR vectoriel suivant :

$$\begin{pmatrix} \omega_i \\ n_i \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ b_i & 0 \end{pmatrix} \begin{pmatrix} \omega_{i-1} \\ n_{i-1} \end{pmatrix} + \begin{pmatrix} 1 \\ a_i \end{pmatrix} u_i \tag{4.29}$$

En effet, d'après ce modèle de bruit AR vectoriel, nous avons que $n_i = b_i \omega_{i-1} + a_i u_i$, mais aussi $\omega_i = u_i$ et donc $n_i = a_i u_i + b_i u_{i-1}$, ce qui est exactement le bruit MA de l'équation (4.28). Le bruit u'_i défini par :

$$u'_i = \begin{pmatrix} 1 \\ a_i \end{pmatrix} u_i \tag{4.30}$$

est également centré et de variance :

$$\begin{aligned} P_{u'_i} &= E[u'_i (u'_i)^T] \\ &= \begin{pmatrix} 1 & a_i \end{pmatrix} E[u_i^2] \begin{pmatrix} 1 \\ a_i \end{pmatrix} \\ &= \sigma_i^2 \begin{pmatrix} 1 & a_i \\ a_i & a_i^2 \end{pmatrix} \end{aligned} \tag{4.31}$$

Le modèle espace d'état équivalent est donc :

$$\begin{cases} \begin{pmatrix} x_i \\ \omega_i \\ n_i \end{pmatrix} = \begin{pmatrix} I & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & 0 & 0 \\ \mathbf{0}^T & b_i & 0 \end{pmatrix} \begin{pmatrix} x_{i-1} \\ \omega_{i-1} \\ n_{i-1} \end{pmatrix} + \begin{pmatrix} v_i \\ u_i \\ a_i u_i \end{pmatrix} \\ y_i = h_i(x_i) + n_i \end{cases} \tag{4.32}$$

La façon de considérer un bruit d'observation à moyenne mobile dans le cadre du filtrage de Kalman, il ne reste plus qu'à utiliser ce principe pour étendre les différences temporelles de Kalman.

4.2.3 Algorithmes résultants

Maintenant que nous avons vu comment transformer un bruit MA scalaire en bruit AR vectoriel, il suffit d'appliquer la technique présentée au modèle de bruit (4.22) de façon à étendre le modèle espace-d'état (3.4). Une fois ce nouveau modèle espace-d'état obtenu, l'extension de KTD à XKTD est assez aisée.

Soit ω_i une variable aléatoire auxiliaire. Le bruit MA scalaire (4.22) est équivalent au bruit AR vectoriel suivant :

$$\begin{pmatrix} \omega_i \\ n_i \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_{i-1} \\ n_{i-1} \end{pmatrix} + \begin{pmatrix} 1 \\ -\gamma \end{pmatrix} u_i \tag{4.33}$$

Le bruit blanc associé $u'_i = \begin{pmatrix} u_i & -\gamma u_i \end{pmatrix}^T$ est centré et d'après (4.31) de variance :

$$P_{u'_i} = \sigma_i^2 \begin{pmatrix} 1 & -\gamma \\ -\gamma & \gamma^2 \end{pmatrix} \tag{4.34}$$

Ce nouveau bruit ayant été défini, il est possible d'étendre le modèle espace-d'état (3.4), ce qui donne le nouveau modèle suivant :

$$\begin{cases} \mathbf{x}_i = F\mathbf{x}_{i-1} + v'_i \\ r_i = g_{t_i}(\mathbf{x}_i) \end{cases} \tag{4.35}$$

Nous détaillons chacun de ces termes. Le vecteur de paramètres est maintenant étendu avec le bruit AR vectoriel $\begin{pmatrix} \omega_i & n_i \end{pmatrix}^T$:

$$\mathbf{x}_i = \begin{pmatrix} \theta_i \\ \omega_i \\ n_i \end{pmatrix} \tag{4.36}$$

La matrice d'évolution F prend en compte la structure du bruit d'observation MA. Nous rappelons que p désigne le nombre de paramètres et I_p la matrice identité de taille $p \times p$. La matrice d'évolution s'écrit par blocs de la façon suivante :

$$F = \begin{pmatrix} I_p & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & 0 & 0 \\ \mathbf{0}^T & 1 & 0 \end{pmatrix} \tag{4.37}$$

Le bruit d'évolution v_i est également étendu de façon à prendre en compte ce bruit coloré, $v'_i = \begin{pmatrix} v_i & u'_i \end{pmatrix}^T$; ce nouveau bruit de process est centré et sa matrice de variance $P_{v'_i}$ peut s'écrire par blocs en utilisant le fait que les bruits v_i et u'_i sont indépendants (et donc décorrélés) :

$$v'_i = \begin{pmatrix} v_i \\ u'_i \end{pmatrix} \sim \left(\begin{pmatrix} \mathbf{0} \\ 0 \\ 0 \end{pmatrix}, P_{v'_i} = \begin{pmatrix} P_{v_i} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \sigma_i^2 & -\gamma\sigma_i^2 \\ \mathbf{0}^T & -\gamma\sigma_i^2 & \gamma^2\sigma_i^2 \end{pmatrix} \right) \tag{4.38}$$

Enfin, l'équation d'observation reste la même, malgré la notation quelque peu abusive :

$$r_i = g_{t_i}(\mathbf{x}_i) = g_{t_i}(\theta_i) + n_i \tag{4.39}$$

Il est à noter qu'à présent le bruit d'observation fait partie de l'équation d'évolution, il va donc être estimé.

La nouvelle formulation espace-d'état ayant été présentée, KTD peut être étendu à XKTD. Le principe pour dériver ce nouvel algorithme est exactement le même que celui exposé dans le chapitre 3.2, avec cependant de légères différences. Premièrement, les équations de prédiction sont légèrement modifiées, la matrice d'évolution devant être prise en compte. Si l'on reprend l'équation (3.9), la prédiction pour le vecteur étendu moyen se fait selon :

$$\begin{aligned}\hat{\mathbf{x}}_{i|i-1} &= E[\mathbf{x}_i|r_{1:i-1}] \\ &= E[F\mathbf{x}_{i-1} + v'_i|r_{1:i-1}] \\ &= E[F\mathbf{x}_{i-1}|r_{1:i-1}] \\ &= F\hat{\mathbf{x}}_{i-1|i-1}\end{aligned} \tag{4.40}$$

De même, par rapport à l'équation (3.18) le calcul de la variance associée est légèrement différent :

$$\begin{aligned}P_{i|i-1} &= \text{cov}\left(\tilde{\mathbf{x}}_{i|i-1}|r_{1:i-1}\right) \\ &= \text{cov}\left(F\tilde{\mathbf{x}}_{i-1|i-1} + v'_i|r_{1:i-1}\right) \\ &= FP_{i-1|i-1}F^T + P_{v'_i}\end{aligned} \tag{4.41}$$

Une autre différence significative est que le bruit n'étant plus blanc, la variance de l'innovation ne peut plus être simplifiée en séparant le terme de variance du bruit d'observation, autrement dit le résultat de l'équation (3.20) n'est plus valable. Il faut considérer son expression avant simplification :

$$P_{r_i} = E\left[(g_{t_i}(\theta_i) - \hat{r}_{i|i-1} + n_i)^2|r_{1:i-1}\right] \tag{4.42}$$

Pour la même raison, la simplification de l'équation (3.19) n'est plus possible, nous avons donc :

$$P_{\mathbf{x}r_i} = \left[(\mathbf{x}_i - \hat{\mathbf{x}}_{i|i-1})(g_{t_i}(\theta_i) - \hat{r}_{i|i-1} + n_i)|r_{1:i-1}\right] \tag{4.43}$$

La prédiction de la récompense doit également prendre en compte le bruit d'observation, son espérance conditionnée aux récompenses observées dans le passé n'étant pas forcément nulle :

$$\hat{r}_{i|i-1} = E\left[g_{t_i}(\theta_i) + n_i|r_{1:i-1}\right] \tag{4.44}$$

Tous les autres développements faits pour le modèle espace-d'état (3.4) sont valables pour (4.35). L'approche générale de XKTD est résumée dans l'algorithme 4.1.

Exactement de la même façon que la transformation non-parfumée présentée section 3.4 a été utilisée pour spécialiser l'algorithme KTD, elle peut être utilisée pour spécialiser l'algorithme XKTD. Nous rappelons que pour l'instant nous nous focalisons sur le problème d'évaluation pure de la fonction de valeur (ou de qualité) et donc aux algorithmes XKTD-V et XKTD-SARSA. Le cas de XKTD-Q sera discuté dans la section suivante. Nous rappelons que lorsque les transformations sont linéaires (cas d'une paramétrisation linéaire pour le problème d'évaluation pure) la transformation non-parfumée n'est plus une approximation et les algorithmes qui en résultent sont équivalents à la formulation algébrique qui peut en être faite en résolvant les équations de Kalman de façon analytique. Nous ne donnons donc que la formulation basée sur la transformation non-parfumée.

Algorithme 4.1 : XKTD général

Initialisation : *a priori* $\hat{\mathbf{x}}_{0|0}$ et $P_{0|0}$;

pour $i \leftarrow 1, 2, \ldots$ **faire**

Observer la transition t_i ainsi que la récompense r_i ;

Phase de prédiction;
$\hat{\mathbf{x}}_{i|i-1} = F\hat{\mathbf{x}}_{i-1|i-1}$;
$P_{i|i-1} = FP_{i-1|i-1}F^T + P_{v'_i}$;

Calcul des statistiques d'intérêt;
$\hat{r}_{i|i-1} = E[g_{t_i}(\theta_i) + n_i | r_{1:i-1}]$;
$P_{\mathbf{x}r_i} = E\left[(\mathbf{x}_i - \hat{\mathbf{x}}_{i|i-1})(g_{t_i}(\theta_i) + n_i - \hat{r}_{i|i-1}) | r_{1:i-1}\right]$;
$P_{r_i} = E\left[(g_{t_i}(\theta_i) + n_i - \hat{r}_{i|i-1})^2 | r_{1:i-1}\right]$;

Phase de correction;
$K_i = P_{\mathbf{x}r_i} P_{r_i}^{-1}$;
$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_{i|i-1} + K_i\left(r_i - \hat{r}_{i|i-1}\right)$;
$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

Nous nous intéressons d'abord à l'évaluation de la fonction de valeur pour une paramétrisation générique $\hat{V}_\theta$. La formulation générale espace-d'état de l'équation (4.35) se spécialise en :

$$\begin{cases} \mathbf{x}_i = F\mathbf{x}_{i-1} + v'_i \\ r_i = \hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) + n_i \end{cases} \tag{4.45}$$

où nous rappelons que $\mathbf{x}_i^T = \begin{pmatrix} \theta_i^T & \omega_i & n_i \end{pmatrix}$. Le problème est encore de calculer les statistiques d'intérêt, ce qui est fait en utilisant la transformation non-parfumée. Au temps i, les estimations $\hat{\mathbf{x}}_{i-1|i-1}$ et $P_{i-1|i-1}$ sont connues. Elles sont utilisées pour calculer les prédictions $\hat{\mathbf{x}}_{i|i-1}$ et $P_{i|i-1}$. Ces prédictions sont à leur tour utilisées pour calculer l'ensemble des sigma-points, comme décrit dans la section 3.4 (les poids associés sont aussi calculés) :

$$\mathbf{X}_{i|i-1} = \left\{ \hat{\mathbf{x}}_{i|i-1}^{(j)},\ 0 \leq j \leq 2p \right\} \tag{4.46}$$

$$\mathcal{W} = \{ w_j,\ 0 \leq j \leq 2p \} \tag{4.47}$$

Nous précisons que ces sigma-points sont la concaténation d'un vecteur de paramètres, d'une variable aléatoire auxiliaire et du bruit d'observation, soit :

$$\hat{\mathbf{x}}_{i|i-1}^{(j)} = \begin{pmatrix} \hat{\theta}_{i|i-1}^{(j)} \\ \hat{\omega}_{i|i-1}^{(j)} \\ \hat{n}_{i|i-1}^{(j)} \end{pmatrix} \tag{4.48}$$

L'image de ces sigma-points est ensuite calculée en utilisant l'équation d'observation du modèle espace d'état (4.45) :

$$\mathcal{R}_{i|i-1} = \left\{ \hat{r}_{i|i-1}^{(j)} = \hat{V}_{\hat{\theta}_{i|i-1}^{(j)}}(s_i) - \gamma \hat{V}_{\hat{\theta}_{i|i-1}^{(j)}}(s_{i+1}) + \hat{n}_{i|i-1}^{(j)},\ 0 \leq j \leq 2p+4 \right\} \tag{4.49}$$

Notons que le vecteur étendu comporte $2p+2$ composantes (p composantes pour les paramètres, une pour la variable aléatoire auxiliaire et une pour le bruit d'observation), ce qui porte le nombre de sigma-points à $2p+5$. Enfin les sigma-points et leurs images sont utilisés pour approcher les statistiques d'intérêt, qui permettront la mise à jour des moments d'ordre un et deux du vecteur étendu :

$$\hat{r}_{i|i-1} \approx \sum_{j=0}^{2p+4} w_j \hat{r}_{i|i-1}^{(j)} \tag{4.50}$$

$$P_{r_i} \approx \sum_{j=0}^{2p+4} w_j \left(\hat{r}_{i|i-1}^{(j)} - \hat{r}_{i|i-1}\right)^2 \tag{4.51}$$

$$P_{\mathbf{x}r_i} \approx \sum_{j=0}^{2p+4} w_j \left(\hat{\mathbf{x}}_{i|i-1}^{(j)} - \hat{\mathbf{x}}_{i|i-1}\right)\left(\hat{r}_{i|i-1}^{(j)} - \hat{r}_{i|i-1}\right) \tag{4.52}$$

L'algorithme XKTD-SARSA s'obtient exactement de la même façon, en considérant la chaîne de Markov valuée induite par l'espace joint état-action plutôt que la chaîne induite par l'espace d'état. Les deux approches sont résumées dans l'algorithme 4.2.

Comme nous l'avons déjà fait remarquer, l'équation d'évolution concerne maintenant des vecteurs de taille $p+2$, au lieu de p. L'équation d'observation reste scalaire. Dans la mesure où seuls deux scalaires sont ajoutés au vecteur de paramètres, la complexité de cette nouvelle famille d'algorithmes est la même que pour leurs pendants avec bruit blanc, les développements proposés dans la section 3.5.5 restent valables : la complexité, tant computationnelle que mémorielle, est en $O(p^2)$.

4.3 XKTD pondéré

Nous proposons maintenant une version pondérée de XKTD, que nous nommons wXKTD (pour *weighted eXtended Kalman Temporal Differences*). Cette variante de XKTD a été tout d'abord pensée pour traiter l'apprentissage *off-policy*, qui pose problème en raison de l'aspect "mémoriel" du bruit d'observation coloré. Finalement elle ne permet pas de résoudre cette difficulté en particulier, pour une raison que nous expliquons, mais elle s'avère utile dans le cas de politiques aléatoires et non-stationnaires, comme nous l'illustrons section 4.5 dans le cas d'une politique ϵ-gloutonne.

Nous rappelons qu'un apprentissage est dit *off-policy* si une politique est apprise (la politique cible) alors qu'une autre politique est suivie (la politique comportementale). KTD-Q (et plus généralement les algorithmes du type itération de la valeur) est un algorithme *off-policy*, dans la mesure où la politique cible est la politique optimale alors que la politique comportementale peut être n'importe quelle politique suffisamment exploratrice. De plus, l'apprentissage *off-policy* a un intérêt plus général que ce cas particulier, par exemple pour réutiliser des trajectoires déjà observées où encore si la politique comportementale n'est pas contrôlable (apprentissage par observation).

Utiliser un bruit d'observation coloré résulte en un effet mémoire, l'effet étant très similaire à ce qu'il se passe pour les traces d'éligibilité [105]. Nous avons d'ailleurs détaillé section 2.2.2 l'incompatibilité entre apprentissage *off-policy* et concept des traces d'éligibilité (sans modification des algorithmes du moins). Nous présentons de façon moins intuitive cet effet mémoire ainsi que le lien aux traces d'éligibilité en nous appuyant sur les équations

Algorithme 4.2 : XKTD-V et XKTD-SARSA

Initialisation : *a priori* $\hat{\mathbf{x}}_{0|0} = \begin{pmatrix} \hat{\theta}_{0|0}^T & 0 & 0 \end{pmatrix}^T$ et $P_{0|0}$;

pour $i \leftarrow 1, 2, \ldots$ **faire**

Observer la transition $t_i = \begin{cases} (s_i, s_{i+1}) \text{ (XKTD-V)} \\ (s_i, a_i, s_{i+1}, a_{i+1}) \text{ (XKTD-SARSA)} \end{cases}$ ainsi que la récompense associée r_i ;

Phase de prédiction;

$\hat{\mathbf{x}}_{i|i-1} = F\hat{\mathbf{x}}_{i-1|i-1}$;

$P_{i|i-1} = FP_{i-1|i-1}F^T + P_{v'_{i-1}}$;

Calcul des sigma-points ;

$\mathbf{X}_{i|i-1} = \left\{ \hat{\mathbf{x}}_{i|i-1}^{(j)}, \quad 0 \leq j \leq 2p+4 \right\}$ (en utilisant $\hat{\mathbf{x}}_{i|i-1}$ et $P_{i|i-1}$);

$\mathcal{W} = \{w_j, \quad 0 \leq j \leq 2p+4 \quad \}$;

`/* notons que` $(\hat{\mathbf{x}}_{i|i-1}^{(j)})^T = \begin{pmatrix} (\hat{\theta}_{i|i-1}^{(j)})^T & \hat{\omega}_{i|i-1}^{(j)} & \hat{n}_{i|i-1}^{(j)} \end{pmatrix}$ `*/`

$\mathcal{R}_{i|i-1} =$

$$\begin{cases} \text{(XKTD-V)} \\ \left\{ \hat{r}_{i|i-1}^{(j)} = \hat{V}_{\hat{\theta}_{i|i-1}^{(j)}}(s_i) - \gamma \hat{V}_{\hat{\theta}_{i|i-1}^{(j)}}(s_{i+1}) + \hat{n}_{i|i-1}^{(j)}, 0 \leq j \leq 2p+4 \right\} \\ \text{(XKTD-SARSA)} \\ \left\{ \hat{r}_{i|i-1}^{(j)} = \hat{Q}_{\hat{\theta}_{i|i-1}^{(j)}}(s_i, a_i) - \gamma \hat{Q}_{\hat{\theta}_{i|i-1}^{(j)}}(s_{i+1}, a_{i+1}) + \hat{n}_{i|i-1}^{(j)}, 0 \leq j \leq 2p+4 \right\} \end{cases} ;$$

Calcul des statistiques d'intérêt;

$\hat{r}_{i|i-1} = \sum_{j=0}^{2p+4} w_j \hat{r}_{i|i-1}^{(j)}$;

$P_{\mathbf{x}r_i} = \sum_{j=0}^{2p+4} w_j (\hat{\mathbf{x}}_{i|i-1}^{(j)} - \hat{\mathbf{x}}_{i|i-1})(\hat{r}_{i|i-1}^{(j)} - \hat{r}_{i|i-1})$;

$P_{r_i} = \sum_{j=0}^{2p+4} w_j \left(\hat{r}_{i|i-1}^{(j)} - \hat{r}_{i|i-1} \right)^2$;

Phase de correction;

$K_i = P_{\mathbf{x}r_i} P_{r_i}^{-1}$;

$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_{i|i-1} + K_i \left(r_i - \hat{r}_{i|i-1} \right)$;

$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

de Kalman propres à XKTD. D'après l'équation de prédiction, nous avons :

$$\begin{aligned} \hat{\mathbf{x}}_{i|i-1} &= F\hat{\mathbf{x}}_{i-1|i-1} \\ \Leftrightarrow \begin{pmatrix} \hat{\theta}_{i|i-1} \\ \hat{\omega}_{i|i-1} \\ \hat{n}_{i|i-1} \end{pmatrix} &= \begin{pmatrix} \hat{\theta}_{i-1|i-1} \\ 0 \\ \hat{\omega}_{i-1|i-1} \end{pmatrix} \end{aligned} \tag{4.53}$$

D'après l'équation de correction nous avons :

$$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_{i|i-1} + K_i\tilde{r}_i \tag{4.54}$$

où nous rappelons que K_i est le gain de Kalman et $\tilde{r}_i = r_i - \hat{r}_{i|i-1}$ est l'innovation. Nous adoptons la notation suivante :

$$\hat{g}_{t_i} = E[g_{t_i}(\theta_i)|r_{1:i-1}] \tag{4.55}$$

Nous pouvons ainsi réécrire la prédiction de récompense, en utilisant l'équation de prédiction pour la dernière étape :

$$\begin{aligned} \hat{r}_{i|i-1} &= E[g_{t_i}(\theta_i) + n_i|r_{1:i-1}] \\ &= \hat{g}_{t_i} + \hat{n}_{i|i-1} \\ &= \hat{g}_{t_i} + \hat{\omega}_{i-1|i-1} \end{aligned} \tag{4.56}$$

Enfin, nous adoptons la notation suivante, par blocs, pour le gain de Kalman :

$$K_i = \begin{pmatrix} K_{\theta_i} \\ K_{\omega_i} \\ K_{n_i} \end{pmatrix} \tag{4.57}$$

ce qui permet de réécrire l'équation de correction des paramètres moyens par bloc :

$$\begin{pmatrix} \hat{\theta}_{i|i} \\ \hat{\omega}_{i|i} \\ \hat{n}_{i|i} \end{pmatrix} = \begin{pmatrix} \hat{\theta}_{i-1|i-1} \\ 0 \\ \hat{\omega}_{i-1|i-1} \end{pmatrix} + \begin{pmatrix} K_{\theta_i} \\ K_{\omega_i} \\ K_{n_i} \end{pmatrix} \left(r_i - \hat{g}_{t_i} - \hat{\omega}_{i-1|i-1}\right) \tag{4.58}$$

De cette dernière équation il est possible de déduire une forme générale de mise à jour des paramètres :

$$\hat{\theta}_{i|i} = \hat{\theta}_{i-1|i-1} + K_{\theta_i}\left(r_i - \hat{g}_{t_i} - K_{w_{i-1}}\tilde{r}_{i-1}\right) \tag{4.59}$$

Ainsi les paramètres sont corrigés en fonction de l'innovation, cette innovation étant définie comme la somme de l'erreur de différence temporelle locale (le terme $r_i - \hat{g}_{t_i}$) et d'un terme proportionnel à l'innovation au temps précédent. L'innovation au temps i est donc une combinaison des erreurs de différences temporelles effectuées tout le long de la trajectoire, d'où l'effet mémoriel d'une part et l'analogie aux traces d'éligibilité d'autre part. Il est difficile d'aller plus loin dans ce cadre très général. Cependant, dans l'analyse de convergence de la section 4.4 nous nous plaçons dans un cadre plus restreint, à savoir paramétrisation linéaire et pas de bruit d'évolution. Nous montrons alors que l'algorithme résultant converge vers la solution de LSTD(1). Cela renforce encore le parallèle avec les traces d'éligibilité et XKTD peut approximativement être vu comme un algorithme dont la trace vaut 1 (ce qui est connu pour ne pas être le cas idéal).

Comme pour les algorithmes à base de traces d'éligibilité plus classiques, XKTD appliqué dans le cadre de l'apprentissage *off-policy* d'une politique est fortement susceptible de faillir, étant donné qu'il inclut les effets des transitions effectuées sur l'ensemble de la trajectoire, qui sont contaminées par la politique comportementale, cet effet n'étant compensé d'aucune façon. Cependant, nous proposons de prendre ces effets en compte dans le modèle du bruit. Considérons l'extension du bruit (4.19) à la Q-fonction :

$$N(s_i, a_i, s_{i+1}, a_{i+1}) = \Delta Q(s_i, a_i) - \gamma \Delta Q(s_{i+1}, a_{i+1}) \tag{4.60}$$

Soit $(s_i, a_i, s_{i+1}, a_{i+1}, s_{i+2}, a_{i+2})$ une portion de trajectoire générée en suivant une politique comportementale b. Supposons que l'on souhaite apprendre une politique π donnée, la mise à jour doit se faire en accord avec $(s_i, a_i, s_{i+1}, a^\pi_{i+1})$ où l'action a^π_{i+1} est choisie en accord avec la politique π. Notons que ce type de mise à jour *off-policy* est valable pour KTD (avec un MDP aux transitions déterministes), grâce au caractère local des mises à jour associées. La variance du bruit est toujours la même, néanmoins la corrélation entre les bruits à deux instants consécutifs devient :

$$\begin{aligned} &E[N(s_i, a_i, s_{i+1}, a^\pi_{i+1}) N(s_{i+1}, a_{i+1}, s_{i+2}, a^\pi_{i+2})] \\ &\quad = -\gamma \pi(a_{i+1}|s_{i+1}) E[(\Delta Q(s_{i+1}, a_{i+1}))^2] \end{aligned} \tag{4.61}$$

La corrélation est donc pondérée par la probabilité que l'action a_{i+1} soit choisie par la politique π. Cela mène à une légère modification de XKTD. La variance $P_{u'_i}$ (4.34) du bruit u'_i du modèle auto-régressif vectoriel devient :

$$P_{u'_i} = \sigma_i^2 \begin{pmatrix} 1 & -\gamma\pi(a_{i+1}|s_{i+1}) \\ -\gamma\pi(a_{i+1}|s_{i+1}) & \gamma^2 \end{pmatrix} \tag{4.62}$$

Notons que dans le cas de l'apprentissage *on-policy* d'une politique déterministe, l'algorithme XKTD est obtenu (la probabilité valant un).

L'algorithme wXKTD partage des similitudes avec le Q(λ) de Watkins [105] (qui coupe la trace d'éligibilité dès qu'une action non-gloutonne est prise) et plus encore avec l'algorithme *tree backup* [86] qui pondère la trace d'éligibilité par la probabilité que l'action (échantillonnée par la politique comportementale) ait été choisie par la politique cible, tout comme l'approche proposée. C'est un argument en faveur de wXKTD. Cependant, pour ces deux dernières approches, si les traces sont coupées, les mises à jour sont faites en accord avec les algorithmes Q(0) et SARSA(0) respectivement, qui sont des estimateurs non biaisés de la fonction de qualité. D'un autre côté, si le bruit est coupé[6] dans wXKTD, alors la mise à jour se fera selon KTD, qui est un estimateur biaisé de la Q-fonction lorsque les transitions sont stochastiques. Ainsi cette approche peut éventuellement réduire le biais causé par l'aspect *off-policy*, mais est peu susceptible de le supprimer totalement (deux types de biais sont en fait combinés, biais causé par l'aspect *off-policy* et biais causé par la stochasticité des transitions).

Cependant, dans le cas *on-policy*, si la politique suivie est stochastique et non-stationnaire (par exemple une politique ϵ-greedy dans un contexte d'itération optimiste de la politique), l'approche pondérée proposée est toujours valable et peut apporter des améliorations par rapport à XKTD. Tous ces aspects (inadéquation de (w)XKTD au cas *off-policy* et intérêt

[6] Si la probabilité qu'une action soit échantillonnée par la politique cible est nulle, le bruit d'observation devient localement blanc, ce que nous appelons couper le bruit.

de wXKTD par rapport à XKTD dans le cas *on-policy* si la politique est stochastique) sont expérimentés et illustrés dans la section 4.5. En perspective, d'autres types de pondérations pourraient être envisagées. L'algorithme wXKTD peut être vu comme l'introduction d'une nouvelle forme de bruit coloré.

4.4 Analyse de convergence

Dans cette section, nous montrons que l'estimateur de XKTD minimise un coût quadratique pondéré qui associe à chaque valeur une estimation par Monte Carlo du cumul pondéré de récompenses. Nous en proposons deux preuves. La première est la plus générale, elle ne fait d'hypothèse ni sur le bruit d'évolution, ni sur la linéarité de la paramétrisation. Par contre, elle repose sur une conjoncture que nous introduisons d'abord et qui postule que si deux modèles espace-d'état sont équivalents et que l'estimateur de l'un est l'estimateur du maximum *a posteriori*, alors l'estimateur de l'autre l'est également, relativement à son propre modèle espace-d'état. La seconde preuve que nous proposons fournit le même résultat, sans utiliser cette conjoncture, mais avec des hypothèses restrictives supplémentaires (linéarité de la paramétrisation, pas de bruit d'évolution). Toute l'analyse théorique de cette section est faite dans le cadre de l'évaluation de la fonction de valeur, l'extension à l'évaluation de la fonction de qualité étant évidente (pour les raisons déjà énoncées, nous ne considérons pas l'extension à l'optimisation directe de la Q-fonction).

Conjecture 4.1 (Equivalence de modèles espace-d'état et estimateurs MAP). *Soit le modèle espace-d'état pour XKTD, pour lequel le bruit d'observation n_i est coloré (4.22) et dont l'équation d'évolution est fonction des paramètres :*

$$\begin{cases} \theta_i = \theta_{i-1} + v_i \\ r_i = \hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) + n_i \end{cases} \tag{4.63}$$

Soit également le modèle espace-d'état étendu équivalent (4.35) :

$$\begin{cases} \mathbf{x}_i = F\mathbf{x}_{i-1} + v'_i \\ r_i = g_{t_i}(\mathbf{x}_i) \end{cases} \tag{4.64}$$

Si $\hat{\mathbf{x}}_{i|i}$ est l'estimateur du maximum a posteriori relativement au modèle espace-d'état (4.64), alors nous conjecturons que $\hat{\theta}_{i|i}$ est l'estimateur du maximum a posteriori relativement au modèle espace-d'état (4.63), ces derniers étant équivalents.

Cette conjecture étant posée, nous pouvons proposer notre résultat le plus général concernant les différences temporelles de Kalman étendues.

Théorème 4.2 (Convergence de XKTD). *Supposons que les distributions a posteriori et des bruits soit gaussiennes et que l'a priori soit également gaussien, de moyenne θ_0 et variance P_0. Alors l'estimateur de XKTD minimise l'erreur quadratique, pondérée et régularisée, liant les valeurs des états aux retours observés de Monte Carlo :*

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmin}} \left(\sum_{j=1}^{i} \frac{1}{\sigma_{j-1}^2} \left(\hat{V}_\theta(s_j) - \sum_{t=j}^{i} \gamma^{t-j} r_t \right)^2 + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \right) \tag{4.65}$$

Démonstration. Nous utilisons à nouveau le résultat de [116, chapitre 4.5] (voir preuve du théorème 3.1). La preuve de ce résultat est faite dans [116] pour un modèle de marche aléatoire (c'est-à-dire que la matrice d'évolution est l'identité), cependant elle peut être facilement étendue à un modèle d'évolution linéaire, ce que nous ne faisons pas ici, afin d'alléger le propos. Ce résultat peut donc être appliqué à XKTD dans sa formulation espace-d'état (4.64) :

$$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_i^{\text{MAP}} = \underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{x}|r_{1:i}) \tag{4.66}$$

Les modèles espace-d'état (4.64) et (4.63) étant équivalents, d'après la conjoncture que nous avons posée l'estimateur de XKTD dans sa formulation espace-d'état (4.63) est également l'estimateur du maximum *a posteriori* :

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmax}}\, p(\theta|r_{1:i}) = \underset{\theta}{\operatorname{argmin}}(-\ln(p(\theta|r_{1:i}))) \tag{4.67}$$

En appliquant la règle de Bayes, la distribution *a posteriori* $p(\theta|r_{1:i})$ peut être exprimée comme le produit (normalisé) de la vraisemblance $p(r_{1:i}|\theta)$ par la distribution *a priori* $p(\theta)$:

$$p(\theta|r_{1:i}) = \frac{p(r_{1:i}|\theta)p(\theta)}{p(r_{1:i})} \tag{4.68}$$

Le facteur de normalisation $p(r_{1:i})$ ne dépend pas des paramètres, le maximum *a posteriori* se réduit donc au produit de la vraisemblance et de l'*a priori* :

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmax}}\, p(r_{1:i}|\theta)p(\theta) \tag{4.69}$$

Cependant, le bruit d'observation n'étant plus blanc, il n'est plus possible de simplifier la vraisemblance jointe comme cela a été fait dans la preuve du théorème 3.1. Avant de poursuivre, nous introduisons quelques notations. Nous écrivons $V_i(\theta)$, R_i et N_i les vecteurs de taille $i \times 1$ représentant respectivement l'historique des valeurs, des récompenses et des bruits :

$$V_i(\theta) = \begin{pmatrix} \hat{V}_\theta(s_1) & \hat{V}_\theta(s_2) & \dots & \hat{V}_\theta(s_i) \end{pmatrix}^T \tag{4.70}$$

$$R_i = \begin{pmatrix} r_1 & r_2 & \dots & r_i \end{pmatrix}^T \tag{4.71}$$

$$N_i = \begin{pmatrix} n_1 & n_2 & \dots & n_i \end{pmatrix}^T \tag{4.72}$$

Nous notons $\mathbf{H}_i$ la matrice bi-diagonale de taille $i \times i$ définie par :

$$\mathbf{H}_i = \begin{pmatrix} 1 & -\gamma & 0 & \dots \\ 0 & 1 & -\gamma & 0 \\ \vdots & \ddots & \ddots & -\gamma \\ 0 & \dots & 0 & 1 \end{pmatrix} \tag{4.73}$$

Il peut être vérifié assez facilement que sont inverse est donnée par :

$$\mathbf{H}_i^{-1} = \begin{pmatrix} 1 & \gamma & \dots & \gamma^{i-1} \\ 0 & 1 & \gamma & \dots \\ \vdots & \ddots & \ddots & \gamma \\ 0 & \dots & 0 & 1 \end{pmatrix} \tag{4.74}$$

Enfin, nous définissons $\Sigma_{N_i} = E[N_i N_i^T]$ la matrice de variance du bruit N_i, qui rend donc compte de la coloration du bruit d'observation. Etant donnée la définition de n_i (4.19), elle est tri-diagonale et donnée par :

$$\Sigma_{N_i} = \begin{pmatrix} \sigma_0^2 + \gamma^2\sigma_1^2 & -\gamma\sigma_1^2 & 0 & \dots \\ -\gamma\sigma_1^2 & \sigma_1 + \gamma^2\sigma_2^2 & -\gamma\sigma_2^2 & \vdots \\ \vdots & \ddots & \ddots & -\gamma\sigma_{i-1}^2 \\ 0 & \dots & -\gamma\sigma_{i-1}^2 & \sigma_{i-1}^2 + \gamma^2\sigma_i^2 \end{pmatrix} \tag{4.75}$$

Le bruit étant supposé gaussien, la vraisemblance l'est donc également. De plus elle est colorée en raison du bruit d'observation. Sa loi est donnée par :

$$r_{1:i}|\theta \sim \mathcal{N}(R_i - \mathbf{H}_i V_i(\theta), \Sigma_{N_i}) \tag{4.76}$$

A l'*a priori* près, l'estimateur de XKTD minimise l'opposé du logarithme de la vraisemblance, donc étant donnée sa distribution (4.76), nous vérifions :

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmin}} \left((R_i - \mathbf{H}_i V_i(\theta))^T \Sigma_{N_i}^{-1} (R_i - \mathbf{H}_i V_i(\theta) + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0)) \right) \tag{4.77}$$

Or la variance du bruit joint coloré peut s'écrire en fonction de la matrice $\mathbf{H}_i$ et d'une matrice diagonale dont les éléments sont les variances des résidus :

$$\Sigma_{N_i} = \mathbf{H}_i \Sigma_i \mathbf{H}_i^T \text{ avec } \Sigma_i = \operatorname{diag}(\sigma_0^2, \dots, \sigma_{i-1}^2) \tag{4.78}$$

Le coût que minimise l'estimateur de XKTD peut donc être réécrit en utilisant cette dernière égalité :

$$\begin{aligned} \hat{\theta}_{i|i} &= \underset{\theta}{\operatorname{argmin}} \left((R_i - \mathbf{H}_i V_i(\theta))^T \Sigma_{N_i}^{-1} (R_i - \mathbf{H}_i V_i(\theta)) + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \right) \\ &= \underset{\theta}{\operatorname{argmin}} \left((R_i - \mathbf{H}_i V_i(\theta))^T (\mathbf{H}_i \Sigma_i \mathbf{H}_i^T)^{-1} (R_i - \mathbf{H}_i V_i(\theta)) + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \right) \\ &= \underset{\theta}{\operatorname{argmin}} \left((\mathbf{H}_i^{-1} R_i - V_i(\theta))^T \Sigma_i^{-1} (\mathbf{H}_i^{-1} R_i - V_i(\theta)) + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \right) \end{aligned} \tag{4.79}$$

Etant donnée la forme de l'inverse (4.74) de la matrice $\mathbf{H}_i$, cette dernière équation prouve le résultat. □

Ce résultat est très général, cependant il repose sur une conjecture que nous ne savons pas montrer. Nous proposons donc également une preuve alternative de ce même résultat, dans un cas plus restrictif.

Théorème 4.3 (Convergence de XKTD, résultat alternatif). *Si le bruit d'évolution associé aux paramètres est nul (c'est-à-dire le bruit non-étendu, autrement dit $P_{v_i} = 0$), si le bruit d'observation est gaussien, si l'a priori est gaussien de moyenne θ_0 et variance P_0 et si la paramétrisation est linéaire, alors XKTD converge vers la même solution que LSTD(1) (dans sa version récursive), c'est-à-dire :*

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmin}} \left(\sum_{j=1}^{i} \frac{1}{\sigma_{j-1}^2} \left(\theta^T \phi(s_j) - \sum_{t=j}^{i} \gamma^{t-j} r_t \right)^2 + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \right) \tag{4.80}$$

Démonstration. L'idée de cette preuve est de montrer que XKTD-V est en fait le même algorithme qu'un autre appelé MC-GPTD paramétrique, obtenu en utilisant le même modèle de bruit, des processus Gaussiens et des arguments bayésiens et qui converge vers la solution de LSTD(1) [27, chapitre 4.4.-4.5.], à savoir la minimisation du coût (4.80). L'algorithme MC-GPTD paramétrique est donné par les équations suivantes :

$$\begin{cases} \hat{\theta}_{i|i} = \hat{\theta}_{i|i} + \frac{P_{\theta r_i}}{P_{r_i}} \tilde{r}_i \\ P_{\theta_{i|i}} = P_{\theta_{i-1|i-1}} - \frac{P_{\theta r_i} P_{\theta r_i}^T}{P_{r_i}} \\ \tilde{r}_i = r_i - H_i^T \hat{\theta}_{i-1|i-1} + \frac{\gamma \sigma_{i-1}^2}{P_{r_{i-1}}} \tilde{r}_{i-1} \\ P_{\theta r_i} = P_{\theta_{i-1|i-1}} H_i + \frac{\gamma \sigma_{i-1}^2}{P_{r_{i-1}}} P_{\theta r_{i-1}} \\ P_{r_i} = \left(P_{\theta r_i} - \frac{\gamma \sigma_{i-1}^2}{P_{r_i}} \right)^T H_i + \sigma_{i-1}^2 + \gamma^2 \sigma_i^2 - \gamma^2 \frac{\sigma_{i-1}^4}{P_{r_i}} \end{cases} \tag{4.81}$$

Il s'agit donc de montrer que XKTD, sous les hypothèses faites, se réduit aux mêmes équations. Les deux premières sont classiques (dans le contexte de Kalman) et déjà vérifiées, ce sont les trois dernières qui demandent plus de travail.

Rappelons les définitions de $\mathbf{x}_i$ (4.36), F (4.37), H_i (3.23) et $P_{v_i'}$ (4.38). Par hypothèse le bruit d'évolution est nul, soit $P_{v_i} = 0$. La fonction de valeur est paramétrée linéairement (2.35). Le modèle espace d'état (4.35) peut donc se réécrire :

$$\begin{cases} \mathbf{x}_i = F \mathbf{x}_{i-1} + v_i' \\ r_i = \begin{pmatrix} H_i^T & 0 & 1 \end{pmatrix} \mathbf{x}_i \end{cases} \tag{4.82}$$

Pour montrer l'équivalence des deux approches (XKTD-V et MC-GPTD paramétrique), les équations de Kalman de l'algorithme 4.1 sont calculées analytiquement. Nous considérons d'abord les équations de prédiction. La prédiction des paramètres moyens peut se calculer par blocs :

$$\begin{aligned} \hat{\mathbf{x}}_{i|i-1} = F \hat{\mathbf{x}}_{i-1|i-1} &\Leftrightarrow \begin{pmatrix} \hat{\theta}_{i|i-1} \\ \hat{\omega}_{i|i-1} \\ \hat{n}_{i|i-1} \end{pmatrix} = \begin{pmatrix} I_p & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & 0 & 0 \\ \mathbf{0}^T & 1 & 0 \end{pmatrix} \begin{pmatrix} \hat{\theta}_{i-1|i-1} \\ \hat{\omega}_{i-1|i-1} \\ \hat{n}_{i-1|i-1} \end{pmatrix} \\ &\Leftrightarrow \begin{pmatrix} \hat{\theta}_{i|i-1} \\ \hat{\omega}_{i|i-1} \\ \hat{n}_{i|i-1} \end{pmatrix} = \begin{pmatrix} \hat{\theta}_{i-1|i-1} \\ 0 \\ \hat{\omega}_{i-1|i-1} \end{pmatrix} \end{aligned} \tag{4.83}$$

La prédiction de la matrice de variance associée peut également être calculée analytiquement :

$$\begin{aligned} & P_{i|i-1} = F P_{i-1|i-1} F^T + P_{v_i} \\ \Leftrightarrow & \begin{pmatrix} P_{\theta_{i|i-1}} & P_{\theta\omega_{i|i-1}} & P_{\theta n_{i|i-1}} \\ P_{\theta\omega_{i|i-1}}^T & P_{\omega_{i|i-1}} & P_{n_{i|i-1}} \\ P_{\theta n_{i|i-1}}^T & P_{\omega n_{i|i-1}} & P_{n_{i|i-1}} \end{pmatrix} = F P_{i|i-1} F^T + \begin{pmatrix} 0 I_p & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \sigma_i^2 & -\gamma\sigma_i^2 \\ \mathbf{0}^T & -\gamma\sigma_i^2 & \gamma^2\sigma_i^2 \end{pmatrix} \\ \Leftrightarrow & \begin{pmatrix} P_{\theta_{i|i-1}} & P_{\theta\omega_{i|i-1}} & P_{\theta n_{i|i-1}} \\ P_{\theta\omega_{i|i-1}}^T & P_{\omega_{i|i-1}} & P_{n_{i|i-1}} \\ P_{\theta n_{i|i-1}}^T & P_{\omega n_{i|i-1}} & P_{n_{i|i-1}} \end{pmatrix} = \begin{pmatrix} P_{\theta_{i-1|i-1}} & \mathbf{0} & P_{\theta\omega_{i-1|i-1}} \\ \mathbf{0}^T & \sigma_i^2 & -\gamma\sigma_i^2 \\ P_{\theta\omega_{i-1|i-1}}^T & -\gamma\sigma_i^2 & P_{\omega_{i-1|i-1}} + \gamma^2\sigma_i^2 \end{pmatrix} \end{aligned} \tag{4.84}$$

L'étape suivante consiste à calculer les statistiques d'intérêt. C'est fait de façon très similaire à la section 3.3. La première quantité à calculer est la prédiction de la récompense :

$$\begin{aligned}\hat{r}_{i|i-1} &= \begin{pmatrix} H_i^T & 0 & 1 \end{pmatrix} \hat{\mathbf{x}}_{i|i-1} = H_i^T \hat{\theta}_{i|i-1} + \hat{n}_{i|i-1} \\ &= H_i^T \hat{\theta}_{i-1|i-1} + \hat{\omega}_{i-1|i-1} \end{aligned} \tag{4.85}$$

La covariance entre les paramètres et les récompenses doit également être calculée, la matrice $P_{i|i-1}$ étant définie dans l'équation (4.84) :

$$P_{\mathbf{x}r_i} = P_{i|i-1} \begin{pmatrix} H_i^T & 0 & 1 \end{pmatrix}^T \Leftrightarrow \begin{pmatrix} P_{\theta r_i} \\ P_{\omega r_i} \\ P_{n r_i} \end{pmatrix} = \begin{pmatrix} P_{\theta_{i-1|i-1}} H_i + P_{\theta\omega_{i-1|i-1}} \\ -\gamma\sigma_i^2 \\ P_{\theta\omega_{i-1|i-1}}^T H_i + P_{\omega_{i-1|i-1}} + \gamma^2\sigma_i^2 \end{pmatrix} \tag{4.86}$$

La dernière statistique à calculer est la variance sur l'innovation :

$$\begin{aligned} P_{r_i} &= \begin{pmatrix} H_i^T & 0 & 1 \end{pmatrix} P_{i-1|i-1} \begin{pmatrix} H_i^T & 0 & 1 \end{pmatrix}^T = \begin{pmatrix} H_i^T & 0 & 1 \end{pmatrix} P_{\mathbf{x}r_i} \\ &= H_i^T P_{\theta_{i-1|i-1}} H_i + 2 H_i^T P_{\theta\omega_{i-1|i-1}} + P_{\omega_{i-1|i-1}} + \gamma^2\sigma_i^2 \end{aligned} \tag{4.87}$$

Pour terminer la récurrence de Kalman, la dernière chose à faire est d'exprimer les équations de correction. La gain de Kalman n'est pas facile à simplifier et nous l'écrivons de façon générique :

$$K_i = \begin{pmatrix} K_{\theta_i} \\ K_{\omega_i} \\ K_{n_i} \end{pmatrix} = \frac{1}{P_{r_i}} P_{\mathbf{x}r_i} = \frac{1}{P_{r_i}} \begin{pmatrix} P_{\theta r_i} \\ P_{\omega r_i} \\ P_{n r_i} \end{pmatrix} \tag{4.88}$$

Le vecteur de paramètres moyen est corrigé en utilisant la mise à jour linéaire maintenant classique :

$$\begin{aligned} &\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_{i|i-1} + K_i \tilde{r}_i \\ \Leftrightarrow &\begin{pmatrix} \hat{\theta}_{i|i} \\ \hat{\omega}_{i|i} \\ \hat{n}_{i|i} \end{pmatrix} = \begin{pmatrix} \hat{\theta}_{i-1|i-1} \\ 0 \\ \hat{\omega}_{i-1|i-1} \end{pmatrix} + \begin{pmatrix} K_{\theta_i} \\ K_{\omega_i} \\ K_{n_i} \end{pmatrix} \left(r_i - H_i^T \hat{\theta}_{i-1|i-1} - \hat{\omega}_{i-1|i-1} \right) \end{aligned} \tag{4.89}$$

La correction de la matrice de variance des paramètres est donnée par :

$$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T \tag{4.90}$$

Comme cette matrice est symétrique, il est suffisant d'étudier la partie triangulaire supérieure par bloc :

$$P_{i|i} = \begin{pmatrix} P_{\theta_{i|i}} & P_{\theta\omega_{i|i}} & P_{\theta n_{i|i}} \\ \cdots & P_{\omega_{i|i}} & P_{\omega n_{i|i}} \\ \cdots & \cdots & P_{n_{i|i}} \end{pmatrix}$$

$$\text{avec } \begin{cases} P_{\theta_{i|i}} = P_{\theta_{i-1|i-1}} - K_{\theta_i} P_{r_i} K_{\theta_i}^T \\ P_{\theta\omega_{i|i}} = -K_{\theta_i} P_{r_i} K_{\omega_i} \\ P_{\theta n_{i|i}} = P_{\theta\omega_{i-1|i-1}} - K_{n_i} P_{r_i} K_{\theta_i} \\ P_{\omega_{i|i}} = \sigma_i^2 - K_{\omega_i}^2 P_{r_i} \\ P_{\omega n_{i|i}} = -\gamma\sigma_i^2 - K_{\omega_i} P_{r_i} K_{n_i} \\ P_{n_{i|i}} = P_{\omega_{i-1|i-1}} + \gamma^2\sigma_i^2 \end{cases} \tag{4.91}$$

La dernière chose à faire pour montrer l'équivalence entre les deux approches est de manipuler les équations obtenues, l'objectif étant d'obtenir une récurrence qui ressemble à Kalman sur les paramètres θ plutôt qu'une récurrence de Kalman pure sur les paramètres étendus $\mathbf{x}_i$. Notons tout d'abord que, d'après les équations (4.86, 4.88, 4.91), la composante du gain de Kalman correspondant à la variable aléatoire auxiliaire peut s'écrire :

$$K_{\omega_i} = \frac{P_{\omega r_i}}{P_{r_i}} = -\frac{\gamma\sigma_i^2}{P_{r_i}} \tag{4.92}$$

Premièrement, le terme d'innovation est réécrit comme une récurrence, de façon similaire à ce qui a été fait dans la section 4.3 dans un cadre plus général. En utilisant l'équation (4.85), l'innovation s'écrit :

$$\tilde{r}_i = r_i - \hat{r}_{i|i-1} = r_i - H_i^T\hat{\theta}_{i-1|i-1} - \hat{\omega}_{i-1|i-1} \tag{4.93}$$

Cependant, d'après les équations (4.89, 4.92), la variable aléatoire auxiliaire corrigée est :

$$\hat{\omega}_{i|i} = -\frac{\gamma\sigma_i^2}{P_{r_i}}\tilde{r}_i \tag{4.94}$$

Ainsi le terme d'innovation peut s'écrire comme une récurrence :

$$\tilde{r}_i = r_i - H_i^T\hat{\theta}_{i-1|i-1} + \frac{\gamma\sigma_{i-1}^2}{P_{r_{i-1}}}\tilde{r}_{i-1} \tag{4.95}$$

Notons que le terme $r_i - H_i^T\hat{\theta}_{i-1|i-1}$ est une erreur de différence temporelle.

Deuxièmement, la statistique $P_{\theta r_i}$ peut également être définie de façon récursive. L'équation (4.86) rappelle l'expression de $P_{\theta r_i}$. D'après les équations (4.86, 4.88, 4.91, 4.92), la correction de la corrélation entre les paramètres et la variable auxiliaire est :

$$P_{\theta\omega_{i|i}} = -K_{\theta_i}P_{r_i}K_{w_i} = \frac{\gamma\sigma_i^2}{P_{r_i}}P_{\theta r_i} \tag{4.96}$$

Ainsi une récurrence est obtenue sur $P_{\theta r_i}$:

$$P_{\theta r_i} = P_{\theta_{i-1|i-1}}H_i + \frac{\gamma\sigma_{i-1}^2}{P_{r_{i-1}}}P_{\theta r_{i-1}} \tag{4.97}$$

Troisièmement, une récurrence similaire peut être obtenue pour la variance de l'innovation. L'équation (4.84) rappelle l'expression de P_{r_i}. Rappelons également l'expression de $P_{\theta\omega_{i|i}}$ (4.96). En utilisant les équations (4.86, 4.88, 4.91), il peut être écrit que :

$$P_{\omega_{i|i}} = \sigma_i^2 - \gamma^2\frac{\sigma_i^2}{P_{r_i}} \tag{4.98}$$

En utilisant toutes les équations précédemment citées, la récurrence sur la variance de l'innovation est obtenue :

$$P_{r_i} = \left(P_{\theta r_i} - \frac{\gamma\sigma_{i-1}^2}{P_{r_i}}\right)^T H_i + \sigma_{i-1}^2 + \gamma^2\sigma_i^2 - \gamma^2\frac{\sigma_{i-1}^4}{P_{r_i}} \tag{4.99}$$

Les équations (4.95, 4.97, 4.99) définissent exactement l'algorithme MC-GPTD paramétrique. Il peut être montré [27, chapitre 4.4.-4.5.] qu'il converge vers la même solution que LSTD(1), c'est à dire une régression par moindres carrés de la fonction de valeur, les cibles de cette régression étant des échantillons de Monte Carlo (statistiquement corrélés) du cumul des récompenses pondérées (résultat montré dans [18] pour LSTD(1)). Donc XKTD-V converge vers la même solution, ce qui prouve le résultat. □

Ainsi les théorèmes 4.2 et 4.3 montrent le même résultat, sous des hypothèses différentes. Le premier est basé sur une conjecture et est valable dans le cadre le plus général, le second n'utilise pas cette conjecture mais fait des hypothèses plus restrictives. Nous rappelons que l'objectif de l'introduction de ce bruit coloré était d'obtenir un estimateur non-biaisé de la fonction de valeur. Dans le cas d'une paramétrisation linéaire, XKTD-V converge vers la même solution que LSTD(1)[7]. Dans ce cas au moins, nous sommes assurés d'avoir un estimateur consistant, *id est* asymptotiquement non-biaisé, LSTD(1) produisant un estimateur non-biaisé[8] de la fonction de valeur. Nous conjecturons que ce caractère consistant de l'estimateur est toujours valable lorsqu'on lève les hypothèses d'absence de bruit d'évolution et de linéarité de la paramétrisation.

4.5 Expérimentations

Dans cette section est proposé un ensemble de benchmarks usuels en apprentissage par renforcement dont l'objectif est d'illustrer les différents aspects de (w)(X)KTD et de comparer cette famille d'algorithmes à l'état de l'art. Nous en donnons ici un aperçu :

chaîne de Boyan : cette expérience a été utilisée dans le chapitre précédent pour illustrer le biais causé par les transitions stochastiques pour KTD. Nous montrons que XKTD n'est pas biaisé, tout en étant robuste à la non-stationnarité ;

chaîne contrôlée : cet environnement peut basiquement être vu comme une version contrôlée de la chaîne de Boyan. L'objectif est ici triple : montrer l'efficacité de XKTD pour un problème d'évaluation pure de la politique, illustrer l'avantage de wXKTD dans le cas d'une politique ϵ-gloutonne et montrer le problème posé par l'apprentissage *off-policy* ;

***mountain car* stochastique :** c'est une version stochastique du problème présenté dans le chapitre précédent sur laquelle nous comparons les variantes de KTD à des algorithmes de l'état de l'art dans un schéma d'itération optimiste de la politique.

Les autres algorithmes considérés sont toujours TD, SARSA, Q-learning ainsi que LSTD et MC-GPTD. Comme avant, nous considérons peu leurs extensions aux traces d'éligibilité.

[7] La fonction de coût de LSTD(1) est généralement donnée sans terme de régularisation dans la littérature. Cependant, lorsqu'on considère la version récursive de cet algorithme, c'est bien le coût régularisé (4.80) qui est minimisé.

[8] Pour voir cela, on peut adopter un point de vue variable instrumentale : l'utilisation d'une variable instrumentale (correctement définie, c'est-à-dire corrélée avec l'entrée et décorrélée avec le bruit d'entrée) produit de façon générale un estimateur non biaisé, l'estimateur résultant est donc non-biaisé. L'introduction des traces d'éligibilité revient simplement à définir une nouvelle variable instrumentale, qui est toujours correctement définie. Notons que le point de vue variable instrumentale est très peu courant dans la littérature sur le renforcement, à notre connaissance seuls [19, 115] l'ont.

4.5.1 Chaîne de Boyan

Nous avons déjà présenté la chaîne de Boyan dans la section 3.8.1. Nous considérons ici sa version stochastique et comme précédemment pour simuler la non-stationnarité le signe de la récompense est inversé au bout du centième épisode. La paramétrisation de la fonction de valeur ainsi que la mesure de performance restent les mêmes.

Le facteur d'actualisation γ est fixé à 1 pour cette tâche épisodique. Pour TD, le taux d'apprentissage est fixé à $\alpha = 0.1$. Pour LSTD, l'*a priori* est fixé à $P_{0|0} = I$. Pour KTD-V le même *a priori* est utilisé. La variance des résidus (qui participe à la définition du bruit d'observation) est choisie constante, fixée à $\sigma_i^2 = 10^{-3}$. Rappelons la notation de l'équation (4.91), $P_{\theta_{i|i}}$ représente le bloc supérieur gauche de taille $p \times p$ de la matrice de variance $P_{i|i}$ (c'est cette matrice pour KTD). Nous choisissons comme bruit d'évolution sur les paramètres le bruit adaptatif déjà présenté dans le chapitre précédent, section 3.8, c'est-à-dire $P_{v_i} = \eta P_{\theta_{i-1|i-1}}$ avec η choisi ici égal à 10^{-2}. La variance des résidus et le bruit d'évolution sur les paramètres définissent le bruit d'évolution étendu :

$$P_{v'_i} = \begin{pmatrix} \eta P_{\theta_{i-1|i-1}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \sigma_i^2 & -\gamma\sigma_i^2 \\ \mathbf{0}^T & -\gamma\sigma_i^2 & \gamma^2\sigma_i^2 \end{pmatrix} \tag{4.100}$$

Nous rappelons que choisir ces paramètres peut nécessiter un peu de pratique, mais pas forcément plus que de choisir un taux d'apprentissage pour d'autres algorithmes. Nous considérons également l'algorithme MC-GPTD, qui dans ce cas de paramétrisation linéaire est proche de notre méthode, sans possibilité cependant de définir un bruit d'évolution. Les autres paramètres sont choisis identiques à ceux de XKTD-V. Pour toutes les approches considérées le vecteur de paramètres est initialisé à zéro. L'apprentissage se fait sur 200 épisodes et les résultats sont moyennés sur 200 essais (un essai correspondant à 200 épisodes). Les résultats sont présentés figure 4.1.

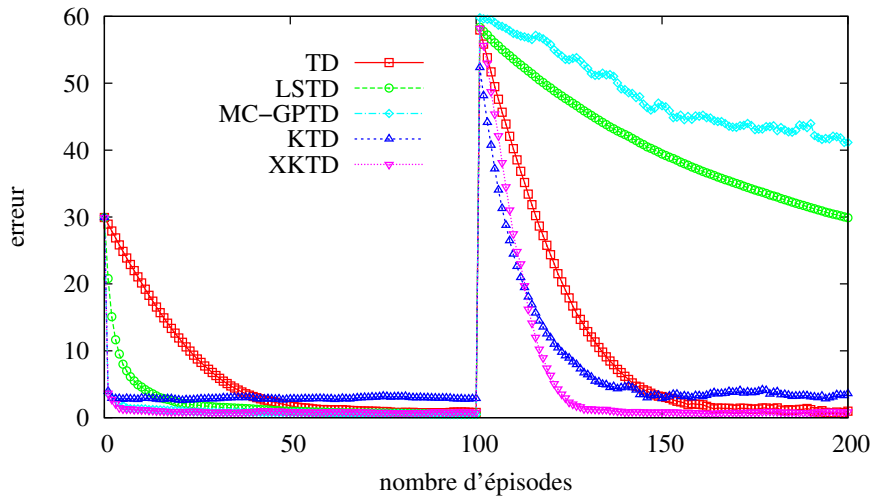


FIG. 4.1 – Chaîne de Boyan, cas stochastique et non-stationnaire.

Avant le changement de récompense, KTD converge plus vite que LSTD, qui converge lui-même plus vite que TD. Cependant, comme prévu, KTD est biaisé. L'algorithme XKTD converge aussi rapidement que KTD, cependant sans être biaisé, ce qui était l'objectif. Les

comportements de MC-GPTD et XKTD sont sensiblement les mêmes dans cette première phase. Après le changement de signe de la récompense, des résultats similaires à ceux de la section 3.8 sont obtenus. LSTD est très lent à s'adapter au nouveau MDP, MC-GPTD est encore plus lent et semble-t-il moins stable, TD s'en sort mieux, en partie grâce au taux d'apprentissage choisi constant et KTD s'adapte encore plus rapidement. XKTD s'adapte à une vitesse comparable, sans le problème de biais. Cette adaptation est toutefois légèrement plus lente, en raison de l'effet mémoire induit par l'utilisation d'un bruit coloré, effet que nous avons discuté dans la section 4.3. Il est à noter que choisir le bruit d'évolution peut être difficile. S'il y a beaucoup de bruit, le processus d'adaptation va être rapide, mais l'apprentissage sera plutôt instable, particulièrement si les transitions sont stochastiques (une transition stochastique pouvant être interprétée comme une non-stationnarité d'un MDP déterministe). Si le bruit est trop faible, l'adaptation sera lente. C'est une forme de dilemme entre plasticité et stabilité, mais il faut tout de même noter que le même genre de problème est inhérent au choix d'un taux d'apprentissage. Choisir un bruit d'évolution adaptatif comme celui que nous utilisons réduit en partie ce problème.

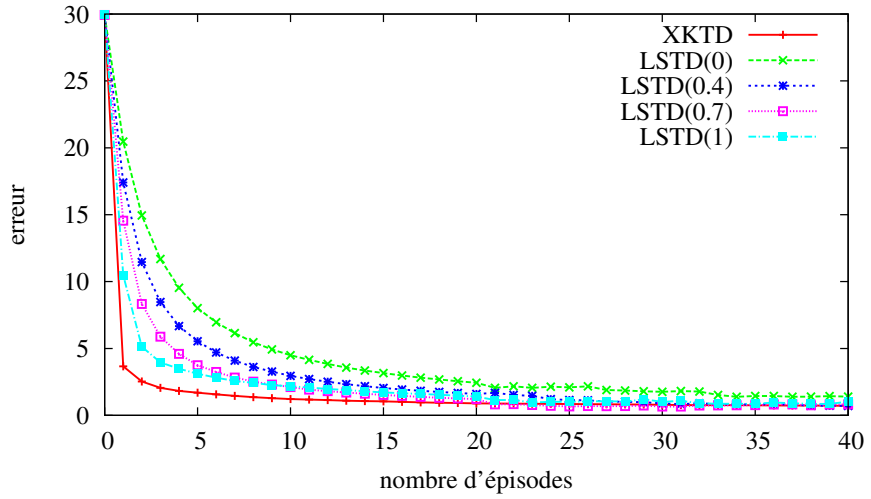


FIG. 4.2 – Chaîne de Boyan stochastique, comparaison à LSTD(λ).

Nous comparons également XKTD à LSTD(λ), pour différentes valeurs de λ, de façon similaire à ce qui a été fait dans la section expérimentale du chapitre 3. Les paramètres sont les mêmes que précédemment et les résultats correspondants sont représentés sur la figure 4.2. Le gain de performance de XKTD sur LSTD(λ) est moins marqué sur cette chaîne stochastique que ce qui avait pu être observé lors de la comparaison de KTD à LSTD(λ) sur sa version déterministe. Cependant, si augmenter λ améliore le comportement de LSTD(λ), c'est toujours XKTD qui a le meilleur comportement, tant en terme de vitesse de convergence qu'asymptotiquement. Nous ne considérons donc plus les extensions de LSTD aux traces d'éligibilité dans le reste de cette partie expérimentale.

4.5.2 Chaîne contrôlée

Le MDP correspondant à cet environnement est représenté figure 4.3. Il comporte 6 états et 2 actions. Les actions consistent à aller à gauche ou à droite. Avec une probabilité de $p = 0.9$, l'action choisie est effectuée, sinon c'est son contraire. La récompense est de -3

pour chaque état sauf pour le dernier où elle vaut -2. La politique optimale est de façon assez évidente toujours "aller à droite". Une représentation tabulaire de la Q-fonction est choisie (c'est un choix de paramétrisation parmi d'autres, totalement justifié dans ce cas). Le facteur d'actualisation est fixé à $\gamma = 0.9$. Dans toutes ces expériences la performance est mesurée avec le RMSE (*root mean-square error* pour erreur quadratique moyenne) $\|Q^* - \hat{Q}_\theta\|^2$.

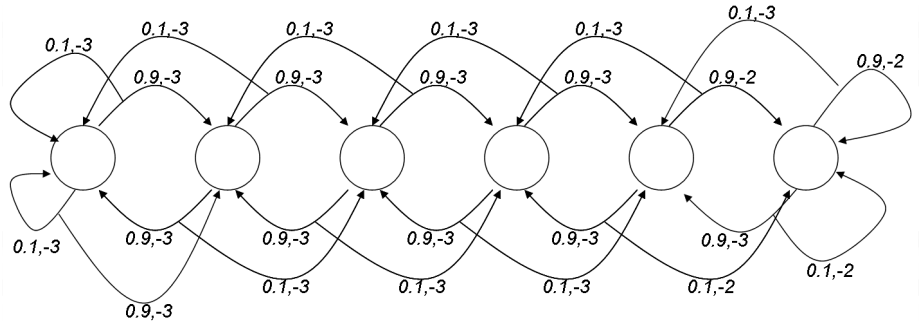


FIG. 4.3 – Chaîne contrôlée illustrée.

Evaluation de la Q-fonction

Nous nous intéressons dans un premier temps au problème de l'évaluation de la Q-fonction, pour une politique fixée. L'*a priori* pour LSTD et (X)KTD est fixé à $5I$. Pour TD, le taux d'apprentissage est choisi à $\alpha_i = \frac{n_0+1}{n_0+i}$ avec $n_0 = 500$. Il n'y a pas de bruit de process sur les paramètres et la variance du résidu est choisie à $\sigma_i^2 = 5.10^{-3}$. Le bruit d'évolution étendu a donc une matrice :

$$P_{v'_i} = \begin{pmatrix} 0I & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \sigma_i^2 & -\gamma\sigma_i^2 \\ \mathbf{0}^T & -\gamma\sigma_i^2 & \gamma^2\sigma_i^2 \end{pmatrix} \tag{4.101}$$

L'apprentissage de la politique suivie (la politique optimale est choisie) se fait sur 500 épisodes (chaque épisode étant initialisé avec une paire état-action aléatoire et limité à 15 interactions), les résultats sont présentés sur la figure 4.4. Ces résultats sont moyennés sur 100 essais et l'échelle logarithmique est à noter.

Il peut être vu que pour ce problème d'évaluation de la Q-fonction, KTD et XKTD convergent plus rapidement que TD et LSTD, XKTD présentant de plus l'avantage d'être non biaisé, comme prévu. Cela confirme les résultats de la section 4.5.1.

Itération optimiste de la politique

Le problème d'intérêt suivant est l'apprentissage de la Q-fonction optimale avec une politique de type ϵ-gloutonne, ϵ étant choisi égal à 0.1 et le domaine étant le même. Le taux d'apprentissage est le même que pour l'évaluation. L'*a priori* est fixé à I et la variance des résidus est choisie telle que $\sigma_i^2 = 10^{-3}$. Le même bruit de process que dans la section 4.5.1 est utilisé, avec $\eta = 10^{-3}$. Comme la politique ϵ-gloutonne est aléatoire, nous testons également l'algorithme wXKTD, le changement étant dépeint équation (4.62). Le facteur de corrélation $-\gamma\sigma_i^2$ est donc pondéré par $1-\epsilon$ ou ϵ, selon que l'action gloutonne est choisie

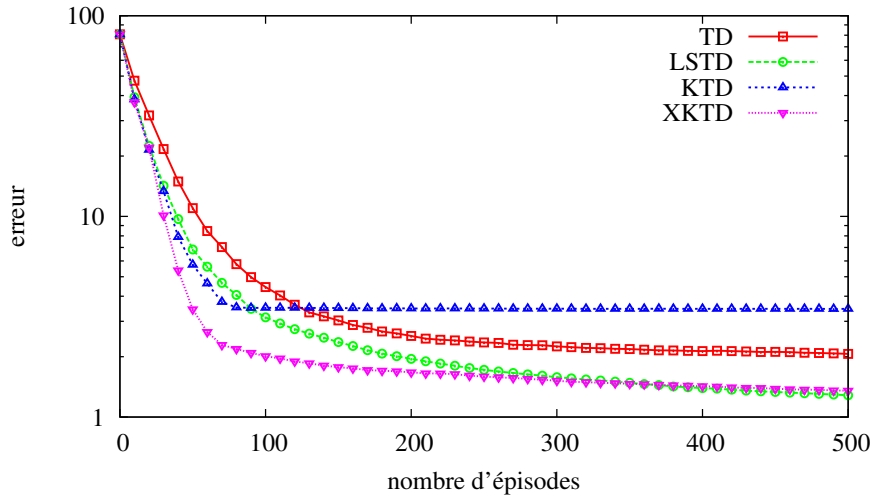


FIG. 4.4 – Chaîne contrôlée, évaluation de la Q-fonction.

ou non. L'apprentissage se fait sur 700 épisodes (chacun étant initialisé avec une paire état-action aléatoire et limité à 15 interactions) et les résultats présentés sur la figure 4.5 sont moyennés sur 100 essais (notons à nouveau l'échelle logarithmique). La performance est toujours mesurée par la distance à la politique optimale, alors que l'algorithme va chercher la politique ϵ-gloutonne optimale, l'erreur ne peut donc atteindre 0.

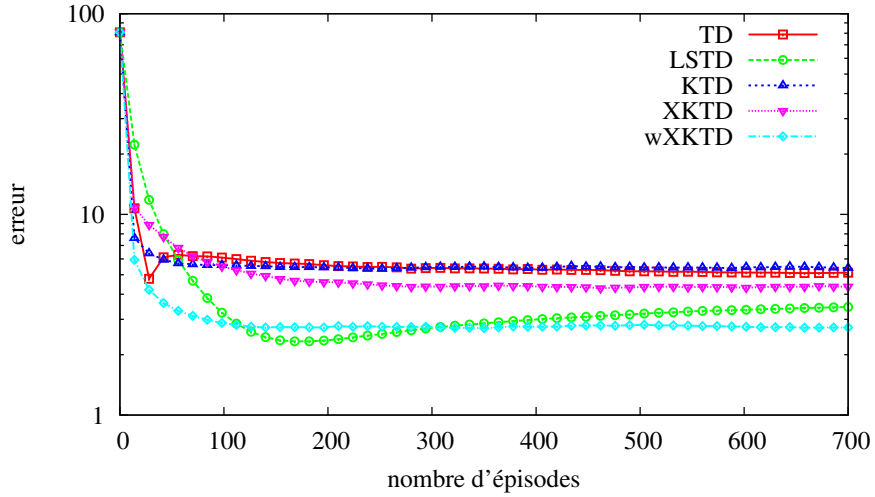


FIG. 4.5 – Chaîne contrôlée, politique ϵ-gloutonne.

KTD et XKTD convergent plus rapidement que TD et LSTD dans un premier temps. Puis KTD est coincé par le biais causé par les transitions stochastiques et XKTD converge plus lentement. LSTD converge plus rapidement, cependant il est moins stable. Après approximativement 150 épisodes, l'erreur associée augmente. Cela est probablement causé par l'imbrication de l'apprentissage et du contrôle qui induit des non-stationnarités, comme nous l'avons déjà expliqué. Les meilleurs résultats sont obtenus avec l'algorithme wXKTD. Pondérer le facteur de corrélation semble donc avoir du sens.

Optimisation directe de la Q-fonction

Dans cette section nous nous proposons d'illustrer le biais causé par l'aspect *off-policy* de l'optimisation directe de la Q-fonction, comme cela a été expliqué dans la section 4.3. Comparaison est faite à l'algorithme Q-learning. Les paramètres sont les mêmes que pour l'expérience sur le même environnement avec la politique ϵ-gloutonne, le bruit de process sur les paramètres en moins. La politique comportementale est totalement aléatoire. La mise à jour se fait en accord avec l'équation d'optimalité de Bellman (ce que serait l'algorithme XKTD-Q que nous n'avons pas explicitement introduit, mais qui est facile à déduire étant donné ce qui a déjà été présenté). Les résultats sont présentés sur la figure 4.6. L'apprentissage se fait sur 100 épisodes et les résultats sont moyennés sur 100 essais.

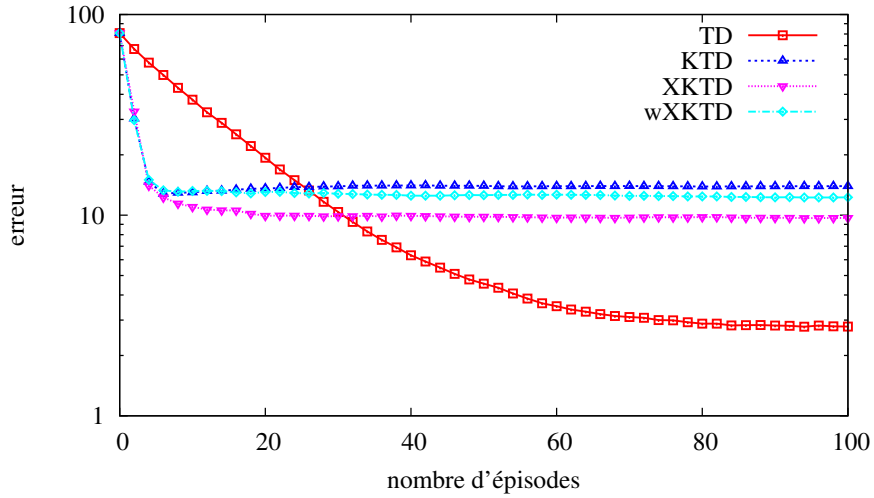


FIG. 4.6 – Chaîne contrôlée, optimisation directe de la Q-fonction.

Quatre algorithmes sont comparés, à savoir Q-learning et (w)(X)KTD-Q. Pour wXKTD, le bruit d'observation à moyenne mobile est coupé à chaque fois que l'action comportementale n'est pas l'action gloutonne. LSTD n'est pas considéré, dans la mesure où ce n'est pas un algorithme du type itération de la valeur. Comme prévu, la famille des algorithmes de type KTD converge plus rapidement, mais ils sont tous biaisés. KTD est biaisé à cause du caractère stochastique des transitions. XKTD est biaisé à cause du caractère *off-policy* de l'apprentissage. Comme prévu, wXKTD combine les deux sources de biais. Cet algorithme ne permet donc pas d'améliorer les performances, contrairement à son application dans le contexte d'une politique ϵ-gloutonne.

4.5.3 *Mountain car* stochastique

L'objectif ici est d'illustrer le comportement de (w)(X)KTD sur un problème de contrôle plus complexe. La dernière expérience que nous menons concerne donc une version stochastique du *mountain car*. Nous rappelons que dans sa forme originale (déterministe), cette tâche consiste à sortir une voiture sous-puissante d'une cuvette. L'environnement général est totalement décrit dans [105, ch.8.4] ainsi que dans la section 3.8.3. Pour rendre l'environnement stochastique ce dernier est légèrement modifié : l'action choisie est appliquée avec une probabilité $p = 0.8$ et une des autres au hasard avec une probabilité $\frac{1-p}{2} = 0.1$. Le facteur d'actualisation est fixé à $\gamma = 0.95$. Les états sont normalisés et la paramétrisation

est composée d'un terme constant et d'un ensemble de 9 noyaux gaussiens équi-répartis (centrés en $\{0, 0.5, 1\} \times \{0, 0.5, 1\}$ et d'écart-type 0.1), cela pour chaque action. Il y a donc 30 fonctions de base.

L'expérimentation compare SARSA avec approximation de la Q-fonction, LSTD et (w)(X)KTD-SARSA dans le cadre d'un schéma d'itération optimiste de la politique. La politique suivie est ϵ-gloutonne avec $\epsilon = 0.1$. Pour SARSA le facteur d'apprentissage est fixé à $\alpha_i = 0.1$. Pour les autres algorithmes, l'*a priori* est choisi $P_{0|0} = 10I$. Pour (w)(X)KTD, la variance du bruit d'observation/du résidu est fixé à 1, soit $P_{n_i} = 1$ pour KTD et $\sigma_i^2 = 1$ pour les autres. Le même bruit de process sur les paramètres que précédemment est utilisé, avec $\eta = 10^{-5}$. Chaque épisode démarre avec une position et une vitesse aléatoire, uniformément échantillonnées. Un maximum de 1500 pas par épisode est autorisé. Pour chaque essai, l'apprentissage se fait sur 200 épisodes et la figure 4.7 montre la longueur de chaque épisode moyenné sur 300 essais.

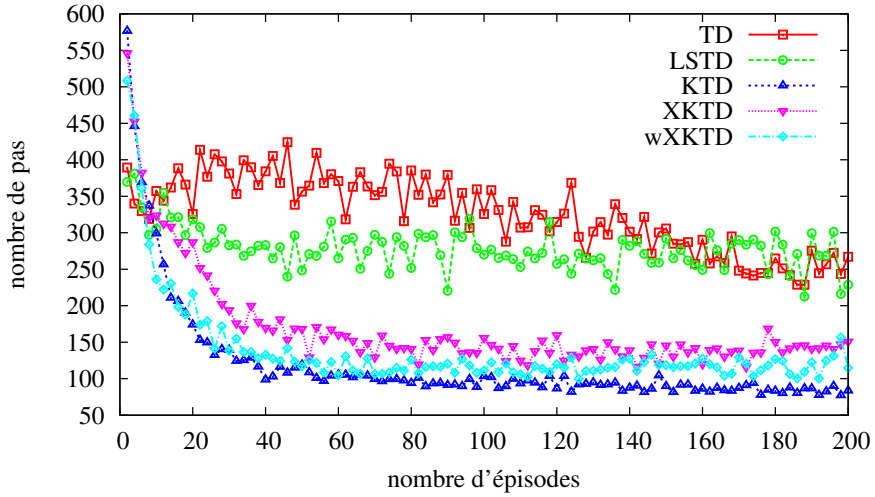


FIG. 4.7 – *Mountain car* stochastique.

Toutes les variantes de KTD convergent plus rapidement et vers une meilleure solution que TD et LSTD. Comme pour la section 4.5.2, pour laquelle le contexte d'imbrication contrôle/apprentissage est le même, de meilleurs résultats sont obtenus avec wXKTD qu'avec XKTD. Cependant, de façon assez surprenante, c'est KTD qui fonctionne le mieux, malgré le problème théorique posé par les transitions stochastiques. On peut donc légitimement se demander dans quelle mesure le biais pose problème pour la convergence vers la politique de contrôle optimale, qui est finalement ce qui nous intéresse. Pour ce problème en particulier, il est possible qu'il ne soit pas assez stochastique, dans le sens où le biais induit sur l'estimateur ne modifie pas l'action de plus grande valeur pour chaque état (par exemple si le biais dépend de l'état, mais pas de l'action). Si c'est le cas et en faisant un parallèle entre (X)KTD et des traces d'éligibilité à zéro ou un, l'avantage de KTD peut être intuité. Il faut aussi noter que la mesure de performance porte sur la politique même et non pas sur la fonction de valeur ou de qualité associée comme c'était le cas pour les expériences précédentes. Une façon d'expliquer les meilleurs résultats de KTD pourrait être qu'une évaluation exacte de la fonction de valeur n'est pas forcément nécessaire pour obtenir une bonne politique de contrôle (et inversement). En ce qui concerne wXKTD, le bruit est quasi-coupé à chaque fois qu'une action non-gloutonne est choisie, ce qui réduit

l'effet temporel que nous supposons poser problème pour la prise en compte de la non-stationnarité. Ceci explique également les meilleures performances de wXKTD par rapport à XKTD. Dans tous les cas les performances de la famille KTD sont meilleures que celles de l'état de l'art.

4.6 Bilan

Dans le chapitre 3 a été mis en avant le problème que risquait de poser les transitions stochastiques en termes de biais de l'estimateur, dans la mesure où KTD minimise un résidu quadratique de Bellman. Dans ce chapitre, nous avons montré le biais qui résulte de l'hypothèse de bruit blanc dans la fonction de coût minimisée par Kalman, ce biais étant logiquement nul dans le cas de transitions déterministes. Les approches classiques permettant de s'affranchir de ce type de biais dans le cas de la minimisation d'un résidu de Bellman (double échantillonnage, introduction d'une fonction auxiliaire ou encore utilisation de variables instrumentales) ne pouvant s'adapter au cadre de travail proposé, nous traitons ce problème en introduisant un bruit coloré particulier. Etant donnée sa nature (c'est un bruit à moyenne mobile), il a fallu proposer une méthodologie pour pouvoir le prendre en compte dans une approche inspirée du filtrage de Kalman. Nous avons discuté l'effet mémoire (qui peut être compris par analogie avec les traces d'éligibilité, avec le facteur $\lambda = 1$) inhérent à l'algorithme résultant (XKTD) et montré en quoi cela posait problème dans le contexte d'un apprentissage *off-policy*. Cela nous a poussé à introduire une version pondérée de XKTD, qui ne résout pas tout à fait le problème mais que nous avons montré expérimentalement se révéler utile dans un cadre d'itération optimiste de la politique. Nous avons montré que XKTD tend à minimiser un coût quadratique associant aux valeurs les retours cumulés de récompenses qui seraient estimés par Monte Carlo. En conséquence, pour un bruit d'observation unitaire, un bruit d'évolution nul et une paramétrisation linéaire, XKTD et LSTD(1) convergent vers la même solution. Dans la partie expérimentale, nous avons montré que XKTD permettait effectivement de supprimer le biais causé par les transitions stochastiques. Cependant, dans un contexte de contrôle, les résultats sont moins satisfaisants que ceux de KTD. Nous pensons que cela est dû à l'effet mémoriel inhérent à l'algorithme et wXKTD, sa version pondérée qui a tendance à le minimiser, présente en effet de meilleurs résultats. Nous allons maintenant généraliser KTD et XKTD en adaptant le concept des traces d'éligibilité à notre cadre de travail.

Chapitre 5

Une extension des différences temporelles de Kalman aux traces d'éligibilité

Nous avons vu dans le chapitre 2 que le concept des traces d'éligibilité jette un pont entre les méthodes de différences temporelles, qui ont un caractère local (la mise à jour ne dépend que de la transition courante) et les méthodes de Monte Carlo, qui ont un caractère plus global (une mise à jour nécessite de simuler une trajectoire dans son ensemble). Un autre avantage est que lorsque le facteur d'éligibilité λ vaut 1, l'approche revient à faire une simulation de Monte Carlo, mais sans nécessité de simuler une trajectoire à partir d'un état pour mettre à jour la valeur de ce dernier (c'est la différence entre "vue avant" et "vue arrière"). L'algorithme KTD met à jour la fonction de valeur de manière locale, de façon similaire aux méthodes plus classiques de différences temporelles. Nous avons montré que l'algorithme XKTD revient à utiliser un algorithme d'apprentissage supervisé (minimisant un coût quadratique) associant à la valeur de chaque état rencontré le cumul espéré de récompenses obtenu par une simulation de Monte Carlo. Il est dès Lors assez naturel de se demander s'il n'est pas possible de jeter un pont entre KTD et XKTD, comme les traces d'éligibilité plus classiques le font entre les algorithmes de différences temporelles et Monte Carlo. C'est ce que nous proposons dans ce chapitre. Cependant, notre approche est sensiblement différente. Plutôt que de maintenir explicitement un vecteur de traces d'éligibilité, comme cela est fait classiquement, nous définissons une nouvelle classe de bruits d'observation colorés.

5.1 Motivation d'une extension aux traces d'éligibilité

Comme nous l'avons annoncé, l'approche que nous proposons et que nous nommerons KTD(λ) jette un pont entre KTD, qui se rapproche des méthodes de différences temporelles par son caractère local et XKTD, qui se rapproche des méthodes de Monte Carlo par son caractère global. Cependant, au delà de son aspect unificateur, l'approche proposée dans ce chapitre présente potentiellement d'autres intérêts.

Il a été montré que l'estimateur relatif à l'algorithme KTD est biaisé dans le cas de transitions stochastiques, ce qui a justifié l'introduction d'un modèle de bruit coloré et de XKTD, l'algorithme résultant. Toutefois, une des motivations principales à l'introduction

de KTD était la gestion des non-stationnarités, plus particulièrement pour prendre en compte le fait que la fonction de valeur cible change en même temps que la politique dans le cadre d'une itération de la politique généralisée. Nous avons vu que XKTD est moins efficace que KTD de ce point de vue. En effet, si la mise à jour de KTD ne dépend que de l'erreur de différence temporelle courante, la mise à jour de XKTD est faite en fonction de tout l'historique des erreurs de différences temporelles, ce que nous pensons ralentir l'adaptation dans le cas non-stationnaire. Cela a d'ailleurs été illustré sur le problème du *mountain-car* stochastique dans la section 4.5 : malgré la nature stochastique du problème, KTD converge plus rapidement que XKTD dans un contexte d'itération de la politique généralisée (politique ϵ-gloutonne dans ce cas précis). L'idée est qu'en introduisant une variation de KTD basée sur le principe des traces d'éligibilité, dont KTD et XKTD seraient deux cas extrêmes, on pourrait envisager de combiner à la fois le caractère non-biaisé de XKTD avec le caractère local de KTD. Nous l'illustrerons dans la partie expérimentale.

Une autre conséquence intéressante de l'introduction des traces d'éligibilité pourrait être une réduction de la variance de l'estimation. En effet, pour les algorithmes d'apprentissage par renforcement plus classiques, l'utilisation des traces d'éligibilité peut permettre une réduction de la variance de l'estimateur de la fonction de valeur (par rapport aux méthodes de Monte Carlo particulièrement, qui présentent naturellement une forte variance). Le même comportement pourrait apparaître pour KTD(λ), ceci sera également expérimenté. De plus, pour les algorithmes classiques de traces d'éligibilité, le meilleur comportement expérimental n'est pas observé pour $\lambda = 1$, au contraire (voir par exemple [105, chapitre 7.18]). Nous montrons expérimentalement que c'est également le cas pour l'extension proposée.

Dans la prochaine section nous présentons le principe sous-jacent à KTD(λ), qui introduit une nouvelle famille de bruits colorés dépendant d'un paramètre λ, que nous appellerons facteur d'éligibilité comme pour les autres approches plus usuelles. De la même façon qu'il a fallu intégrer le modèle de bruit coloré de la section 4.2, l'utilisation de ces nouveaux bruits dans le cadre de KTD demandera quelques hypothèses et un peu de travail. Mais cela permettra de dériver l'algorithme KTD(λ), que nous montrerons être en fait XKTD pour $\lambda = 0$ et KTD pour $\lambda = 1$ (on a donc Monte Carlo qui correspond à $\lambda = 0$, ce qui n'est pas usuel). Nous proposons également une analyse de convergence, similaire à celle qui est faite dans le chapitre 4. Le chapitre se conclut avec une série d'expériences.

5.2 Principe et algorithme

Dans la section 4.2, nous avons introduit un modèle génératif de la récompense basé sur la fonction de valeur en deux états consécutifs plus un bruit coloré composé de deux résidus toujours supposés blancs. Nous en rappelons l'expression :

$$R^\pi(s, s') = V^\pi(s) - \gamma V^\pi(s') + \left(\Delta V^\pi(s) - \gamma \Delta V^\pi(s')\right) \tag{5.1}$$

Nous notons ici $N^{(1)}$ le bruit composé des deux résidus :

$$N^{(1)}(s, s') = \Delta V^\pi(s) - \gamma \Delta V^\pi(s') \tag{5.2}$$

Rappelons que le principe des traces d'éligibilité (en "vue avant") consiste à considérer une moyenne géométrique des erreurs de différences temporelles à k pas (voir section 2.2.2). Nous faisons de même avec ce modèle comportant un terme de bruit. Ainsi, pour une prédiction à deux pas nous avons :

$$R^\pi(s, s') + \gamma R^\pi(s', s'') = V^\pi(s) - \gamma^2 V^\pi(s'') + \left(\Delta V^\pi(s) - \gamma^2 \Delta V^\pi(s'')\right) \tag{5.3}$$

De manière générale, nous notons $N^{(k)}$ le bruit correspondant à une prédiction à k pas :

$$N^{(k)}(s, s^{(k)}) = \Delta V^{\pi}(s) - \gamma^k \Delta V^{\pi}(s^{(k)}) \tag{5.4}$$

Ainsi, on peut généralement considérer une prédiction à k pas :

$$\sum_{i=0}^{k-1} \gamma^i R^{\pi}(s^{(i)}, s^{(i+1)}) = V^{\pi}(s) - \gamma^k V^{\pi}(s^{(k)}) + N^{(k)}(s, s^{(k)}) \tag{5.5}$$

5.2.1 Une nouvelle classe de bruits

Pour l'approche classique des traces d'éligibilité présentée dans le chapitre 2, c'est la moyenne géométrique des différences temporelles d'ordre k qui est considérée. Ici, nous allons plutôt nous intéresser à la moyenne géométrique des bruits d'ordre k. C'est une vision différente d'une même chose, dans la mesure où le bruit d'observation peut être vu comme un équivalent probabiliste de l'erreur de différence temporelle. Par exemple, pour la prédiction d'ordre k nous pouvons écrire :

$$\begin{aligned} &\sum_{i=0}^{k-1} \gamma^i R^{\pi}(s^{(i)}, s^{(i+1)}) = V^{\pi}(s) - \gamma^k V^{\pi}(s^{(k)} + N^{(k)}(s, s^{(k)}) \\ \Leftrightarrow\ & N^{(k)}(s, s^{(k)}) = \sum_{i=0}^{k-1} \gamma^i R^{\pi}(s^{(i)}, s^{(i+1)}) + \gamma^k V^{\pi}(s^{(k)}) - V^{\pi}(s) \end{aligned} \tag{5.6}$$

Nous définissons N^{λ} cette moyenne géométrique des bruits :

$$\begin{aligned} N^{\lambda} &= (1-\lambda) \sum_{k=1}^{\infty} \lambda^{k-1} N^{(k)} \\ &= \Delta V^{\pi}(s) - \gamma(1-\lambda) \left(\Delta V^{\pi}(s') + \cdots + (\gamma\lambda)^{k-1} \Delta V^{\pi}(s^{(k)}) + \dots \right) \end{aligned} \tag{5.7}$$

Comme pour la dérivation de XKTD nous supposons que les résidus sont blancs. Cette hypothèse forte a déjà été discutée. Nous notons u_i le bruit blanc centré de variance théorique $\Delta V^{\pi}(s^{(i)})$. Un modèle de bruit d'observation possible (et équivalent au bruit (5.7) sous les hypothèses données) est donc le suivant :

$$n_i = u_i - \gamma(1-\lambda) \left(u_{i+1} + \gamma\lambda u_{i+2} + \cdots + (\gamma\lambda)^{k-1} u_{i+k} + \dots \right) \tag{5.8}$$

Un premier problème est que ce bruit est non-causal et est donc difficile à intégrer dans un filtre de Kalman, par essence causal. Cependant, ce sont les corrélations induites par ce bruit qui nous intéressent. Si l'on suppose la variance du bruit blanc constante, cette corrélation n'est pas affectée par la causalité (le bruit non-causal et son équivalent causal ont les mêmes corrélations). Nous supposons donc la variance des résidus constante, ce qui permet de réécrire le bruit de manière causale (c'est le pendant des vues avant et arrière des traces d'éligibilité) :

$$n_i = u_i - \gamma(1-\lambda) \left(u_{i-1} + \gamma\lambda u_{i-2} + \cdots + (\gamma\lambda)^{k-1} u_{i-k} + \dots \right) \tag{5.9}$$

Cette hypothèse de variance constante des résidus peut paraître forte. Cependant, nous avons vu dans les sections 3.6 et 4.4 que pour les fonctions de coût minimisées par KTD et XKTD, ce terme de variance n'apparaît qu'en tant que pondération des coûts quadratiques instantanés. Cette hypothèse porte donc moins à conséquence que l'on pourrait le supposer *a priori*. De plus, pratiquement, nous considérons toujours ce bruit stationnaire.

5.2.2 Intégration à Kalman

Le bruit n_i ayant été introduit, il reste encore à l'intégrer dans KTD. La technique est similaire à celle utilisée pour l'obtention de XKTD. Il faut d'abord remarquer que n_i peut se décomposer en la somme d'un bruit blanc u_i et d'un bruit autorégressif b_i :

$$n_i = u_i - \gamma(1-\lambda)b_{i-1} \tag{5.10}$$

$$\begin{aligned} \text{avec } b_i &= u_i + \gamma\lambda u_{i-1} + \cdots + (\gamma\lambda)^k u_{i-k} + \dots \\ &= u_i + \gamma\lambda b_{i-1} \end{aligned} \tag{5.11}$$

Nous utilisons ensuite les deux bruits n_i et b_i pour introduire un bruit auto-régressif vectoriel $\begin{pmatrix} b_i & n_i \end{pmatrix}^T$:

$$\begin{pmatrix} b_i \\ n_i \end{pmatrix} = \begin{pmatrix} \gamma\lambda & 0 \\ -\gamma(1-\lambda) & 0 \end{pmatrix} \begin{pmatrix} b_{i-1} \\ n_{i-1} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u_i \tag{5.12}$$

Pour intégrer ce bruit coloré à KTD, il suffit d'étendre le modèle espace d'état comme cela a déjà été fait auparavant, ce qui donne le modèle suivant :

$$\begin{cases} \mathbf{x}_i = F(\lambda)\mathbf{x}_{i-1} + v'_i \\ r_i = g_{t_i}(\mathbf{x}_i) \end{cases} \tag{5.13}$$

Nous détaillons chacun de ces termes. Le vecteur de paramètres est maintenant étendu avec le bruit AR vectoriel $\begin{pmatrix} b_i & n_i \end{pmatrix}^T$:

$$\mathbf{x}_i = \begin{pmatrix} \theta_i \\ b_i \\ n_i \end{pmatrix} \tag{5.14}$$

La matrice d'évolution $F(\lambda)$ rend compte de la structure du nouveau bruit d'observation. Nous rappelons que I_p désigne la matrice identité de taille $p \times p$:

$$F(\lambda) = \begin{pmatrix} I_p & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \gamma\lambda & 0 \\ \mathbf{0}^T & -\gamma(1-\lambda) & 0 \end{pmatrix} \tag{5.15}$$

Le bruit d'évolution v_i est également étendu de façon à prendre en compte ce bruit coloré, $v'_i = \begin{pmatrix} v_i & u_i & u_i \end{pmatrix}^T$; ce nouveau bruit de process est centré et sa matrice de variance $P_{v'_i}$ peut s'écrire par blocs en utilisant le fait que les bruits v_i et u_i sont indépendants (et donc décorrélés) :

$$v'_i = \begin{pmatrix} v_i \\ u_i \\ u_i \end{pmatrix} \sim \left(\begin{pmatrix} \mathbf{0} \\ 0 \\ 0 \end{pmatrix}, P_{v'_i} = \begin{pmatrix} P_{v_i} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \sigma_i^2 & \sigma_i^2 \\ \mathbf{0}^T & \sigma_i^2 & \sigma_i^2 \end{pmatrix} \right) \tag{5.16}$$

Enfin, l'équation d'observation reste la même, malgré la notation quelque peu abusive maintenant connue :

$$r_i = g_{t_i}(\mathbf{x}_i) = g_{t_i}(\theta_i) + n_i \tag{5.17}$$

Comme précédemment, le bruit d'observation est une part de l'équation d'évolution, il doit donc être estimé.

L'approche résultante est résumée dans l'algorithme 5.1, très similaire à l'algorithme 4.1 qui correspond à XKTD (la seule différence résultant dans la matrice d'évolution utilisée).

Algorithme 5.1 : KTD(λ) général

Initialisation : *a priori* $\hat{\mathbf{x}}_{0|0}$ et $P_{0|0}$;

pour $i \leftarrow 1, 2, \ldots$ **faire**

Observer la transition t_i ainsi que la récompense r_i ;

Phase de prédiction;
$\hat{\mathbf{x}}_{i|i-1} = F(\lambda)\hat{\mathbf{x}}_{i-1|i-1}$;
$P_{i|i-1} = F(\lambda)P_{i-1|i-1}F(\lambda)^T + P_{v_i'}$;

Calcul des statistiques d'intérêt;
$\hat{r}_{i|i-1} = E[g_{t_i}(\theta_i) + n_i | r_{1:i-1}]$;
$P_{\mathbf{x}r_i} = E\left[(\mathbf{x}_i - \hat{\mathbf{x}}_{i|i-1})(g_{t_i}(\theta_i) + n_i - \hat{r}_{i|i-1})|r_{1:i-1}\right]$;
$P_{r_i} = E\left[(g_{t_i}(\theta_i) + n_i - \hat{r}_{i|i-1})^2|r_{1:i-1}\right]$;

Phase de correction;
$K_i = P_{\mathbf{x}r_i}P_{r_i}^{-1}$;
$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_{i|i-1} + K_i\left(r_i - \hat{r}_{i|i-1}\right)$;
$P_{i|i} = P_{i|i-1} - K_iP_{r_i}K_i^T$;

Une fois encore la transformation non-parfumée présentée chapitre 3 peut être utilisée pour dériver des algorithmes pratiques. L'approche est la même que pour XKTD, présentée chapitre 4, nous donnons simplement l'algorithme 5.2 correspondant. Notons que pour la même raison que nous n'avons pas défini d'extension XKTD-Q, c'est-à-dire le caractère *off-policy* des algorithmes basés sur l'itération de la valeur, nous ne proposons pas d'extension KTD-Q(λ). Les algorithmes permettant respectivement l'évaluation de la valeur et de la fonction de qualité (soit KTD-V(λ) et KTD-SARSA(λ)) sont donnés.

5.2.3 Lien avec KTD et XKTD

Nous étudions à présent deux cas limites de KTD(λ), lorsque $\lambda = 0$ et $\lambda = 1$. Rappelons la forme du bruit introduit :

$$\begin{pmatrix} b_i \\ n_i \end{pmatrix} = \begin{pmatrix} \gamma\lambda & 0 \\ -\gamma(1-\lambda) & 0 \end{pmatrix} \begin{pmatrix} b_{i-1} \\ n_{i-1} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u_i \tag{5.18}$$

Considérons tout d'abord le cas où $\lambda = 1$. Nous obtenons :

$$\begin{pmatrix} b_i \\ n_i \end{pmatrix} = \begin{pmatrix} \gamma & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} b_{i-1} \\ n_{i-1} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u_i \tag{5.19}$$

Dans ce cas, le bruit auto-régressif b_i est toujours estimé mais ce n'est plus une composante du bruit d'observation n_i, qui se réduit au bruit blanc u_i. Nous obtenons exactement l'algorithme KTD. Considérons l'autre cas extrême, pour lequel $\lambda = 0$. Nous avons alors :

$$\begin{pmatrix} b_i \\ n_i \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ -\gamma & 0 \end{pmatrix} \begin{pmatrix} b_{i-1} \\ n_{i-1} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u_i \tag{5.20}$$

Dans ce cas, b_i joue le rôle de variable aléatoire auxiliaire (dont le rôle est la mémorisation du bruit blanc à l'instant précédent) et l'on retrouve le modèle de bruit à moyenne mobile

Algorithme 5.2 : KTD-V(λ) et KTD-SARSA(λ)

Initialisation : *a priori* $\hat{\mathbf{x}}_{0|0} = \begin{pmatrix} \hat{\theta}_{0|0}^T & 0 & 0 \end{pmatrix}^T$ et $P_{0|0}$;

pour $i \leftarrow 1, 2, \ldots$ **faire**

Observer la transition $t_i = \begin{cases} (s_i, s_{i+1}) \text{ (KTD-V}(\lambda)\text{)} \\ (s_i, a_i, s_{i+1}, a_{i+1}) \text{ (KTD-SARSA}(\lambda)\text{)} \end{cases}$ ainsi que la récompense associée r_i ;

Phase de prédiction;

$\hat{\mathbf{x}}_{i|i-1} = F(\lambda)\hat{\mathbf{x}}_{i-1|i-1}$;

$P_{i|i-1} = F(\lambda)P_{i-1|i-1}F(\lambda)^T + P_{v'_{i-1}}$;

Calcul des sigma-points ;

$\mathbf{X}_{i|i-1} = \left\{ \hat{\mathbf{x}}_{i|i-1}^{(j)}, \quad 0 \leq j \leq 2p+4 \right\}$ (en utilisant $\hat{\mathbf{x}}_{i|i-1}$ et $P_{i|i-1}$);

$\mathcal{W} = \{ w_j, \quad 0 \leq j \leq 2p+4 \quad \}$;

`/* notons que` $(\hat{\mathbf{x}}_{i|i-1}^{(j)})^T = \left((\hat{\theta}_{i|i-1}^{(j)})^T \quad \hat{b}_{i|i-1}^{(j)} \quad \hat{n}_{i|i-1}^{(j)} \right)$ `*/`

$\mathcal{R}_{i|i-1} =$

$$\begin{cases} \text{(KTD-V}(\lambda)\text{)} \\ \left\{ \hat{r}_{i|i-1}^{(j)} = \hat{V}_{\hat{\theta}_{i|i-1}^{(j)}}(s_i) - \gamma \hat{V}_{\hat{\theta}_{i|i-1}^{(j)}}(s_{i+1}) + \hat{n}_{i|i-1}^{(j)}, \, 0 \leq j \leq 2p+4 \right\} \\ \text{(KTD-SARSA}(\lambda)\text{)} \\ \left\{ \hat{r}_{i|i-1}^{(j)} = \hat{Q}_{\hat{\theta}_{i|i-1}^{(j)}}(s_i, a_i) - \gamma \hat{Q}_{\hat{\theta}_{i|i-1}^{(j)}}(s_{i+1}, a_{i+1}) + \hat{n}_{i|i-1}^{(j)}, \, 0 \leq j \leq 2p+4 \right\} \end{cases} ;$$

Calcul des statistiques d'intérêt;

$\hat{r}_{i|i-1} = \sum_{j=0}^{2p+4} w_j \hat{r}_{i|i-1}^{(j)}$;

$P_{\mathbf{x}r_i} = \sum_{j=0}^{2p+4} w_j (\hat{\mathbf{x}}_{i|i-1}^{(j)} - \hat{\mathbf{x}}_{i|i-1})(\hat{r}_{i|i-1}^{(j)} - \hat{r}_{i|i-1})$;

$P_{r_i} = \sum_{j=0}^{2p+4} w_j \left(\hat{r}_{i|i-1}^{(j)} - \hat{r}_{i|i-1} \right)^2$;

Phase de correction;

$K_i = P_{\mathbf{x}r_i} P_{r_i}^{-1}$;

$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_{i|i-1} + K_i \left(r_i - \hat{r}_{i|i-1} \right)$;

$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

utilisé pour XKTD. Remarquons que si les modèles de bruit ne sont pas *stricto sensu* les mêmes, les corrélations induites sont identiques (sous hypothèse de variance constante des résidus).

Ainsi, nous avons montré que les approches proposées dans les chapitres 3 et 4 sont des cas particuliers de KTD(λ). Notons tout de même que contrairement à l'usage, ici c'est $\lambda = 0$ qui correspond à la méthode se rapprochant le plus de Monte Carlo (soit XKTD) et non $\lambda = 1$, qui correspond ici à la mise à jour la plus locale. Pour se ramener à des notations plus usuelles il suffirait de remplacer λ par $1 - \lambda$ dans la matrice d'évolution $F(\lambda)$.

5.3 Analyse de convergence

Nous avons montré dans les chapitres précédents que l'estimateur de KTD minimise un résidu de Bellman quadratique et que l'estimateur de XKTD minimise l'erreur quadratique liant les états aux estimation par Monte Carlo du cumul de récompenses. Nous montrons ici que KTD(λ) minimise une fonction de coût intermédiaire.

Théorème 5.1 (Convergence de KTD(λ)). *Supposons que les distributions des bruits et a posteriori soient gaussiennes et que l'a priori soit également gaussien, de moyenne θ_0 et variance P_0. Supposons également que la variance des résidus σ^2 soit constante. Alors l'estimateur de KTD(λ) minimise le coût suivant :*

$$\begin{aligned}\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmin}} \Big(\frac{1}{\sigma^2} \sum_{j=0}^{i} \Big\{ & r_j + \gamma \hat{V}_\theta(s_{j+1}) - \hat{V}_\theta(s_j) \\ & + \gamma(1-\lambda) \sum_{k=j+1}^{i} \gamma^{k-j-1} \Big(r_k + \gamma \hat{V}_\theta(s_{k+1}) - \hat{V}_\theta(s_k) \Big) \Big\}^2 \\ & + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \Big)\end{aligned} \tag{5.21}$$

Un résultat similaire existe pour la Q-fonction.

Démonstration. La démonstration est similaire à celle qui a été faite pour la convergence de XKTD. Nous utilisons à nouveau le résultat de [116, chapitre 4.5]. Nous rappelons que cette preuve est faite dans le cas d'un modèle d'évolution basé sur une marche aléatoire (c'est-à-dire une matrice d'évolution qui vaut l'identité), mais elle peut facilement être étendue à un modèle d'évolution linéaire. Ce résultat peut donc être appliqué au modèle espace-d'état (5.13) et l'estimateur correspondant est l'estimateur du maximum *a posteriori* (MAP) :

$$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_i^{\text{MAP}} = \underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{x}|r_{1:i}) \tag{5.22}$$

Or le modèle espace-d'état (5.13) est équivalent au modèle espace-d'état plus usuel pour lequel le bruit d'observation n_i est coloré :

$$\begin{cases} \theta_i = \theta_{i-1} + v_i \\ r_i = \hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) + n_i \end{cases} \tag{5.23}$$

En utilisant la conjecture 4.1, les espaces-état étant équivalents et l'estimateur de l'un étant l'estimateur du MAP, alors l'estimateur de l'autre l'est également, ce qui nous mène à :

$$\hat{\theta}_{i|i} = \hat{\theta}_i^{\text{MAP}} = \underset{\theta}{\operatorname{argmax}}\, p(\theta|r_{1:i}) \tag{5.24}$$

Nous appliquons la loi de Bayes à la densité *a posteriori* :

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmax}} \frac{p(r_{1:i}|\theta)p(\theta)}{p(r_{1:i})} \tag{5.25}$$

Or le dénominateur ne dépend pas des paramètres, la maximisation de la distribution *a posteriori* revient donc à la maximisation du produit de la vraisemblance et de l'*a priori* :

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmax}}\, p(r_{1:i}|\theta)p(\theta) \tag{5.26}$$

De plus, la maximisation d'une densité de probabilité est équivalente à la minimisation du négatif de son logarithme, donc finalement :

$$\hat{\theta}_{i|i} = \operatorname{argmin} \left(-\ln\left(p(r_{1:i}|\theta)\right) - \ln(p(\theta))\right) \tag{5.27}$$

Cependant, le bruit d'observation n'étant pas blanc, la vraisemblance ne peut pas être décomposée. Nous rappelons quelques notations utilisées dans la preuve du théorème 4.2 et en définissons d'autres. Soient $V_i(\theta)$, R_i et N_i les vecteurs contenant respectivement l'historique des valeurs, des récompenses et des bruits :

$$V_i(\theta) = \begin{pmatrix} \hat{V}_\theta(s_1) & \hat{V}_\theta(s_2) & \dots & \hat{V}_\theta(s_i) \end{pmatrix}^T \tag{5.28}$$

$$R_i = \begin{pmatrix} r_1 & r_2 & \dots & r_i \end{pmatrix}^T \tag{5.29}$$

$$N_i = \begin{pmatrix} n_1 & n_2 & \dots & n_i \end{pmatrix}^T \tag{5.30}$$

Nous rappelons également l'expression de la matrice $\mathbf{H}_i$ de taille $i \times i$:

$$\mathbf{H}_i = \begin{pmatrix} 1 & -\gamma & 0 & \dots \\ 0 & 1 & -\gamma & 0 \\ \vdots & \ddots & \ddots & -\gamma \\ 0 & \dots & 0 & 1 \end{pmatrix} \tag{5.31}$$

Nous notons également $\Sigma_{N_i} = E[N_i N_i^T]$ la matrice de variance du vecteur de bruit N_i. Le bruit d'observation étant coloré, cette matrice n'est pas diagonale (vue matricielle de la non-décomposition de la vraisemblance). Cependant, par hypothèse le bruit d'observation est gaussien et nous connaissons donc la distribution de la vraisemblance :

$$r_{1:i}|\theta \sim \mathcal{N}\left(R_i - \mathbf{H}_i V_i(\theta), \Sigma_{N_i}\right) \tag{5.32}$$

Comme l'estimateur minimise le négatif du logarithme de la vraisemblance (au terme d'*a priori* près), nous avons :

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmin}} \left((R_i - \mathbf{H}_i V_i(\theta))^T \Sigma_{N_i}^{-1} (R_i - \mathbf{H}_i V_i(\theta)) + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \right) \tag{5.33}$$

Il reste donc à déterminer l'inverse de la variance du bruit, ce qui est l'approche que nous avons déjà utilisée pour montrer le premier résultat de convergence pour XKTD. Introduisons la matrice $\mathbf{F}_i(\lambda)$, qui à la transposition près est la matrice qui transforme un bruit blanc u_i en le bruit coloré n_i :

$$\mathbf{F}_i(\lambda) = \begin{pmatrix} 1 & -\gamma(1-\lambda) & -\gamma^2\lambda(1-\lambda) & \dots & -\gamma^{i-1}\lambda^{i-2}(1-\lambda) \\ 0 & 1 & -\gamma(1-\lambda) & \dots & -\gamma^{i-2}\lambda^{i-3}(1-\lambda) \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} \tag{5.34}$$

Rappelons que nous avons supposé la variance des résidus constante, soit σ^2 cette variance. La matrice de variance du bruit N_i s'écrit :

$$\Sigma_{N_i} = \sigma^2 \mathbf{F}_i(\lambda)\mathbf{F}_i(\lambda)^T \tag{5.35}$$

Or la matrice $\mathbf{F}_i$ peut également être décomposée. Pour cela, nous introduisons $\mathbf{G}_i(\lambda)$, qui est la même matrice que $\mathbf{H}_i$ en remplaçant γ par $\gamma\lambda$. Nous avons donc immédiatement :

$$\mathbf{G}_i(\lambda) = \begin{pmatrix} 1 & -\gamma\lambda & 0 & \dots & 0 \\ 0 & 1 & -\gamma\lambda & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} \tag{5.36}$$

$$\mathbf{G}_i^{-1}(\lambda) = \begin{pmatrix} 1 & \gamma\lambda & (\gamma\lambda)^2 & \dots & (\gamma\lambda)^{i-1} \\ 0 & 1 & \gamma\lambda & \dots & (\gamma\lambda)^{i-2} \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} \tag{5.37}$$

Ces notations ayant été introduites, c'est un simple exercice algébrique que de vérifier que :

$$\mathbf{F}_i(\lambda) = \mathbf{H}_i\mathbf{G}_i^{-1}(\lambda) \tag{5.38}$$

Nous pouvons donc écrire pour l'estimateur :

$$\begin{aligned} \hat{\theta}_{i|i} &= \underset{\theta}{\operatorname{argmin}} \left((R_i - \mathbf{H}_i V_i(\theta))^T \Sigma_{N_i}^{-1} (R_i - \mathbf{H}_i V_i(\theta)) + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \right) \\ &= \underset{\theta}{\operatorname{argmin}} \left((R_i - \mathbf{H}_i V_i(\theta))^T \sigma^{-2} \left((\mathbf{G}_i(\lambda)\mathbf{H}_i^{-1})^T (\mathbf{G}_i(\lambda)\mathbf{H}_i^{-1}) \right) (R_i - \mathbf{H}_i V_i(\theta)) + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \right) \\ &= \underset{\theta}{\operatorname{argmin}} \left(\frac{1}{\sigma^2} \left\| \mathbf{G}_i(\lambda)\mathbf{H}_i^{-1} R_i - \mathbf{G}_i(\lambda) V_i(\theta) \right\|^2 + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \right) \end{aligned} \tag{5.39}$$

Sous cette forme et en remarquant que $\mathbf{G}_i(0) = I$ et que $\mathbf{G}_i(1) = \mathbf{H}_i$, le parallèle avec KTD et XKTD est assez clair. Pour obtenir le résultat tel qu'énoncé, on peut réécrire la dernière équation, ou encore plus simplement :

$$\begin{aligned} \hat{\theta}_{i|i} &= \underset{\theta}{\operatorname{argmin}} \left((R_i - \mathbf{H}_i V_i(\theta))^T \Sigma_{N_i}^{-1} (R_i - \mathbf{H}_i V_i(\theta)) + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \right) \\ &= \underset{\theta}{\operatorname{argmin}} \Big((R_i - \mathbf{H}_i V_i(\theta))^T \sigma^{-2} \left((\mathbf{F}_i^{-1}(\lambda))^T (\mathbf{F}_i^{-1}(\lambda)) \right) (R_i - \mathbf{H}_i V_i(\theta)) \\ &\qquad + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \Big) \\ &= \underset{\theta}{\operatorname{argmin}} \left(\frac{1}{\sigma^2} \left\| \mathbf{F}_i^{-1}(\lambda)(R_i - \mathbf{H}_i V_i(\theta)) \right\|^2 + (\theta - \theta_0)^T P_0^{-1} (\theta - \theta_0) \right) \end{aligned} \tag{5.40}$$

Il est facile de vérifier que la matrice $\mathbf{F}_i^{-1}(\lambda)$ s'écrit :

$$\mathbf{F}_i^{-1}(\lambda) = \begin{pmatrix} 1 & \gamma(1-\lambda) & \gamma^2(1-\lambda) & \dots & \gamma^{i-1}(1-\lambda) \\ 0 & 1 & \gamma(1-\lambda) & \dots & \gamma^{i-2}(1-\lambda) \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} \tag{5.41}$$

Les deux dernières équations montrent le résultat. □

Le coût quadratique instantané est donc une combinaison des erreurs de différences temporelles de tous les états rencontrés sur la trajectoire. Nous obtenons une fonction de coût intermédiaire entre la minimisation d'un résidu quadratique de Bellman (que l'on retrouve pour $\lambda = 1$) et la minimisation d'un coût quadratique associant les valeurs au cumul de récompenses associé obtenu par Monte Carlo (que l'on retrouve pour $\lambda = 0$). Une question qui reste ouverte est de savoir si l'estimateur correspondant est biaisé ou non. Nous tentons d'y répondre expérimentalement.

5.4 Expérimentations

Dans cette section nous proposons principalement de reprendre les expérimentations de la section 4.5 concernant la chaîne de Boyan, la chaîne contrôlée ainsi que la version stochastique du *mountain car*. Nous avons vu dans le chapitre 4 que les résultats de XKTD sont sensiblement moins bons que ceux de KTD dans un contexte de contrôle. Nous supposons que cela est dû principalement au caractère global des mises à jour de XKTD, qui poserait problème lorsque la fonction de valeur (ou de qualité) cible n'est pas stationnaire. Nous pensons que l'introduction des traces d'éligibilité devrait permettre d'une part de réduire le biais par rapport à l'estimateur de KTD et d'autre part de rendre les mises à jour plus locales que pour XKTD.

5.4.1 Chaîne de Boyan

Dans cette section nous expérimentons la chaîne de Boyan stochastique, telle que décrite dans la section 3.8.1. Nous nous intéressons dans un premier temps au biais ainsi qu'à la vitesse de convergence de KTD(λ) pour différentes valeur de λ dans un version stationnaire de la chaîne. Nous considérons également les algorithmes TD et LSTD comme étalon. La même paramétrisation et le même facteur d'actualisation sont utilisés. Pour TD le taux d'apprentissage est choisi constant comme auparavant : $\alpha = 0.1$. Pour LSTD, l'*a priori* est choisi égal à $P_{0|0} = I$. Pour KTD-V(λ) le même *a priori* est utilisé, la variance σ^2 des résidus est choisie égale à 10^{-3} et nous ne mettons pas de bruit de process : $P_{v_i} = 0I$. Notons que nous définissons un bruit nul sur les paramètres, mais étant donné le modèle de bruit particulier envisagé, le bruit sur le vecteur d'état étendu $\mathbf{x}_i$ n'est pas nul. Il est centré et sa variance vaut :

$$P_{v'_i} = \begin{pmatrix} 0I & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \sigma^2 & \sigma^2 \\ \mathbf{0}^T & \sigma^2 & \sigma^2 \end{pmatrix} \tag{5.42}$$

L'apprentissage se fait sur 100 épisodes et les résultats de la figure 5.1 sont moyennés sur 300 essais.

D'après cette expérience, la vitesse de convergence de KTD(λ) est sensiblement la même pour les différentes valeurs de λ et le seul algorithme qui semble être biaisé est KTD(1), soit l'algorithme KTD introduit au chapitre 3. Ce résultat est intéressant : il semble que KTD(λ) réduise le biais de KTD, pour tout λ strictement inférieur à 1. C'est l'un des comportements que nous espérions pour l'algorithme et il se vérifie.

L'une des motivations principales à l'introduction des différences temporelles de Kalman est la capacité de l'algorithme à prendre en compte les non-stationnarités et ce surtout dans le cadre d'une itération de la politique généralisée. Nous testons cela sur la chaîne de Boyan que l'on rend non-stationnaire en inversant le signe de la récompense comme

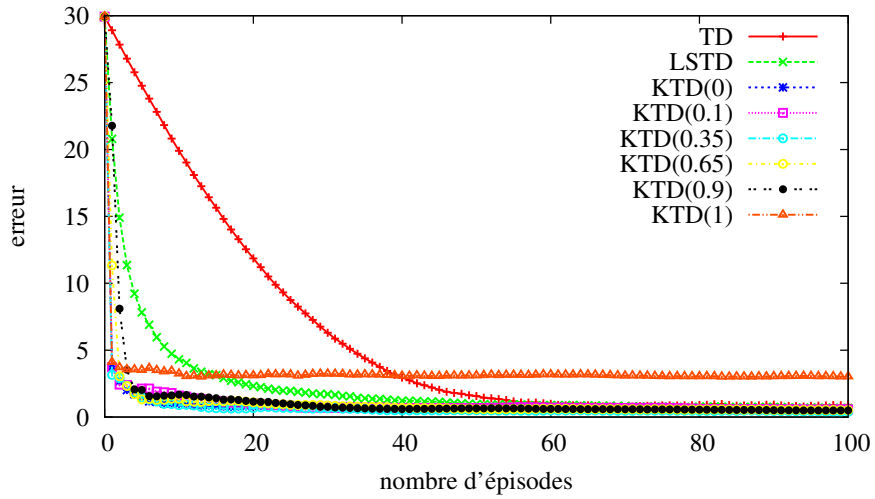


FIG. 5.1 – Chaîne de Boyan, cas stochastique et stationnaire.

précédemment. Les paramètres des algorithmes sont les mêmes que précédemment, sauf le bruit d'observation pour lequel on adopte le schéma de bruit adaptatif déjà utilisé dans les chapitres précédents (voir par exemple la section 4.5.1) :

$$P_{v'_i} = \begin{pmatrix} \eta P_{\theta_{i-1|i-1}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \sigma^2 & -\gamma\sigma^2 \\ \mathbf{0}^T & -\gamma\sigma^2 & \gamma^2\sigma^2 \end{pmatrix} \tag{5.43}$$

Ici η est fixé à 10^{-2}. L'apprentissage se fait sur 140 épisodes (changement du signe de la récompense au $70^{\text{ème}}$ épisode) et les résultats de la figure 5.2 sont moyennés sur 300 essais.

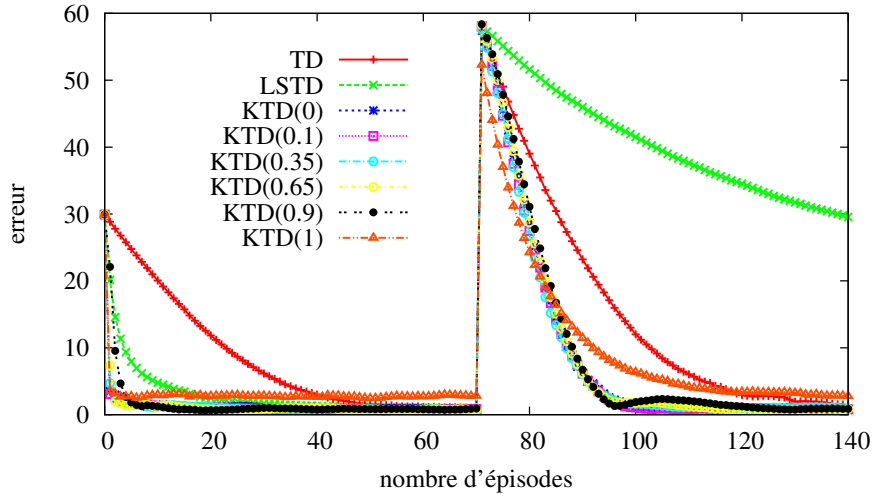


FIG. 5.2 – Chaîne de Boyan, cas stochastique et non-stationnaire.

Cette expérience montre que KTD(λ) permet de prendre en compte la non-stationnarité de l'environnement et ce pour toute valeur de λ. La seule version de KTD à rester biaisée après le changement de récompense (et la période d'adaptation) est KTD(1), soit KTD,

comme précédemment. Sur cet exemple, l'adaptation est sensiblement la même pour les différentes valeurs de λ (exception faite peut-être de $\lambda = 0.9$, qui présente un petit sursaut après le centième épisode que nous ne savons pas expliquer).

Nous avons jusqu'à présent étudié l'erreur moyenne en fonction du nombre d'épisodes. Nous allons à présent nous intéresser à cette erreur moyenne, ainsi qu'à l'écart-type associé, en les voyant comme des fonctions de λ, pour un nombre d'épisodes donnés. L'idée est ici d'étudier l'influence de λ d'une part sur l'erreur moyenne et d'autre part sur la variance associée. Les paramètres sont les mêmes que précédemment, sauf le bruit d'évolution sur les paramètres qui est choisi nul. Les statistiques que nous donnons sont faites sur 1000 essais, chaque essai correspondant à 500 épisodes. L'algorithme considéré est KTD(λ) pour λ prenant 21 valeurs équi-réparties entre 0 et 1. Sur la figure 5.3 nous traçons l'erreur moyenne et l'écart-type associé au bout de 50 épisodes et au bout de 500 épisodes (donc moyenne et écart-type à partir de 1000 éléments), tout cela en fonction du facteur d'éligibilité λ. Notons l'échelle semi-logarithmique.

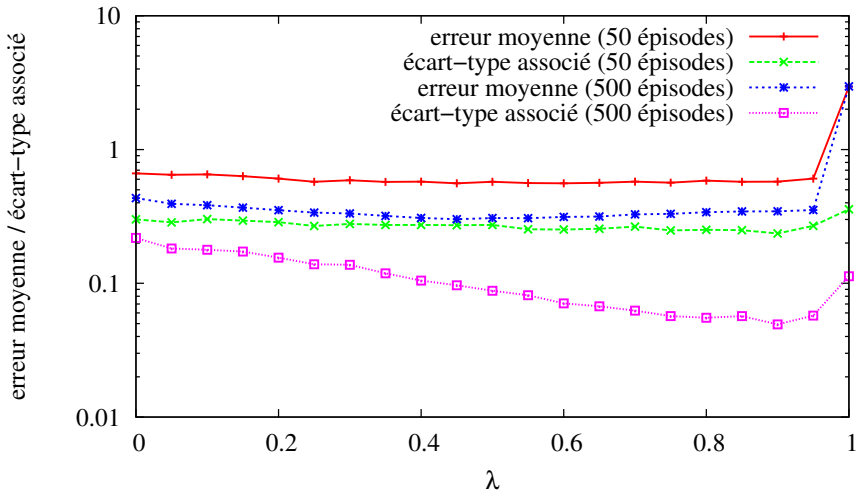


FIG. 5.3 – Chaîne de Boyan, erreur moyenne et écart-type associé.

Au bout de 50 épisodes, il ne semble pas y avoir d'influence significative du facteur d'éligibilité sur l'erreur moyenne ou l'écart-type associé, sauf pour $\lambda = 1$ qui correspond à KTD et donc à un estimateur biaisé. Au bout de 500 épisodes, λ ne semble toujours pas influer sur l'erreur moyenne. Notons que pour $\lambda = 1$ la valeur de l'erreur moyenne est la même pour 50 ou 500 épisodes. Le biais est donc atteint tôt. Cependant, au bout de 500 épisodes, le facteur d'éligibilité a une influence sur l'écart-type de l'erreur, avec un minimum autour de $\lambda = 0.9$. Ainsi, l'extension de KTD aux traces d'éligibilité proposée pourrait permettre de réduire la variance de l'estimateur correspondant, au moins asymptotiquement, ce qui peut s'avérer intéressant. De plus cette variance semble la plus faible pour les valeurs proche de $\lambda = 1$, ce qui correspond à quelques chose de non-biaisé d'une part et de relativement local d'autre part, c'est donc le cas qui nous intéresse *a priori* le plus dans un contexte d'itération de la politique généralisée.

5.4.2 Chaîne contrôlée

Dans cette section nous reprenons les expériences faites sur la chaîne contrôlée dans la section 4.5.2. Nous ne refaisons pas l'expérience sur l'évaluation de la Q-fonction, qui est redondante avec les expérimentations sur la chaîne de Boyan (dans la mesure où l'évaluation de la fonction de valeur ou de la fonction de qualité revient au même, la différence résidant dans la chaîne de Markov valuée induite). Nous présentons une expérience étudiant le comportement dans le contexte d'une itération optimiste de la politique (plus particulièrement une politique ϵ-gloutonne) ainsi que dans le contexte de l'itération de la valeur. La représentation de la Q-fonction et la mesure de performance sont les mêmes que dans la section 4.5.2. Nous comparons KTD(λ) à TD, LSTD et ponctuellement wXKTD (pour l'itération optimiste de la politique).

Itération optimiste de la politique

Nous nous intéressons à l'apprentissage de la Q-fonction optimale avec une politique de type ϵ-gloutonne, ϵ étant fixé à 0.1. Pour TD le taux d'apprentissage choisi est égal à $\alpha_i = \frac{n_0+1}{n_0+i}$ avec $n_0 = 500$. L'*a priori* pour KTD(λ), LSTD et wXKTD est fixé à 5I. La variance des résidus pour KTD(λ) et wXKTD est fixée à $\sigma^2 = 5.10^{-3}$. Nous utilisons toujours un bruit d'évolution adaptatif avec ici $\eta = 10^{-3}$. L'apprentissage se fait sur 700 épisodes (chacun étant initialisé avec une paire état-action aléatoire et limité à 15 interactions) et les résultats présentés sur la figure 4.5 sont moyennés sur 100 essais (notons à nouveau l'échelle logarithmique).

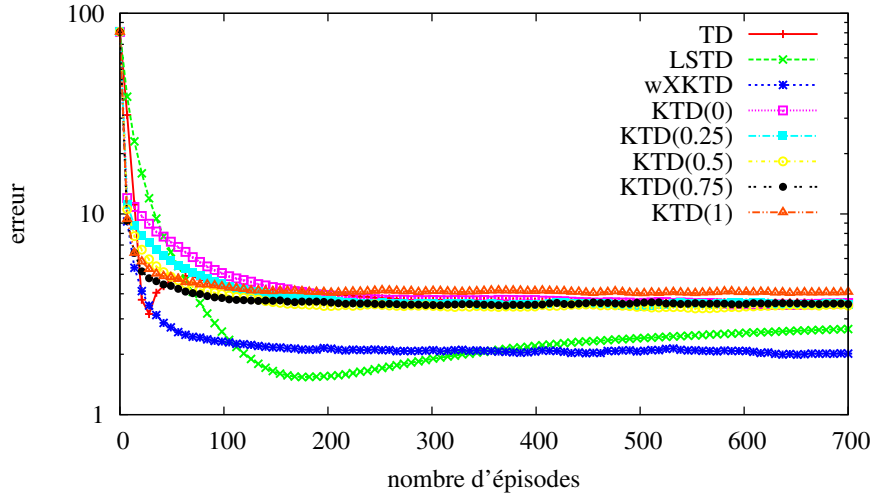


FIG. 5.4 – Chaîne contrôlée, itération optimiste de la politique.

Nous obtenons des résultats relativement cohérents, à savoir que les courbes de performance pour différentes valeurs de λ sont globalement encadrées par les courbes correspondant à $\lambda = 1$ (KTD) et $\lambda = 0$ (XKTD). Les meilleurs résultats sont toujours obtenus avec wXKTD[1]. Nous aurions pu espérer que pour certaines valeurs du facteur d'éligibilité KTD(λ) se comporte mieux que pour les valeurs extrêmes, mais cela ne semble pas être significativement le cas. Cependant, nous verrons pour l'expérience relative au *mountain*

[1]Il est intéressant de noter que wXKTD correspond finalement à une autre sorte de bruit coloré.

car stochastique que dans un contexte de contrôle il y a une amélioration. Remarquons que si nous n'avons pas proposé d'extension aux traces d'éligibilité de wXKTD, c'est que les problèmes de causalité du bruit dont on s'affranchit en supposant la variance des résidus constante pour KTD(λ) seraient plus importants dans le cadre d'une extension pondérée et donc plus difficiles à prendre en compte.

Optimisation directe de la Q-fonction

Ici nous nous intéressons au biais causé par l'aspect *off-policy* de l'optimisation directe de la Q-fonction (liée, nous le rappelons, à l'équation d'optimalité de Bellman). Etant donné qu'en jouant sur λ on peut jouer sur le caractère plus ou moins local des mises à jour, il est naturel de se demander si les traces d'éligibilité permettent de traiter, au moins partiellement, ce problème dû au caractère *off policy* de l'algorithme. Comparaison est faite au Q-learning et les paramètres sont les mêmes que précédemment, exception faite du bruit d'évolution qui est choisi nul. L'apprentissage est fait sur 100 épisodes et les résultats présentés figure 5.5 sont moyennés sur 100 essais. Notons l'échelle semi-logarithmique.

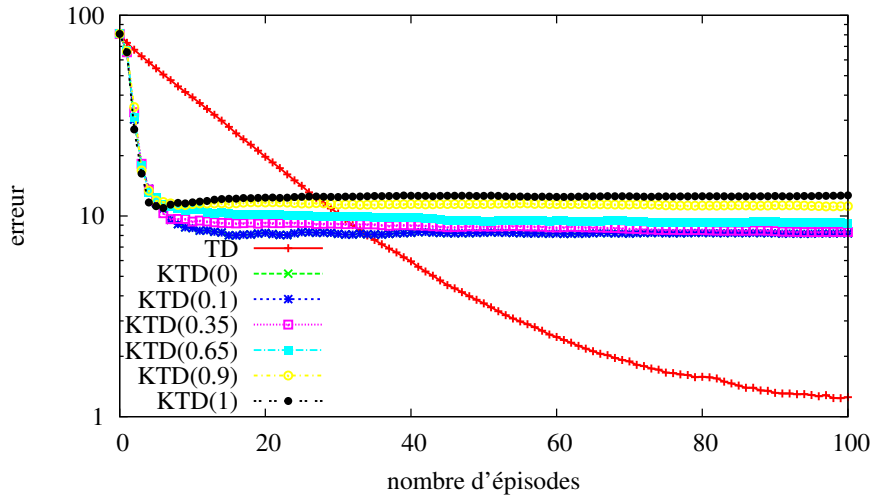


FIG. 5.5 – Chaîne contrôlée, optimisation directe de Q.

Là encore les courbes de performance pour différentes valeurs de λ sont globalement encadrées par les courbes correspondant aux valeurs extrêmes $\lambda = 0$ et $\lambda = 1$. L'extension de KTD aux traces d'éligibilité ne semble donc pas permettre de prendre en compte ce problème de biais causé par les l'aspect *off-policy* des algorithmes basés sur le principe de l'itération de la valeur, comme le Q-learning ou les variations de KTD-Q.

5.4.3 *Mountain-car* stochastique

Nous proposons ici de tester KTD(λ) sur le problème du *mountain-car* stochastique décrit section 4.5.3. L'objectif est encore de comparer les différents algorithmes dans un cadre d'itération optimiste de la politique, plus particulièrement pour une politique ϵ-gloutonne. La représentation de la Q-fonction et les paramètres des différents algorithmes considérés sont les mêmes que pour la section 4.5.3, la mesure de performance est toujours la qualité de la politique (mesurée avec le nombre de pas nécessaires pour sortir de la

cuvette). L'apprentissage se fait sur 200 épisodes et les résultats présentés sur la figure 5.6 sont moyennés sur 300 essais.

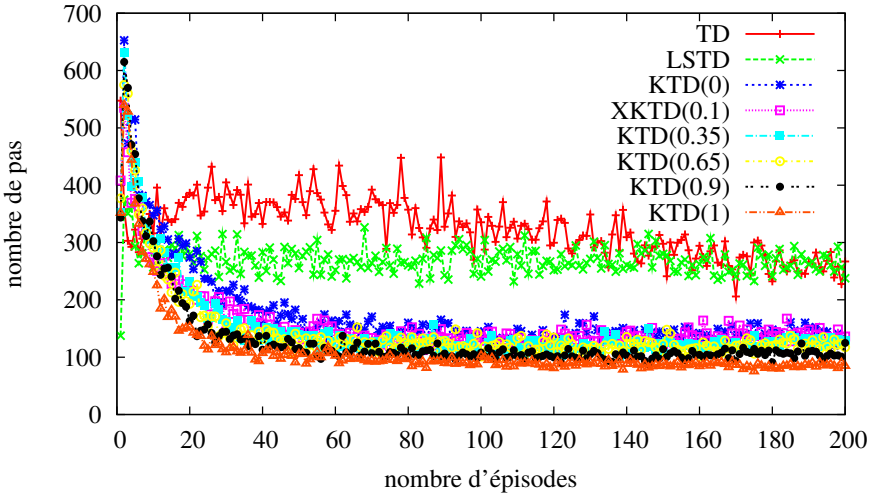


FIG. 5.6 – *Mountain Car* stochastique.

Les résultats obtenus sont toujours de même nature. Les courbes de performance pour différentes valeurs de λ sont globalement encadrées par les courbes correspondant aux valeurs extrêmes $\lambda = 0$ et $\lambda = 1$. Dans ce problème particulier, le biais causé par la stochasticité des transitions ne pose pas de problème du point de vue du contrôle et c'est KTD(1) qui présente les meilleurs résultats. Cependant, pour des valeurs de λ proches de 1 les performances sont très similaires et elles semblent plus stables (moins de variance) que pour l'algorithme wXKTD dont la performance est représentée sur la figure 4.7. Dans le cas où le biais introduit par les transitions stochastiques poserait problème d'un point de vue de contrôle (ce qui ne se vérifie pas vraiment sur cette expérience), il semblerait donc que KTD(λ) avec un facteur d'éligibilité proche de 1 soit une méthode de choix.

5.4.4 Un labyrinthe pathologique

Nous proposons dans cette section de tester les algorithmes sur un problème de labyrinthe pathologique illustré sur la figure 5.7. L'environnement est un labyrinthe de 9 cases. La case S en $(0,0)$ correspond à l'état de départ, les états T en $(1,1)$ et $(1,2)$ correspondent à des pièges et l'état G en $(2,1)$ correspond au but. La tâche est épisodique et se termine lorsque l'agent arrive au but ou sur l'un des pièges. Le critère est γ-pondéré avec $\gamma = 0.95$. Les récompenses sont $+10$ lorsque l'agent arrive en G, -100 lorsqu'il arrive en T et -1 partout ailleurs. Les actions sont aller en haut, en bas, à gauche ou à droite. Avec une probabilité $p = 0.9$, l'agent se dirige dans la direction souhaitée et avec une probabilité $1-p$ dans la direction perpendiculaire (sens trigonométrique, soit haut à la place de droite par exemple). La politique optimale est indiquée sur la figure 5.7. Notons que si l'on baisse trop la probabilité p (même en la maintenant au dessus de 0.5), cette politique n'est plus optimale.

Nous pensons cette tâche pathologique car dans les états $(1,0)$ et $(2,1)$, si l'action optimale est choisie, il y a une probabilité $1-p$ de tomber dans le piège. Il existe donc un risque que l'agent préfère rester coincé dans le haut du labyrinthe. La politique qui

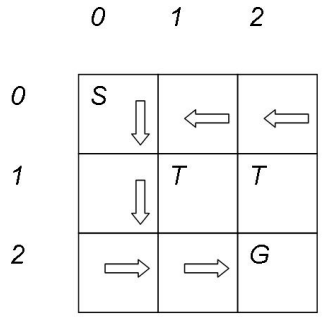


FIG. 5.7 – Labyrinthe pathologique.

consiste à rester coincé en haut présente un cumul pondéré de récompense de -20, celle que nous avons présentée comme optimale de -15 environ. L'apprentissage se fait sur 300 épisodes pour chaque essai et les résultats sont moyennés sur 300 essais. Tous les 30 épisodes l'apprentissage est figé et la politique gloutonne testée, le résultat étant une moyenne de 100 épisodes. Les algorithmes considérés sont TD, LSTD et KTD(λ) pour différentes valeurs de λ, dans un schéma ϵ-glouton, avec $\epsilon = 0.1$. Le taux d'apprentissage pour TD est fixé à $\alpha = 0.1$. L'*a priori* pour LSTD est KTD(λ) est choisi égal à $P_{0|0} = I$. La variance des résidus est choisie unitaire, c'est-à-dire $\sigma^2 = 1$. Enfin, pour KTD(λ), le bruit d'évolution adaptatif est considéré avec $\eta = 10^{-4}$. Les résultats sont présentés sur la figure 5.8.

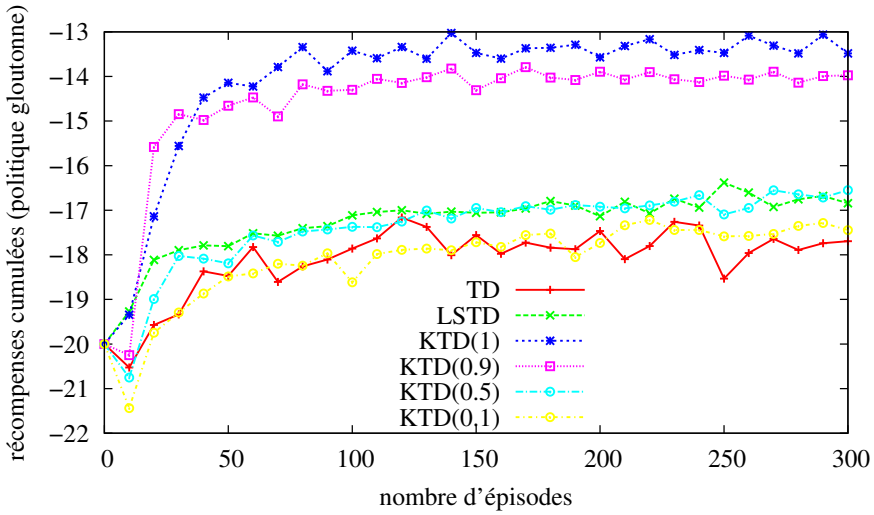


FIG. 5.8 – Labyrinthe pathologique : résultats.

Ce labyrinthe a été introduit dans l'idée que le biais induit par les transitions stochastiques poserait problème d'un point de vue contrôle à KTD(1). Ce n'est pas le cas et cet algorithme fournit les meilleurs résultats. Les algorithmes LSTD et TD ont tendance à moins bien fonctionner et à préférer rester coincé dans le haut du labyrinthe. Les performances de KTD(λ) décroissent lorsque λ tend vers 0. Nous supposons que l'effet mémoriel induit par le bruit coloré empêche la convergence rapide de l'algorithme dans ce cadre d'itération de la politique optimiste (nous supposons qu'asymptotiquement l'algorithme converge, pour tout λ). Là encore choisir une valeur de λ proche de 1 semble préférable.

5.5 Bilan

Dans ce chapitre, nous avons introduit une extension des différences temporelles de Kalman inspirée du concept des traces d'éligibilité. L'approche est cependant sensiblement différente, dans la mesure où nous ne considérons pas de traces à proprement parler, mais nous introduisons une nouvelle classe de bruits colorés paramétrée par le facteur d'éligibilité. Cela résulte du fait que plutôt que de considérer les erreurs de différences temporelles d'ordre k, ce sont les bruits correspondants d'ordre k (étant donné notre vision probabiliste) que nous prenons en compte. L'algorithme résultant, KTD(λ), présente comme cas particuliers les algorithmes des chapitres 3 (avec $\lambda = 1$) et 4 (avec $\lambda = 0$). Le coût quadratique que tend à minimiser ce nouvel estimateur a également été présenté. Nous avons montré empiriquement que l'introduction de ce bruit coloré permet de réduire le biais de KTD(1) et permet également de réduire la variance de l'estimateur, selon les valeurs choisies pour le facteur d'éligibilité. De plus, KTD(λ) permet de réduire le caractère global des transitions de XKTD. Cependant, KTD(λ) avec un facteur d'éligibilité différent de 1 ne semble pas permettre de prendre en compte l'aspect *off-policy* de l'apprentissage : si les mises à jour sont moins globales, elles ne sont toutefois pas purement locales. D'autre part, d'un point de vue contrôle, si l'extension aux traces d'éligibilité proposée permet d'améliorer les résultats par rapport à XKTD, sur les expériences que nous avons menées c'est toujours KTD qui fournit le meilleur contrôle. Nous pensons que cela vient principalement du fait que nous n'avons pas trouvé d'expérience pour laquelle le biais causé par le caractère stochastique des transitions pose problème à KTD(1) du point de vue de la qualité du contrôle. Cependant, pour un tel problème, nous pensons que choisir KTD(λ) avec λ proche de 1 permettrait à la fois de s'affranchir du problème de biais, tout en conservant un caractère relativement local des mises à jour, c'est-à-dire en conservant les capacités d'adaptation de l'algorithme.

Chapitre 6

Utilisation de l'information d'incertitude : apprentissage actif et dilemme entre exploration et exploitation

Dans le cadre de travail proposé, les paramètres étant modélisés par un vecteur aléatoire et la fonction de valeur paramétrée étant pour un état donné fonction de ces paramètres, c'est une variable aléatoire. Nous montrons dans un premier temps comment calculer son espérance et l'incertitude associée (sa variance) à l'aide de la transformation non-parfumée. Le dilemme entre exploration et exploitation, qui est l'une des grandes problématiques de l'apprentissage par renforcement, peut grandement bénéficier d'une telle information d'incertitude, même si ce n'est pas *a priori* une condition nécessaire pour le traiter. Si peu d'approches dans la littérature permettent à la fois d'approximer la fonction de valeur et une incertitude associée, on peut tout de même citer [27], qui ne considère pourtant son utilisation qu'en termes de perspectives. Dans un second temps, nous proposons une approche d'apprentissage actif qui est en quelque sorte une forme de politique totalement exploratrice choisissant les actions en fonction de leurs incertitudes, dans le contexte de KTD-Q. Il est montré sur un exemple section 6.4 que cela permet effectivement d'accélérer l'apprentissage. Nous proposons également un certain nombre d'heuristiques utilisant cette information d'incertitude dans le but de traiter le dilemme entre exploration et exploitation. Les politiques associées sont inspirées et adaptées d'autres travaux, qui cependant ne traitent souvent que le cas tabulaire (c'est-à-dire qu'une représentation exacte de la fonction de valeur est possible). Nous expérimentons ces différents schémas sur un problème de bandits, toujours dans la section 6.4.

6.1 Calcul de l'incertitude

Nous nous intéressons ici à la fonction de valeur, l'extension des résultats présentés à la fonction de qualité étant évidente. Supposons l'estimation de la fonction de valeur $\hat{V}_\theta$ paramétrée par le vecteur aléatoire θ de moyenne $\bar{\theta}$ et de variance P_θ. Notons $\hat{\mu}_{V_\theta}(s)$ son espérance et $\hat{\sigma}^2_{V_\theta}(s)$ sa variance. Si la paramétrisation est linéaire, de la forme $\hat{V}_\theta(s) = \phi(s)^T\theta$ comme dans l'équation (3.21), alors les quantités d'intérêt peuvent être obtenues

analytiquement :

$$\hat{\mu}_{V_\theta}(s) = \phi(s)^T \bar{\theta} \quad \text{et} \quad \hat{\sigma}^2_{V_\theta}(s) = \phi(s)^T P_\theta \phi(s) \tag{6.1}$$

Si la paramétrisation est non-linéaire, il est toujours possible de propager l'incertitude sur les paramètres à la fonction de valeur en appliquant la transformation non-parfumée. Dans un premier temps, l'ensemble des sigma-points Θ et les poids associés $\mathcal{W}$ sont calculés à partir de $\bar{\theta}$ et P_θ, comme expliqué dans la section 3.4 :

$$\Theta = \{\theta^{(j)},\, 0 \leq j \leq 2p\} \tag{6.2}$$

$$\mathcal{W} = \{w_j,\, 0 \leq j \leq 2p\} \tag{6.3}$$

Les images de ces sigma-points par la fonction de valeur, pour un état donné, sont ensuite calculées :

$$\mathcal{V}(s) = \left\{\hat{V}^{(j)}(s) = \hat{V}_{\theta^{(j)}}(s), \quad 0 \leq j \leq 2p\right\} \tag{6.4}$$

Enfin, connaissant ces images et les poids associés, il est possible d'approcher la valeur moyenne ainsi que la variance associée :

$$\hat{\mu}_V(s) = \sum_{j=0}^{2p} w_j \hat{V}^{(j)}(s) \tag{6.5}$$

$$\hat{\sigma}^2_V(s) = \sum_{j=0}^{2p} w_j \left(\hat{V}^{(j)}(s) - \hat{V}(s)\right)^2 \tag{6.6}$$

Donc, pour une représentation de la fonction de valeur modélisée par un vecteur aléatoire, il est aisé de calculer l'information d'incertitude associée pour tout état s. La complexité est là encore quadratique. Ainsi, comme à chaque pas de temps une estimation $\hat{\theta}_{i|i}$ et la variance associée $P_{i|i}$ sont disponibles, l'incertitude sur les paramètres peut être propagée à la fonction de valeur (ou de qualité), si nécessaire. Nous illustrons ce calcul de l'incertitude (à l'aide de la transformation non-parfumée) sur la figure 6.1.

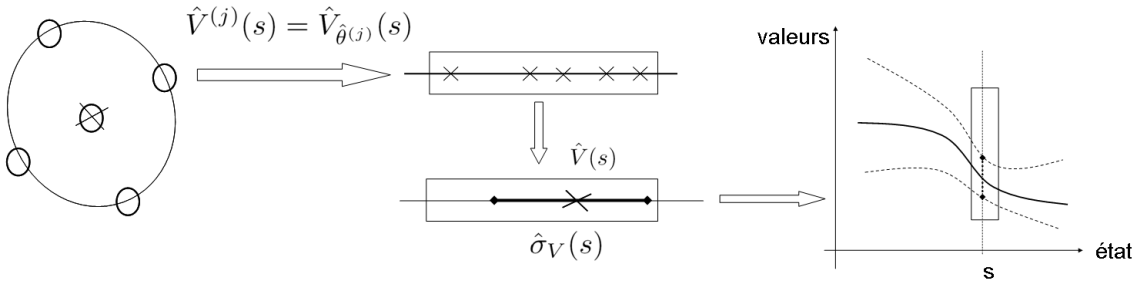


FIG. 6.1 – Calcul de l'incertitude.

Nous pensons important de préciser que la variance calculée est l'incertitude sur l'estimation de la valeur d'un état et pas une estimation de la variance du processus aléatoire dont la fonction de valeur est la moyenne (processus que nous avons noté D^π dans le chapitre 4). Cependant, si cette information de variance du processus D^π est disponible *a priori*, elle peut servir à dimensionner le bruit d'observation.

6.2 Apprentissage actif

Nous donnons ici un exemple d'utilisation effective de l'information d'incertitude disponible. Nous nous intéressons à KTD-Q. Cet algorithme est dit *off-policy*, car la politique apprise π (la politique optimale π^* dans ce cas) est différente de la politique suivie ou comportementale (que nous noterons b). Une question naturelle est de savoir quelle politique comportementale permet l'apprentissage le plus rapide de la politique optimale. Un élément de réponse est proposé ici.

Soit i l'index temporel courant. Le système à contrôler est dans un état s_i et l'agent doit choisir une action a_i. L'algorithme considéré étant KTD-Q, les estimations $\theta_{i-1|i-1}$ et $P_{i-1|i-1}$ sont disponibles. Elles peuvent être utilisées pour approcher l'incertitude de la Q-fonction paramétrée par θ_{i-1} en l'état s_i pour chaque action a, comme expliqué section 6.1. La variance associée est notée $\hat{\sigma}^2_{Q_{\theta_{i-1}}}(s_i, a)$. L'action a_i est ensuite choisie selon la politique comportementale aléatoire b définie par :

$$b(a_i|s_i) = \frac{\hat{\sigma}_{Q_{\theta_{i-1}}}(s_i, a_i)}{\sum_{a \in A} \hat{\sigma}_{Q_{\theta_{i-1}}}(s_i, a)} \tag{6.7}$$

Une politique totalement exploratrice privilégiant les actions pour lesquelles l'incertitude est la plus grande est ainsi obtenue. Ce n'est qu'une façon parmi d'autres d'utiliser cette information d'incertitude, mais elle permet de montrer que la variance sur les valeurs obtenues dans le contexte de KTD a du sens. Nous l'illustrons section 6.4 en montrant le gain en vitesse d'apprentissage obtenu par rapport à une politique comportementale totalement aléatoire.

La mise en œuvre pratique de cette approche est simple. Au temps i, après la phase de prédiction, nous disposons des statistiques $\hat{\theta}_{i|i-1}$ et $P_{i|i-1}$. Ces statistiques sont utilisées pour calculer les sigma-points, nécessaires à la prédiction de la récompense et donc à la mise à jour des moments d'ordre un et deux du vecteur aléatoire de paramètres. Ces mêmes sigma-points peuvent être utilisés pour calculer l'incertitude relative à chaque action, de façon à déterminer la politique b et à choisir une action. L'approche que nous appelons KTD-Q actif est résumée dans l'algorithme 6.1. La complexité algorithmique de KTD-Q actif est la même que celle de KTD-Q, soit $O(\mathcal{A}p^2)$ (moyennes et variances sont calculées pour chaque action, ce qui ne modifie pas la complexité globale).

6.3 Dilemme exploration/exploitation

Nous rappelons que dans le paradigme de l'apprentissage par renforcement, l'agent doit contrôler le système tout en apprenant le contrôle optimal et ce sans connaissance *a priori* du modèle. Se pose alors le dilemme entre exploration et exploitation : l'agent doit-il choisir une action qu'il juge optimale par rapport à sa connaissance (imparfaite) du monde, c'est-à-dire une action exploitante (action gloutonne), ou va-t-il choisir une action qu'il juge sous-optimale par rapport à sa connaissance actuelle du monde, c'est-à-dire une action exploratrice, de façon à améliorer sa connaissance du monde ?

A chaque instant nous disposons de la moyenne $\bar{\theta}$ et de la variance P_θ du vecteur aléatoire θ qui représente la fonction de valeur. Nous nous intéressons à la Q-fonction, dans la mesure où l'on ne souhaite pas apprendre le modèle. Dans ce cas, nous avons vu dans la section 6.1 comment disposer à chaque instant des informations de moyenne et de variance.

Algorithme 6.1 : KTD-Q actif

Initialisation : *a priori* $\hat{\theta}_{0|0}$ et $P_{0|0}$, état s_1;

pour $i \leftarrow 1, 2, \ldots$ **faire**

Phase de prédiction;

$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;

$P_{i|i-1} = P_{i-1|i-1} + P_{v_i}$;

Calcul des Sigma-points;

$\Theta_{i|i-1} = \{\hat{\theta}^{(j)}_{i|i-1}, \quad 0 \leq j \leq 2p\}$;

$\mathcal{W} = \{w_j, \quad 0 \leq j \leq 2p \quad \}$;

Calcul des valeurs moyennes et écarts-type associés;

pour $a \in A$ **faire**

$\mathcal{Q}_{i|i-1}(s_i, a) = \{\hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a), 0 \leq j \leq 2p\}$;

$\hat{\mu}_{Q_{i|i-1}}(s_i, a) = \sum_{j=0}^{2p} w_j \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a)$;

$\hat{\sigma}^2_{Q_{i|i-1}}(s_i, a) = \sum_{j=0}^{2p} w_j \left(\hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a) - \hat{\mu}_{Q_{i|i-1}}(s_i, a) \right)^2$;

Génération d'une action et transition;

Générer l'action a_i selon la politique $b(.|s_i)$, voir équation (6.7);

Observer la récompense r_i et l'état s_{i+1} ;

Calcul des images Sigma-points;

$\mathcal{R}_{i|i-1} = \{\hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i) - \gamma \max_{a \in A} \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, a), 0 \leq j \leq 2p\}$;

Calcul des statistiques d'intérêt;

$\hat{r}_{i|i-1} = \sum_{j=0}^{2p} w_j \hat{r}^{(j)}_{i|i-1}$;

$P_{\theta r_i} = \sum_{j=0}^{2p} w_j (\hat{\theta}^{(j)}_{i|i-1} - \hat{\theta}_{i|i-1})(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})$;

$P_{r_i} = \sum_{j=0}^{2p} w_j (\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})^2 + P_{n_i}$;

Phase de correction;

$K_i = P_{\theta r_i} P_{r_i}^{-1}$;

$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i(r_i - \hat{r}_{i|i-1})$;

$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

Pour résumer, nous avons :

$$\theta \sim \left(\bar{\theta}, P_\theta\right) \text{ et } \hat{Q}_\theta(s,a) \sim \left(\hat{\mu}_{Q_\theta}(s,a), \hat{\sigma}^2_{Q_\theta}(s,a)\right) \tag{6.8}$$

Dans cette section nous proposons un certain nombre d'heuristiques inspirées d'approches de la littérature et utilisant l'information d'incertitude disponible. Indépendamment du type de politique considérée, la mise en œuvre est relativement simple et très semblable à ce qui a été fait pour KTD-Q actif. A chaque instant i, après la phase de prédiction les statistiques $\hat{\theta}_{i|i-1}$ et $P_{i|i-1}$ sont disponibles et utilisées pour calculer les sigma-points. Si nécessaire (cela dépend du schéma considéré), la valeur moyenne et la variance associée de chaque action peut être calculée pour déterminer une politique. Cette dernière est utilisée pour générer une action à appliquer au système. Le reste des étapes se déroule comme précédemment. L'approche, très générique, est résumée dans l'algorithme 6.2 (nous considérons de façon générale KTD(λ), étant donné que l'approche est *on-policy*, contrairement à KTD-Q actif). Notons que la complexité associée est d'au plus $O(\mathcal{A}p^2)$, comme KTD-Q actif (moyennes et variances doivent être calculées pour chaque action et les éléments nécessaires au calcul des différentes politiques ont une complexité en $O(1)$, sauf le cas de la politique de Thompson qui est discuté par la suite).

6.3.1 Politique ϵ-gloutonne

Nous avons déjà présenté (et utilisé) la politique ϵ-gloutonne [105]. Elle consiste à choisir l'action gloutonne (respectivement à la fonction de qualité) avec une probabilité $1-\epsilon$ et une action totalement aléatoire avec une probabilité ϵ. Cette politique n'utilise aucunement l'information d'incertitude disponible. Cependant, elle va nous servir de référence.

Définition 6.1 (Politique ϵ-gloutonne). *Une politique ϵ-gloutonne tire une action aléatoire avec une probabilité ϵ, une action gloutonne sinon.*

$$a \sim \pi(.|s) = \begin{cases} a = \text{argmax}_{b \in A}\, \hat{\mu}_{Q_\theta}(s,b) \text{ avec une probabilité } 1-\epsilon \\ a \sim \mathcal{U}_A \text{ avec une probabilité } \epsilon \end{cases} \tag{6.9}$$

Dans cette définition, $\mathcal{U}_A$ désigne une loi uniforme sur l'espace des actions.

Nous illustrons le principe d'une telle politique sur la figure 6.2. Pour cela nous introduisons 4 actions, dont les Q-valeurs estimées ainsi que les écarts-type associés sont illustrés sur la figure de gauche (Q-valeur moyenne $\pm$ écart-type associé). Les valeurs choisies sont arbitraires, elles ne correspondent pas à un problème particulier, l'objectif étant ici d'illustrer les différentes approches du dilemme entre exploration et exploitation que nous proposons dans ce chapitre. Sur la figure de droite nous illustrons la politique ϵ-gloutonne par sa densité de probabilités. L'action de plus grande valeur a la plus grande probabilité, les autres actions sont équi-probables. L'information d'incertitude n'est pas utilisée ici.

6.3.2 Politique ϵ-gloutonne informée

Cette politique est très similaire à la politique ϵ gloutonne. L'unique différence réside dans le fait que plutôt que de choisir une action selon une loi uniforme (exploration non-dirigée) avec une probabilité ϵ, c'est une action selon la loi définie pour l'apprentissage actif qui sera tirée (exploration dirigée). L'idée ici est que plutôt que de tirer une action totalement aléatoire, il est plus intéressant de privilégier les zones les plus incertaines pour gagner plus d'information.

Algorithme 6.2 : KTD(λ) et contrôle

Initialisation : *a priori* $\hat{\mathbf{x}}_{0|0}$ et $P_{0|0}$, état s_1;

pour $i \leftarrow 1, 2, \ldots$ **faire**

Phase de prédiction;

$\hat{\mathbf{x}}_{i|i-1} = F(\lambda)\hat{\mathbf{x}}_{i-1|i-1}$;

$P_{i|i-1} = F(\lambda)P_{i-1|i-1}F(\lambda)^T + P_{v'_{i-1}}$;

Calcul des Sigma-points;

$\mathbf{X}_{i|i-1} = \left\{\hat{\mathbf{x}}^{(j)}_{i|i-1}, \quad 0 \leq j \leq 2p+4\right\}$ (en utilisant $\hat{\mathbf{x}}_{i|i-1}$ et $P_{i|i-1}$);

$\mathcal{W} = \{w_j, \quad 0 \leq j \leq 2p+4 \quad\}$;

`/* notons que` $(\hat{\mathbf{x}}^{(j)}_{i|i-1})^T = \left((\hat{\theta}^{(j)}_{i|i-1})^T \quad \hat{b}^{(j)}_{i|i-1} \quad \hat{n}^{(j)}_{i|i-1}\right)$ `*/`

Calcul des valeurs moyennes et écarts-type associés;

`/* Cette phase n'est pas systématiquement nécessaire, elle dépend du schéma considéré */`

pour $a \in A$ **faire**

$\mathcal{Q}_{i|i-1}(s_i, a) = \{\hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a), 0 \leq j \leq 2p+4\}$;

$\hat{\mu}_{Q_{i|i-1}}(s_i, a) = \sum_{j=0}^{2p+4} w_j \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a)$;

$\hat{\sigma}^2_{Q_{i|i-1}}(s_i, a) = \sum_{j=0}^{2p+4} w_j \left(\hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a) - \hat{\mu}_{Q_{i|i-1}}(s_i, a)\right)^2$;

Caractérisation de la politique;

Déterminer la politique π, voir sections 6.3.1-6.3.7 ;

Génération d'une action et transition;

Générer l'action a_i selon la politique $\pi(.|s_i)$;

Observer la récompense r_i et l'état s_{i+1} ;

Calcul des images Sigma-points;

$\mathcal{R}_{i|i-1} = \left\{\hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i) - \gamma \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, a_{i+1}) + \hat{n}^{(j)}_{i|i-1}, 0 \leq j \leq 2p+4\right\}$;

Calcul des statistiques d'intérêt;

$\hat{r}_{i|i-1} = \sum_{j=0}^{2p+4} w_j \hat{r}^{(j)}_{i|i-1}$;

$P_{\mathbf{x}r_i} = \sum_{j=0}^{2p+4} w_j (\hat{\mathbf{x}}^{(j)}_{i|i-1} - \hat{\mathbf{x}}_{i|i-1})(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})$;

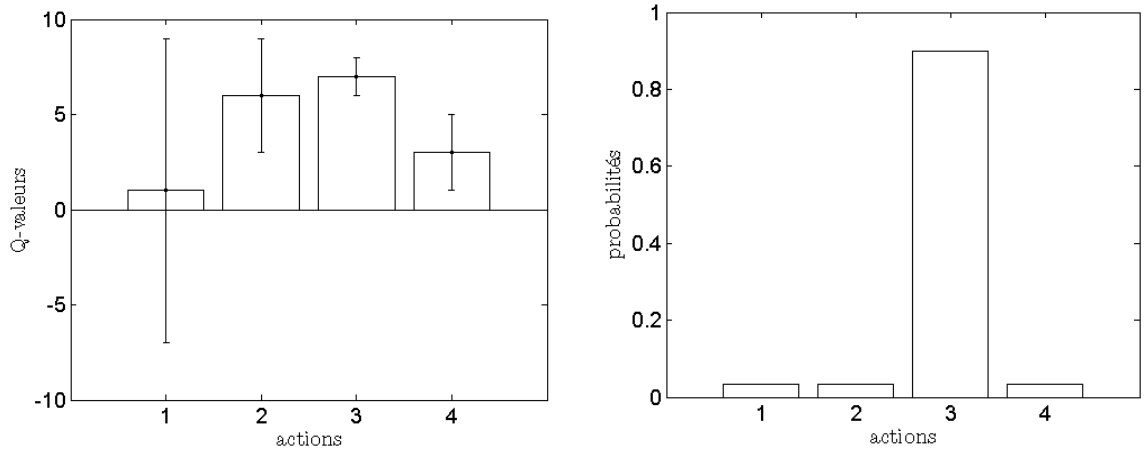
$P_{r_i} = \sum_{j=0}^{2p+4} w_j \left(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1}\right)^2$;

Phase de correction;

$K_i = P_{\mathbf{x}r_i} P_{r_i}^{-1}$;

$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_{i|i-1} + K_i \left(r_i - \hat{r}_{i|i-1}\right)$;

$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;



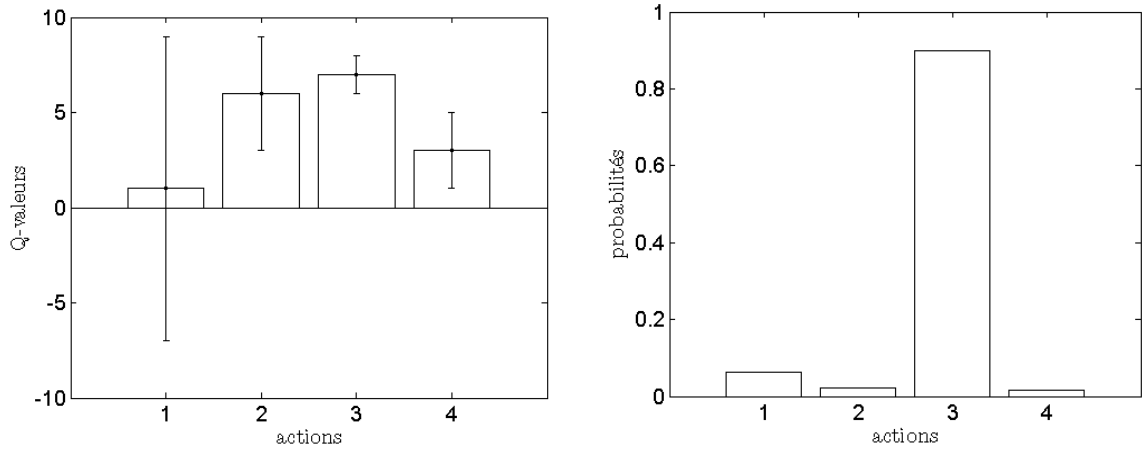
a. Q-valeurs. b. Politique ϵ-gloutonne.

FIG. 6.2 – Politique ϵ-gloutonne illustrée.

Définition 6.2 (Politique ϵ-gloutonne informée). *Une politique ϵ-gloutonne informée tire une action gloutonne avec une probabilité $1-\epsilon$, une action aléatoire sinon, les probabilités étant pondérées par l'incertitude des valeurs.*

$$a \sim \pi(.|s) = \begin{cases} a = \text{argmax}_{b \in A}\, \hat{\mu}_{Q_\theta}(s,a) \text{ avec une probabilité } 1-\epsilon \\ a \sim \mathcal{W}_A \text{ avec une probabilité } \epsilon \end{cases} \tag{6.10}$$

$$\text{avec } \mathcal{W}_A : \pi(a|s) = \frac{\hat{\sigma}_{Q_\theta}(s,a)}{\sum_{b \in A} \hat{\sigma}_{Q_\theta}(s,b)} \tag{6.11}$$



a. Q-valeurs. b. Politique ϵ-gloutonne informée.

FIG. 6.3 – Politique ϵ-gloutonne informée illustrée.

Sur la figure 6.3 nous illustrons une telle politique informée. Les Q-valeurs estimées représentées sur la partie gauche sont toujours les mêmes, elles le resteront pour toutes les politiques considérées. C'est toujours l'action de plus grande valeur qui est choisie avec la plus forte probabilité. Cependant, les autres actions ne sont plus équiprobables, mais pondérée par l'écart-type relatif. Ainsi c'est la première action qui présente la probabilité

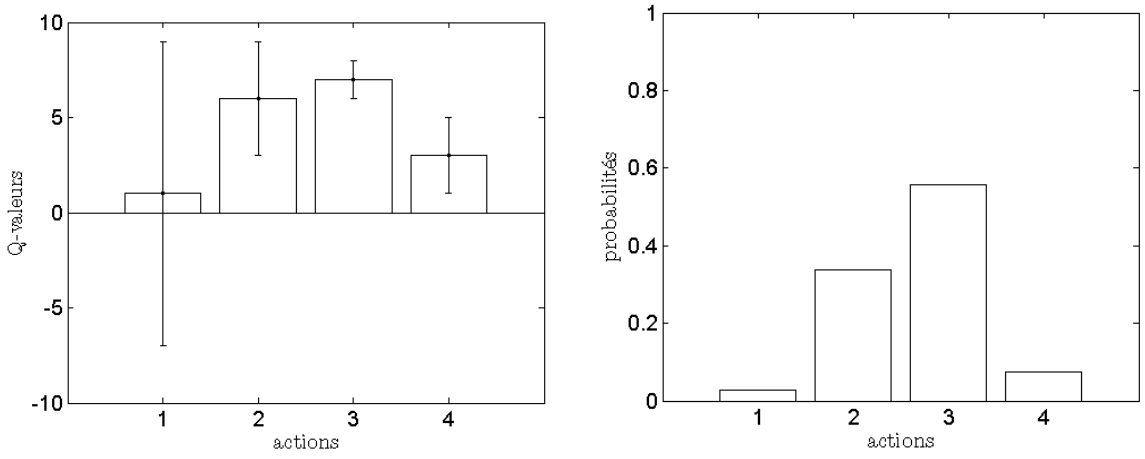
la plus forte (parmi les actions non-gloutonnes), car c'est celle qui possède la plus grande incertitude.

6.3.3 Politique Bayes-gloutonne

Rappelons tout d'abord qu'une politique de Gibbs génère des actions avec une probabilité d'autant plus grande que la valeur associée est importante :

$$\pi(a|s) = \frac{\exp\left(\frac{\hat{\mu}_{Q_\theta}(s,a)}{\tau}\right)}{\sum_{b\in A}\exp\left(\frac{\hat{\mu}_{Q_\theta}(s,b)}{\tau}\right)} \tag{6.12}$$

Le facteur τ est un facteur de température qui rend les actions d'autant plus équiprobables qu'il est grand. Nous avons déjà discuté ce type de politique dans la section 2.2.3 et nous l'illustrons figure 6.4. Les Q-valeurs considérées sont toujours les mêmes et nous représentons la distribution de probabilités qui définit la politique. La probabilité de choisir une action est d'autant plus grande que la Q-valeur associée est élevée, ce type de politique n'utilise pas l'information d'incertitude disponible. La première action, qui a le potentiel d'être intéressante dans la mesure où son estimation est peu certaine, est ainsi quasiment ignorée.



a. Q-valeurs. b. Politique de Gibbs.

FIG. 6.4 – Politique de Gibbs illustrée.

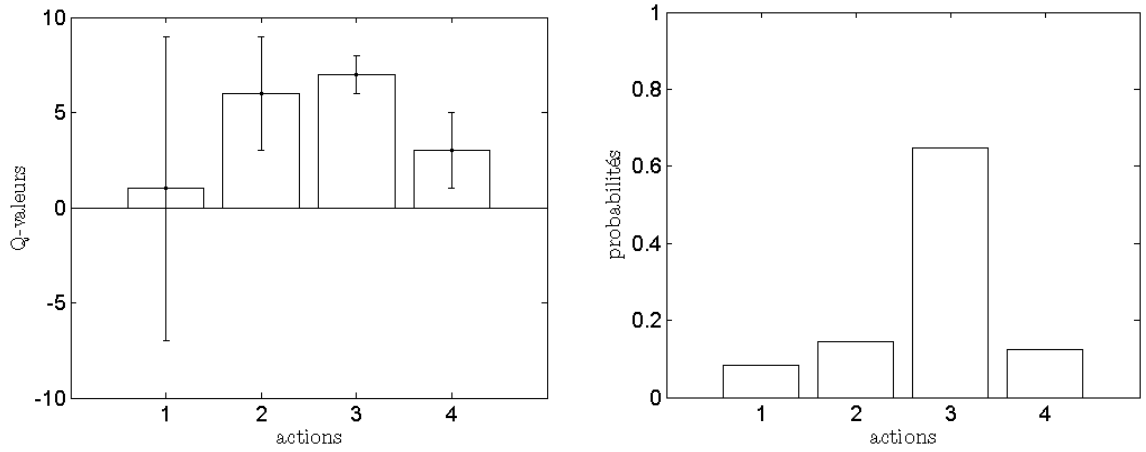
L'idée basique d'une politique Bayes-gloutonne est d'adapter une politique de Gibbs pour laquelle on remplace le facteur de température par l'incertitude sur le couple état-action courant. Cette approche a été proposée par [93], où l'incertitude est modélisée par un "index de fiabilité" (basiquement la racine carré d'une moyenne glissante des carrés des erreurs de différences temporelles). Dans notre cas, nous utilisons l'écart-type associé à la Q-fonction pour les couples état-action considérés. Le nom "Bayes-glouton" a été donné dans [25] (cependant l'incertitude y est obtenue différemment).

Définition 6.3 (Politique Bayes-gloutonne). *Soit ν un paramètre libre, une politique Bayes-gloutonne est une politique de Gibbs pour laquelle le paramètre de température est*

remplacé par une information locale d'incertitude. Elle est définie par :

$$\pi(a|s) = \frac{exp(\nu \frac{\hat{\mu}_{Q_\theta}(s,a)}{\hat{\sigma}_{Q_\theta}(s,a)})}{\sum_{b \in A} exp(\nu \frac{\hat{\mu}_{Q_\theta}(s,b)}{\hat{\sigma}_{Q_\theta}(s,b)})} \tag{6.13}$$

Le paramètre ν est libre, il permet de mettre à l'échelle l'écart-type de la qualité d'un couple état-action donné (par rapport à l'incertitude sur les paramètres). Nous le discuterons plus dans la section expérimentale.



a. Q-valeurs. b. Politique Bayes-gloutonne.

FIG. 6.5 – Politique Bayes-gloutonne illustrée.

Nous représentons la densité de probabilités définissant une telle politique sur la figure 6.5, pour toujours les même Q-valeurs. L'information d'incertitude est maintenant prise en compte, elle favorise les actions dont le résultat est le plus certain. C'est toujours l'action gloutonne qui a le plus de chances d'être choisie. Cependant, il y a moins de différences entre les deuxième et quatrième actions et la première action a plus de chance d'être choisie qu'elle ne l'était avec une politique de Gibbs.

6.3.4 Estimation d'intervalle de confiance

Le principe de ce type d'approche est de maintenir non seulement une estimation de la Q-valeur pour chaque couple état-action, mais également un intervalle de confiance. La politique associée est gloutonne respectivement à la borne supérieure de cet intervalle de confiance. L'approche est relativement ancienne [61], mais récemment des garanties de type PAC (probablement approximativement correct) ont été données, sous certaines conditions [103, 101, 102]. Cela signifie qu'il est possible de montrer que le nombre de pas de temps pendant lesquels la politique n'est pas quasi-optimale est polynomial en les quantités d'intérêt du système et ce avec une probabilité proche de un[1]. Sans autres connaissances *a priori*, nous supposons la distribution des valeurs gaussienne, l'intervalle de confiance est donc proportionnel à l'écart-type.

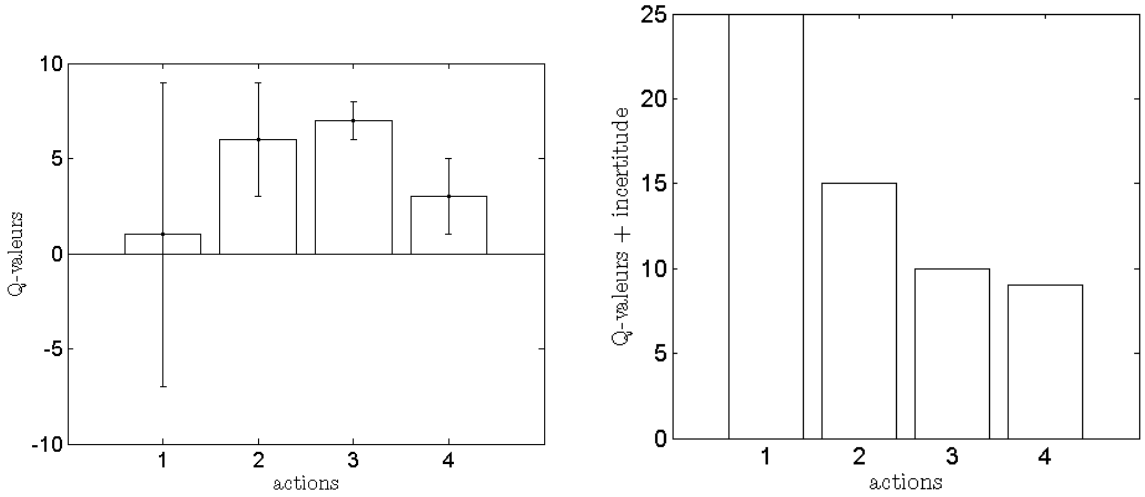
Définition 6.4 (Politique confiante-gloutonne). *Soit α un paramètre libre (typiquement compris entre un et trois). Une politique confiante-gloutonne choisit les actions de*

[1]Par contre ce type de résultat ne donne aucune indication sur quand la politique n'est pas quasi-optimale.

façon gloutonne par rapport à la borne supérieure de l'intervalle de confiance.

$$\pi(a|s) = \underset{b \in A}{\operatorname{argmax}}(\hat{\mu}_{Q_\theta}(s,b) + \alpha \hat{\sigma}_{Q_\theta}(s,b)) \tag{6.14}$$

Le paramètre α dépend de l'intervalle de confiance que l'on choisit. Par exemple (toujours en supposant la distribution gaussienne), une valeur de 3 pour α spécifie un intervalle de confiance à 95%. C'est pour cela que nous recommandons des valeurs comprises entre un et trois. Cependant, comme nous allons le voir dans la section expérimentale, l'évolution de la variance dépend beaucoup des paramètres de l'algorithme et l'on peut être amené à choisir des valeurs plus extrêmes (dans un sens comme dans l'autre, mais en restant positif).



a. Q-valeurs. b. Politique confiante-gloutonne.

FIG. 6.6 – Politique confiante-gloutonne illustrée.

Nous représentons une telle politique sur la figure 6.6. Elle est gloutonne par rapport à la borne supérieure de l'intervalle de confiance, ce qui est ce que nous représentons à droite (intervalle de confiance à 95%). Notons que la politique n'est pas stochastique, mais déterministe, dans la mesure où elle est gloutonne par rapport à une transformation des Q-valeur estimées et des incertitudes associées. Dans cet exemple, c'est la première action qui est choisie.

6.3.5 Bonus d'exploration

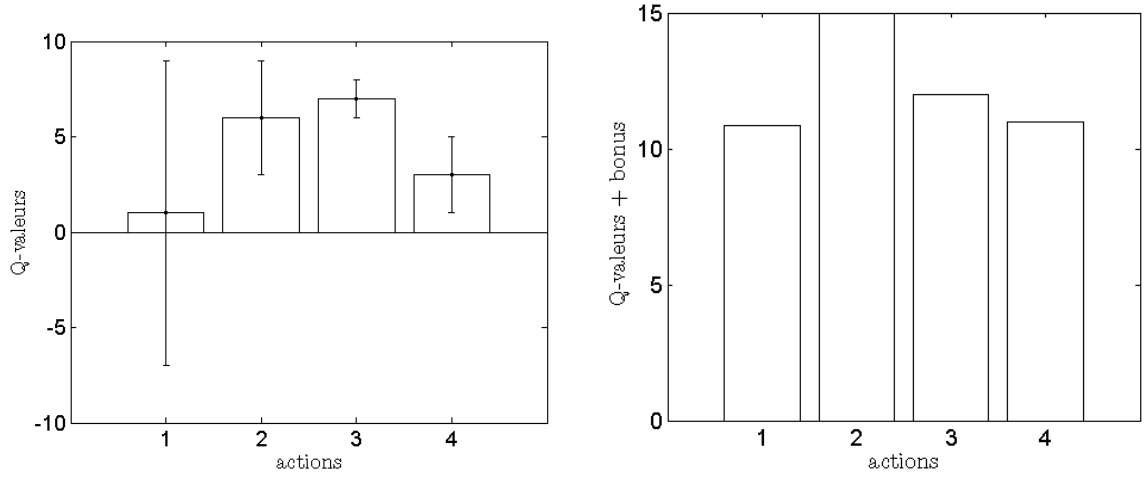
Il existe un certain nombre d'approches qui ajoutent un bonus à la Q-fonction estimée pour favoriser l'exploration. Par exemple, les politiques gloutonnes par rapport à la borne supérieure d'un intervalle de confiance entrent dans cette catégorie, le bonus étant proportionnel à l'écart-type. D'autres approches sont basées sur un principe similaire (comme E^3 [66], R_{max} [20] ou encore MBIE-EB [103]), dont le principe est globalement d'ajouter un bonus d'autant plus grand que la paire état-action considérée aura été visitée peu de fois. Par exemple, pour MBIE-EB, ce bonus est inversement proportionnel à la racine du nombre de visites du couple état-action considéré. Ce type d'approche n'est pas évident à adapter aux cas nécessitant une approximation de la fonction de valeur (cela peut être fait, voir par exemple [75] qui présente une combinaison de LSPI [73] et R_{max}, mais cela suppose au moins une quantification de l'espace d'état). L'approche que nous proposons ici

utilise la variance de la valeur associée au couple état-action courant, elle est plus proche de [70] (qui utilise le nombre de visites plutôt que sa racine carrée).

Définition 6.5 (Politique bonus-gloutonne). *Soient β_0 et β deux paramètres libres. Une politique bonus-gloutonne est gloutonne par rapport à la fonction de qualité estimée plus un bonus dépendant de la variance :*

$$\pi(a|s) = \underset{b \in A}{\operatorname{argmax}} \left(\hat{\mu}_{Q_\theta}(s,b) + \beta \frac{\hat{\sigma}^2_{Q_\theta}(s,b)}{\beta_0 + \hat{\sigma}^2_{Q_\theta}(s,b)} \right) \tag{6.15}$$

Ce bonus, défini par $\beta \frac{\hat{\sigma}^2_{Q_\theta}(s,b)}{\beta_0 + \hat{\sigma}^2_{Q_\theta}(s,b)}$, sera d'autant plus proche de β que la variance sera importante et d'autant plus proche de 0 que la variance sera faible. On obtient bien l'effet recherché, c'est-à-dire ajouter le plus de bonus aux zones les moins visitées. Le choix des paramètres β_0 et β est libre, mais là encore il serait judicieux de le faire en fonction des paramètres de l'algorithme (de façon par exemple à calibrer l'amplitude du bonus lors des premières décisions, ce qui peut être fait en connaissant l'*a priori*).



a. Q-valeurs. b. Politique bonus-gloutonne.

FIG. 6.7 – Politique bonus-gloutonne illustrée.

Nous illustrons une politique bonus-gloutonne sur la figure 6.7, pour toujours les mêmes Q-valeurs. Là encore la politique n'est pas stochastique mais déterministe, dans la mesure où elle est gloutonne par rapport aux valeurs estimées auxquelles on a ajouté un bonus basé sur l'incertitude de ces estimations, c'est ce que nous représentons. C'est donc la deuxième action qui est choisie et les trois autres ont des valeurs proches (en tenant compte du bonus). Notamment, presque autant d'importance est accordée à la première action, qui a une valeur faible avec peu de certitude, qu'à la troisième action, qui a la valeur la plus forte avec moins d'incertitude.

6.3.6 Politique de Thompson

Etant donné que l'on a accès à la moyenne et à la matrice de variance des paramètres et si l'on fait une supposition sur leur distribution, une hypothèse gaussienne en l'occurrence[2],

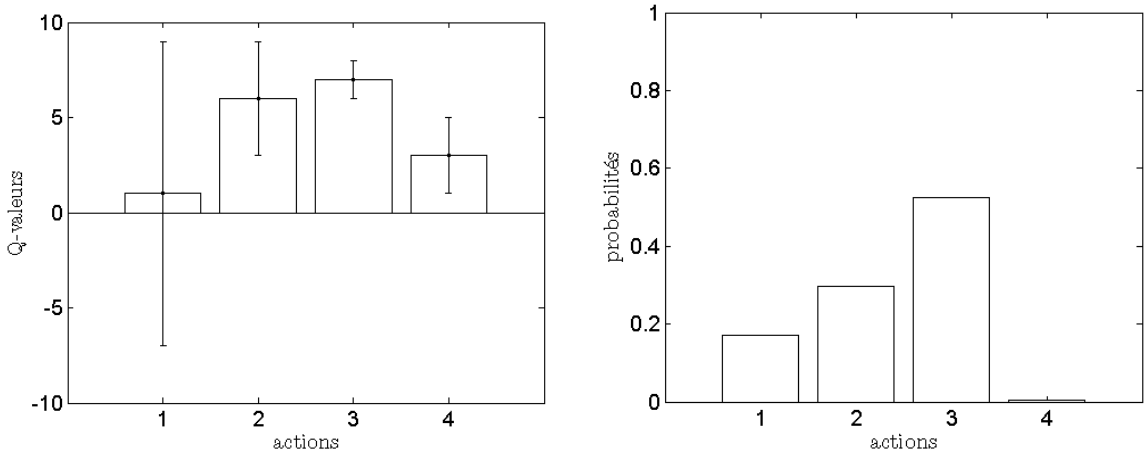
[2]Rappelons que contrairement à d'autres approches, comme GPTD par exemple, nous ne faisons pas cette hypothèse à la base pour obtenir nos algorithmes.

il est possible de tirer aléatoirement un jeu de paramètres puis d'agir de façon gloutonne par rapport à ces derniers. Ce schéma a été proposé pour la première fois par [112] pour un problème de bandit et a récemment re-émergé dans la littérature sur l'apprentissage par renforcement [23, 104].

Définition 6.6 (Politique de Thompson). *Une politique de Thompson consiste à agir de façon gloutonne par rapport à une fonction de qualité dont les paramètres ont été tirés de façon aléatoire en supposant la distribution associée gaussienne et en connaissant les moments d'ordre 1 et 2 associés :*

$$\pi(a|s) = \underset{b \in A}{\operatorname{argmax}} \hat{Q}_\xi(s, b) \text{ avec } \xi \sim \mathcal{N}(\mu_\theta, \Sigma_\theta) \tag{6.16}$$

Cette approche a le mérite d'être simple. Néanmoins, le coût de l'échantillonnage selon une gaussienne multivariée est généralement cubique (dans la mesure où cela implique une décomposition de Cholesky). Mais rappelons qu'une décomposition de Cholesky de la matrice de variance est maintenue dans le cadre de KTD, ce qui permet de conserver une complexité quadratique.



a. Q-valeurs. b. Politique de Thompson.

FIG. 6.8 – Politique de Thompson illustrée.

Nous montrons figure 6.8 la densité de probabilités qui définit la politique de Thompson pour les Q-valeurs déjà considérée. C'est en fait l'image via l'opérateur argmax des distributions sur les Q-valeurs. La dernière action, dont la valeur est estimé moyenne avec une assez bonne confiance, est presque ignorée. Les autres actions ont plus d'importance, notamment la première, dont la valeur est estimée faible mais avec peu de confiance, ce qui en fait potentiellement une bonne action.

6.3.7 Estimation myope de la valeur de l'information parfaite

La dernière approche que nous proposons est adaptée des travaux de [23, 22]. Comme toujours nous supposons disposer d'une distribution pour la Q-valeur de chaque couple état-action (ce qui est le cas, via la distribution sur les paramètres). Soit s un état donné. Pour chaque action a, la valeur de l'information parfaite quantifie le gain qu'il y a à apprendre

la valeur exacte $Q^*(s,a)$ de la fonction de qualité pour le couple état-action donné. Soient a_1 et a_2 les deux meilleures actions respectivement à l'espérance des Q-valeurs :

$$\hat{\mu}_{Q_\theta}(s,a_1) \geq \hat{\mu}_{Q_\theta}(s,a_2) \geq \hat{\mu}_{Q_\theta}(s,a),\ \forall a \neq a_1, a_2 \tag{6.17}$$

Deux cas sont intéressants. Le premier est lorsque la connaissance acquise (sur la vraie Q-valeur d'un couple état-action) indique qu'une action $a \neq a_1$ qui était auparavant considérée comme sous-optimale est en fait optimale (étant donné la croyance de l'agent sur la valeur des autres actions). Apprendre que cette action est bien sous-optimale n'apporte rien, le gain est nul. Pour un couple état-action donné (s,a), apprendre que la vraie Q-valeur associée vaut x induit le gain suivant :

$$\text{Gain}_{s,a}(x) = \left(x - \hat{\mu}_{Q_\theta}(s,a_1)\right)_+, a \neq a_1 \tag{6.18}$$

Nous notons $(x)_+ = \max(0,x)$ la partie positive de x. Le second cas d'intérêt est lorsque la connaissance acquise (sur la vraie Q-valeur) indique que l'action a_1 considérée auparavant comme optimale ne l'est pas. Apprendre que cette action est bien optimale n'apporte rien, le gain est alors nul. Pour un état s donné et l'action a_1 correspondante, apprendre que la vraie Q-valeur associée vaut x induit le gain :

$$\text{Gain}_{s,a_1}(x) = \left(\hat{\mu}_{Q_\theta}(s,a_2) - x\right)_+ \tag{6.19}$$

Comme l'agent ne sait pas à l'avance quelle valeur x de la vraie Q-valeur va être révélée, il est nécessaire de calculer le gain moyen, ce qui définit la valeur de l'information parfaite (VPI pour *value of perfect information*) :

$$\text{VPI}(s,a) = \int_{\mathbb{R}} \text{Gain}_{s,a}(x) \Pr(\hat{Q}_\theta(s,a) = x) dx \tag{6.20}$$

Remarquons également que ce n'est pas la "vraie" Q-valeur qui va être révélée, seulement une estimation améliorée, d'où le caractère myope de l'estimation de la valeur de l'information. En supposant la distribution associée à la Q-fonction gaussienne, il est possible de déterminer analytiquement l'expression (6.20). Pour cela nous avons besoin d'un lemme.

Lemme 6.1. *Soit $X \sim \mathcal{N}(\mu, \sigma^2)$ une variable aléatoire gaussienne de moyenne μ, de variance σ^2 et de densité de probabilité f_X. Alors :*

$$\int_\alpha^\beta x f_X(x) dx = \mu \Pr(\alpha \leq X \leq \beta) - \frac{\sigma}{\sqrt{2\pi}} \left[e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \right]_\alpha^\beta \tag{6.21}$$

Démonstration. Pour une distribution gaussienne, rappelons que :

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{6.22}$$

Il suffit de voir que $x f_X(x)$ est proche de $\nabla_x f(x)$, ce qui mène très directement au résultat :

$$\begin{aligned}
\int_\alpha^\beta x f_X(x) dx &= \int_\alpha^\beta \left(-\frac{\sigma}{\sqrt{2\pi}} \nabla_x + \mu \right) f_X(x) dx \\
&= -\frac{\sigma}{\sqrt{2\pi}} \int_\alpha^\beta \nabla_x f_X(x) dx + \mu \int_\alpha^\beta f_X(x) dx \\
&= \mu \Pr(\alpha \leq X \leq \beta) - \frac{\sigma}{\sqrt{2\pi}} \left[e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \right]_\alpha^\beta
\end{aligned} \tag{6.23}$$

□

Il est à noter que la distribution étant gaussienne, la quantité $\Pr(\alpha \leq X \leq \beta)$ peut se déterminer numériquement en utilisant la fonction erreur erf. Ce résultat étant posé, nous pouvons déterminer analytiquement la valeur de l'information parfaite pour tout couple état-action. Nous distinguons cependant les cas $a = a_1$ et $a \neq a_1$.

Propriété 6.1. *Soit un état s donné, $\hat{Q}_\theta$ la Q-fonction aléatoire de moyenne $\hat{\mu}_{Q_\theta}$ et de variance $\hat{\sigma}^2_{Q_\theta}$. Soient a_1 et a_2 les deux actions les plus gloutonnes, respectivement à $\hat{\mu}_{Q_\theta}$ pris en l'état s. Nous avons alors pour $a = a_1$:*

$$\begin{aligned}\mathrm{VPI}(s, a_1) = &(\hat{\mu}_{Q_\theta}(s, a_2) - \hat{\mu}_{Q_\theta}(s, a_1)) \Pr(\hat{Q}_\theta(s, a_1) < \hat{\mu}_{Q_\theta}(s, a_2)) \\ &+ \frac{\hat{\sigma}_{Q_\theta}(s, a_1)}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\hat{\mu}_{Q_\theta}(s, a_2) - \hat{\mu}_{Q_\theta}(s, a_1)}{\hat{\sigma}_{Q_\theta}(s, a_1)}\right)^2\right) \end{aligned} \tag{6.24}$$

et pour $a \neq a_1$:

$$\begin{aligned}\mathrm{VPI}(s, a) = &(\hat{\mu}_{Q_\theta}(s, a) - \hat{\mu}_{Q_\theta}(s, a_1)) \Pr(\hat{Q}_\theta(s, a_1) > \hat{\mu}_{Q_\theta}(s, a_1)) \\ &+ \frac{\hat{\sigma}_{Q_\theta}(s, a)}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\hat{\mu}_{Q_\theta}(s, a_1) - \hat{\mu}_{Q_\theta}(s, a)}{\hat{\sigma}_{Q_\theta}(s, a)}\right)^2\right) \end{aligned} \tag{6.25}$$

Démonstration. Nous montrons le résultat pour $a = a_1$, la preuve dans l'autre cas étant très similaire. Développons l'expression du gain.

$$\begin{aligned}\mathrm{VPI}(s, a_1) &= \int_{\mathbb{R}} \mathrm{Gain}_{s,a_1}(x) \Pr(\hat{Q}_\theta(s, a_1) = x) dx \\ &= \int_{-\infty}^{\hat{\mu}_{Q_\theta}(s,a_2)} (\hat{\mu}_{Q_\theta}(s, a_2) - x) \Pr(\hat{Q}_\theta(s, a_1) = x) dx \\ &= \hat{\mu}_{Q_\theta}(s, a_2) \Pr(\hat{Q}_\theta(s, a_1) \leq \hat{\mu}_{Q_\theta}(s, a_2)) \\ &\quad - \int_{-\infty}^{\hat{\mu}_{Q_\theta}(s,a_2)} x \Pr(\hat{Q}_\theta(s, a_1) = x) dx \end{aligned} \tag{6.26}$$

En appliquant le lemme précédent avec $\alpha = -\infty$ et $\beta = \hat{\mu}_{Q_\theta}(s, a_2)$, nous obtenons le résultat. □

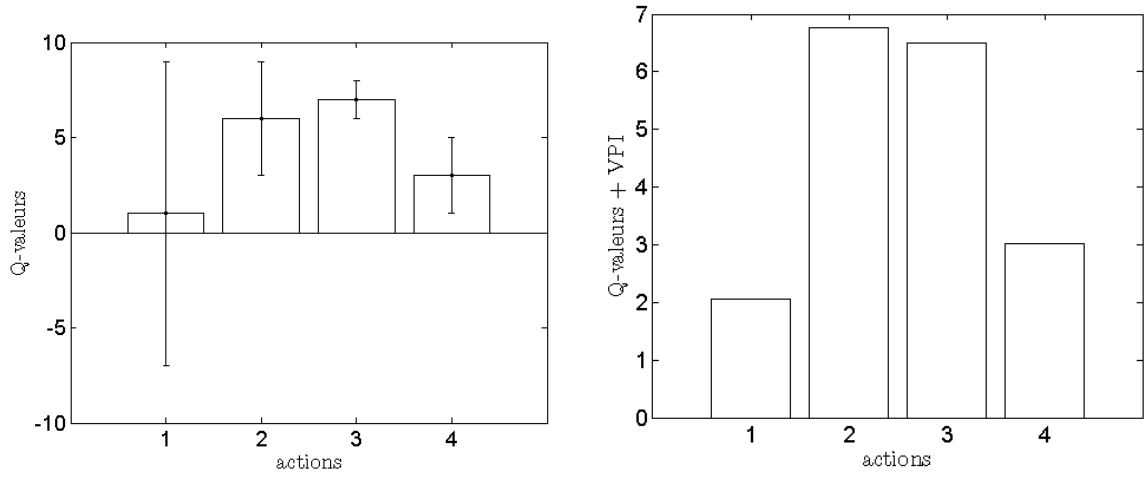
Le gain espéré pour choisir l'action a dans l'état s est donc $\mathrm{VPI}(s, a)$. Choisir une action exploratrice entraîne également un coût, qui est la différence entre la valeur de cette action et la valeur de la meilleure action (étant donnée la connaissance imparfaite du monde), c'est-à-dire $\max_{b \in A} \hat{\mu}_{Q_\theta}(s, b) - \hat{\mu}_{Q_\theta}(s, a)$. On cherche donc à maximiser la différence entre le gain et le coût, soit l'action :

$$\begin{aligned}&\operatorname*{argmax}_a \left(\mathrm{VPI}(s, a) - \left(\max_{b \in A} \hat{\mu}_{Q_\theta}(s, b) - \hat{\mu}_{Q_\theta}(s, a)\right)\right) \\ &\qquad = \operatorname*{argmax}_a \left(\hat{\mu}_{Q_\theta}(s, a) + \mathrm{VPI}(s, a)\right) \end{aligned} \tag{6.27}$$

Définition 6.7 (Politique VPI). *La politique VPI maximise l'estimation moyenne courante de la Q-fonction plus la gain espéré en choisissant l'action courante dans l'état courant :*

$$\pi(a|s) = \operatorname*{argmax}_{b \in A} \left(\hat{\mu}_{Q_\theta}(s, b) + \mathrm{VPI}(s, a)\right) \tag{6.28}$$

L'expression analytique de VPI *est donnée dans la propriété 6.1.*



a. Q-valeurs. b. Politique VPI.

FIG. 6.9 – Politique VPI illustrée.

Nous illustrons une politique VPI sur la figure 6.9. Ce type de politique est déterministe, car gloutonne par rapport aux valeurs estimées auxquelles on ajoute le VPI, ce qui est représenté ici pour les Q-valeurs que nous considérons depuis le début. Dans cet exemple, le gain espéré en choisissant la deuxième action fait privilégier cette dernière à l'action gloutonne (la troisième). Les valeurs des première et quatrième actions (en tenant compte du gain espéré) sont sensiblement plus faibles.

6.4 Expérimentations

Dans cette section nous proposons un certain nombre d'expériences visant à illustrer l'information d'incertitude disponible dans le cadre des différences temporelles de Kalman ainsi que son utilisation. En voici un aperçu :

labyrinthe : ce simple problème de labyrinthe dans lequel l'agent doit trouver la sortie d'une pièce vise à illustrer l'information d'incertitude disponible avec le cadre de travail proposé par KTD ;

pendule inversé : ce problème vise à illustrer l'apprentissage actif proposé en section 6.2. L'algorithme KTD-Q actif est considéré et nous montrons qu'utiliser la politique comportementale informée plutôt qu'une politique totalement aléatoire accélère l'apprentissage ;

bandit : un bandit est un MDP avec un seul état et un certain nombre d'actions. Le problème est simple mais tout à fait adapté à l'illustration du comportement des différentes heuristiques proposées pour traiter le dilemme entre exploration et exploitation.

Il y aura peu de comparaisons à l'état de l'art dans cette section (uniquement une comparaison à Q-learning pour le pendule inversé) car cela se justifie moins dans le contexte présent (le premier problème est purement illustratif, pour le deuxième la politique comportementale uniforme sert de référence et pour le troisième la politique ϵ-gloutonne est le référent).

6.4.1 Simple labyrinthe

Nous illustrons ici l'information d'incertitude obtenue dans le cadre des différences temporelles de Kalman sur un simple problème de labyrinthe. L'espace d'état continu et bi-dimensionnel est le carré unitaire : $(x, y) \in [0, 1]^2$. Les actions sont des déplacements d'amplitude 0.05 vers le haut, le bas, la gauche et la droite. La récompense est $+1$ si l'agent quitte le labyrinthe en $y = 1$ et $x \in [-\frac{3}{8}, \frac{3}{8}]$, -1 si l'agent quitte le labyrinthe en $y = 1$ et $x \in [-1, \frac{3}{8}[\cup]\frac{3}{8}, 1]$ et 0 partout ailleurs. L'algorithme considéré est KTD-V. La paramétrisation est un ensemble de 9 noyaux gaussiens équi-répartis (centrés en $\{0, 0.5, 1\} \times \{0, 0.5, 1\}$) d'écart-type égal à 0.5. Le facteur d'actualisation γ est fixé à 0.9. L'agent commence dans une position aléatoire (x_0, y_0) avec x_0 échantillonné selon une distribution gaussienne, $x \sim \mathcal{N}(\frac{1}{2}, \frac{1}{8})$ et y_0 échantillonné selon une distribution uniforme, $y_0 \sim \mathcal{U}_{[0,0.05]}$. La politique apprise est d'aller en haut avec une probabilité 0.9 et d'aller dans une des trois autres directions avec une probabilité $\frac{0.1}{3}$. Le vecteur de paramètres initial est nul et l'*a priori* sur leur variance est $P_{0|0} = 10I$. Les variances des bruits sont fixées à $P_{n_i} = 1$ et $P_{v_i} = 0I$.

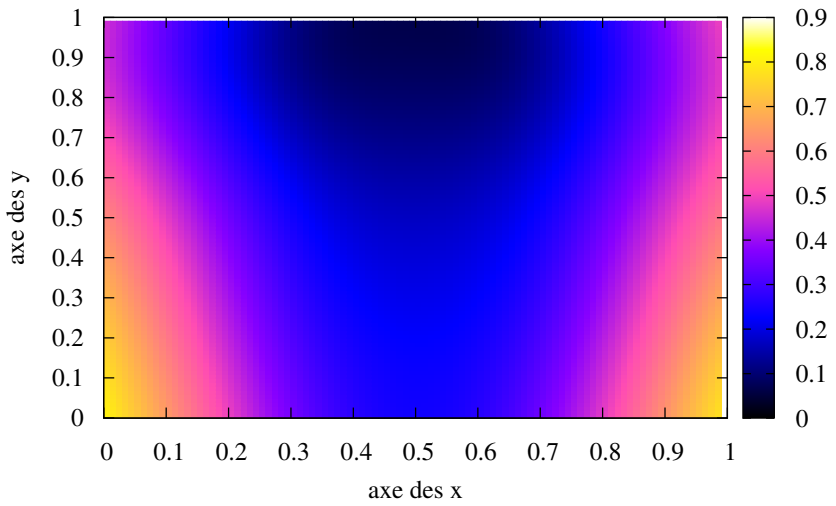


FIG. 6.10 – Simple labyrinthe, illustration de l'incertitude.

La fonction de valeur est bien apprise. Cependant, ce n'est pas le point que nous souhaitons soulever ici. L'objectif est d'illustrer l'incertitude de la fonction de valeur. L'apprentissage est fait sur 30 épisodes, les résultats sont présentés sur la figure 6.10 qui montre l'écart-type de la fonction de valeur approchée sur l'ensemble de l'espace d'état. Si l'on considère l'axe des x, l'incertitude est plus faible au milieu de l'environnement qu'aux bords, ce qui est expliqué par le fait que lors de l'apprentissage les trajectoires sont plus fréquentes vers le centre du domaine (distribution gaussienne pour la position initiale x_0). Si l'on considère l'axe des y, l'incertitude est plus faible au niveau de la borne supérieure ($y = 1$) que près de la borne inférieure ($y = 0$), ce qui est expliqué par le fait que les valeurs rétropropagées sont moins certaines. Ainsi l'information d'incertitude inhérente à KTD a du sens sur ce problème simple et elle devrait s'avérer être un outil utile pour traiter

le dilemme entre exploration et exploitation, d'une part pour avoir un apprentissage plus rapide, mais aussi un apprentissage plus sûr (dans le sens où une action dont la valeur estimée est grande, mais associée à une forte incertitude, ne sera peut-être pas choisie, ce qui peut s'avérer important dans un contexte industriel).

6.4.2 Pendule inversé et apprentissage actif

Cette expérimentation consiste à apprendre le contrôle optimal d'un pendule inversé, tel que décrit par [73] ainsi que dans la section 3.8.2. La même paramétrisation est choisie et nous rappelons les résultats de l'apprentissage de la politique optimale à partir d'une politique totalement exploratrice illustrés figure 3.9.

Nous expérimentons ici la forme d'apprentissage actif (exploration pure) présentée section 6.2. La politique suivie n'est plus uniformément aléatoire, mais pondérée par l'incertitude associée aux actions. La longueur moyenne d'un tel épisode est de 11 pas, ce qui change peu par rapport à une politique totalement aléatoire (10 pas). La durée d'un épisode ne peut donc être que faiblement responsable de l'amélioration des résultats. Pour chaque essai, l'apprentissage est fait sur 300 épisodes. Moins d'épisodes sont considérés pour montrer l'accélération de la convergence, asymptotiquement les deux variations de KTD se comportent aussi bien. Tous les 25 épisodes l'apprentissage est gelé et les politiques évaluées comme précédemment, avec la même mesure de performance. Les résultats de la figure 6.11 sont moyennés sur 100 essais. Notons que l'échelle n'est plus logarithmique.

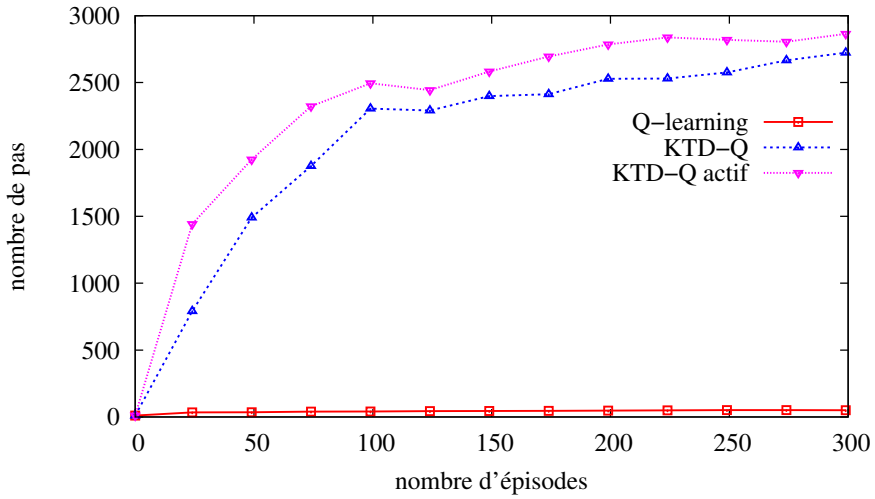


FIG. 6.11 – Pendule inversé, apprentissage actif.

Cette expérience compare KTD-Q actif à KTD-Q et Q-learning. En comparant les deux variantes de KTD, il est clair que choisir les actions en fonction de l'incertitude accélère la convergence. Cette dernière est quasiment doublée sur les 100 premiers épisodes : par exemple, une performance moyenne de 1500 pas est obtenue après seulement 25 épisodes avec KTD-Q actif, alors qu'elle n'est atteinte qu'après environ 50 épisodes par KTD-Q. De plus, le fait d'être en échelle non-logarithmique met encore plus en avant les mauvaises performances de l'algorithme Q-learning avec approximation, la paramétrisation de la fonction de valeur étant la même.

6.4.3 Bandits

Dans cette section nous proposons d'étudier et de comparer les différents schémas d'exploration/exploitation proposés section 6.3 sur un problème de bandits et d'analyser dans une certaine mesure l'influence des différents paramètres. Un problème de bandits est particulièrement simple, il peut être modéliser par un MDP à 1 état et dont le facteur d'actualisation est nul. Cependant, c'est un problème souvent choisi pour tester cet aspect, car les résultats sont facilement interprétables. Surtout, cette étude a un objectif prospectif : l'idée est de montrer que l'information d'incertitude fournie par KTD peut être utile et non pas de tenter de résoudre le dilemme entre exploration et exploitation.

Description du benchmark

Le problème considéré ici est un bandit à N bras. Le MDP correspondant a un état, N actions et le facteur d'actualisation est logiquement nul (il n'y a pas de transitions). Chaque action a produit une récompense de 1 avec une probabilité p_a et une récompense de 0 avec une probabilité de $1 - p_a$ (distribution de Bernouilli). Pour une action a^*, choisie aléatoirement parmi les actions possibles, cette probabilité est fixé à $p_{a^*} = 0.6$. Pour toutes les autres actions, les probabilités correspondantes sont tirées aléatoirement entre 0 et 0.5 : $p_a \sim \mathcal{U}_{[0,0.5]}, \forall a \neq a^*$. Les résultats que nous présentons sont moyennés sur 1000 expériences et pour chaque expérience le benchmark est généré aléatoirement. Chaque expérience est conduite sur 6000 interactions entre l'agent et le bandit. Les figures montrent le pourcentage de choix de l'action correspondant au bras optimal en fonction du nombre d'interactions. Comme bruit d'évolution pour KTD, nous choisissons un bruit adaptatif du type $P_{v_i} = \eta P_{i-1|i-1}$.

Premières expérimentations

Pour cette première série d'expérimentations, nous comparons les différentes approches proposées pour un jeu de paramètres donné. L'algorithme considéré est KTD-SARSA, combiné à l'une des politiques proposées. Nous fixons le nombre de bras du bandit à $N = 10$. Le vecteur de paramètres initial est nul, $\hat{\theta}_{0|0} = 0$ et la variance associée est choisie égale à $P_{i|i} = 10^{-1} I$ (représentation tabulaire de la Q-fonction). Le bruit d'observation est fixé à $P_{n_i} = 1$. Le bruit de process est nul, $\eta = 0$. Pour les politiques ϵ-gloutonnes (informée ou non), le facteur ϵ est fixé à 0.1. Pour la politique Bayes-gloutonne, le paramètre libre ν est choisi égal à 1. Pour la politique confiante gloutonne, la paramètre α est choisi égal à 3, ce qui correspond à un intervalle de confiance à 95% (toujours sous hypothèse d'une distribution gaussienne). Pour la méthode basée sur l'addition d'un bonus d'exploration, nous choisissons $\beta_0 = 1$, relativement arbitrairement, et $\beta = 10$ de façon à avoir un bonus initial de l'ordre du cumul de récompenses maximal (l'écart-type initial de la Q-fonction pour un couple état-action donné se déduit simplement de l'*a priori* sur la variance des paramètres). Pour résumer :

$$\begin{cases} N = 10 \\ \eta = 0 \\ P_{0|0} = 10^{-1} I \\ \theta_{0|0} = \mathbf{0} \\ P_{n_i} = 1 \end{cases} \quad \text{et} \quad \begin{cases} \epsilon = 0.1 \\ \nu = 1 \\ \alpha = 3 \\ \beta_0 = 1 \\ \beta = 10 \end{cases} \tag{6.29}$$

Le problème considéré ayant 10 bras, une politique totalement aléatoire choisirait la bonne action une fois sur 10 en moyenne, ce qui correspond à une performance de 0.1 selon notre critère de mesure. Notons également qu'une politique purement gloutonne choisirait systématiquement l'action pour laquelle la première récompense positive a été observée, étant donnée l'initialisation des paramètres. Les résultats sont présentés figure 6.12.

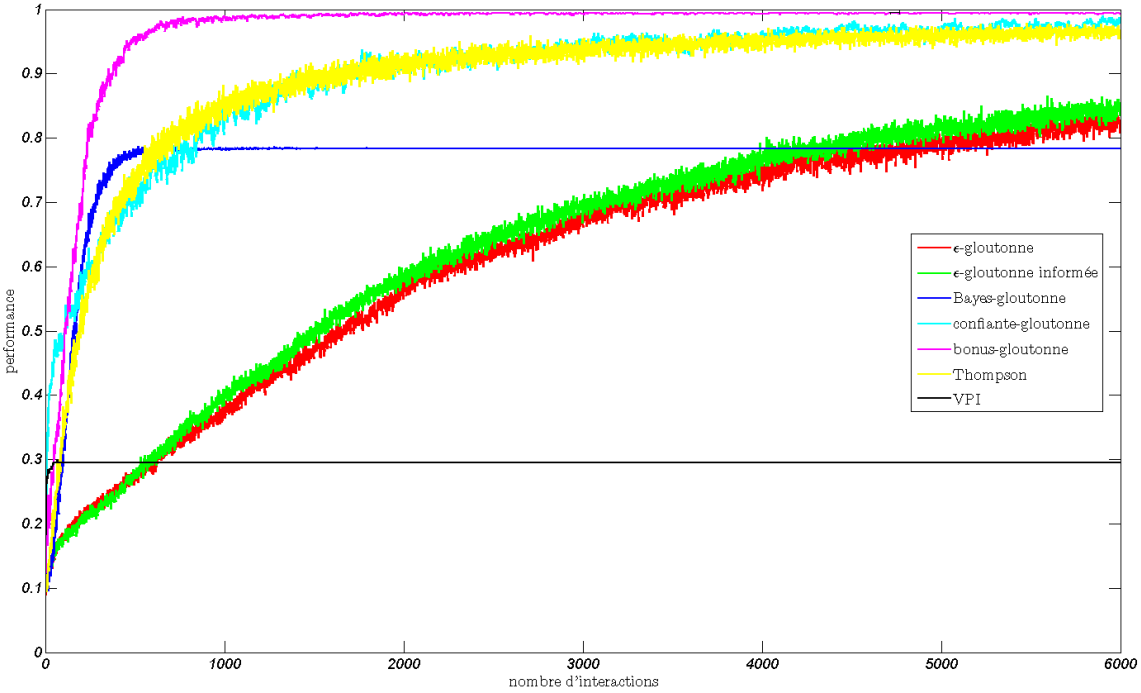


FIG. 6.12 – Bandits, comparaison des différentes approches.

La politique ϵ-gloutonne tend asymptotiquement vers une valeur proche de 1 (il n'y a cependant assez d'itérations sur la figure pour voir cette asymptote), la distance à 1 dépendant de la valeur de ϵ. La convergence est relativement lente. La politique ϵ-greedy informée est légèrement meilleure. Tirer les actions selon l'incertitude associée améliore légèrement les résultats. Cependant, la vitesse de convergence est sensiblement la même. La politique Bayes-gloutonne converge rapidement dans un premier temps, puis elle reste coincée sur un palier, avec d'ailleurs très peu de variance. Nous pensons que cela peut être expliqué de la façon suivante : la variance tend trop rapidement vers 0, KTD devient donc trop confiant en ses propres estimations (effet amplifié par l'exponentielle). En conséquence, la politique aura tendance à choisir les actions dont elle est le plus sûr, plutôt que les actions les plus prometteuses. Il est possible de compenser ce problème en jouant sur le paramètre libre ν, comme nous le verrons par la suite. La politique confiante-gloutonne, basée sur l'estimation de l'intervalle de confiance, se comporte plutôt bien. La convergence est assez rapide (plus que du ϵ-glouton, informé ou non) et le comportement asymptotique semble correct. La politique bonus-gloutonne fournit ici les meilleurs résultats. La convergence est très rapide et le comportement asymptotique est excellent (très proche de 1, avec très peu de variance). Là aussi le choix des paramètres libres joue et ils semblent avoir été mieux choisis que ν. La politique de Thompson présente des résultats très proches de la politique basée sur l'intervalle de confiance. Enfin, la politique basée sur la valeur de l'information converge très rapidement dans un premier temps, mais elle atteint très vite un palier relativement

faible qu'elle ne quitte plus. Nous pensons qu'ici le problème est similaire à celui qui se pose pour la politique Bayes-gloutonne, KTD devient trop rapidement trop confiant en ses estimations. Seulement, il n'y a pas ici de paramètre libre sur lequel jouer pour compenser ce problème.

Influence des paramètres libres

Nous étudions ici succinctement l'influence des paramètres libres pour les politiques Bayes-gloutonne et bonus-gloutonne. Nous nous intéressons d'abord au premier type de politique. La variance initiale de la valeur de chaque action est liée à l'*a priori* $P_{0|0}$ et vaut dans notre cas $\sigma_0^2 = \frac{1}{N}\operatorname{trace}(P_{0|0})$. Or on peut souhaiter que l'écart-type initial considéré dans la politique Bayes-gloutonne soit plutôt proportionnelle au cumul maximal de récompenses qu'il est possible d'espérer. Nous proposons donc de choisir le paramètre ν de la façon suivante :

$$\nu = \frac{\sigma_0(1-\gamma)}{r_{\max}} \text{ avec } \sigma_0 = \sqrt{\frac{1}{N}\operatorname{trace}(P_{0|0})} \tag{6.30}$$

Nous allons donc nous intéresser aux résultats de la politique Bayes-gloutonne, dans le cas où $\nu = 1$ comme précédemment et en utilisant le schéma proposé. Mais avant cela intéressons nous au choix des paramètres libres pour la politique bonus-gloutonne.

Pour l'approche utilisant un bonus d'exploration, il y a deux paramètres libres à choisir, β_0 et β. Nous avons proposé de prendre le premier égal à 1 et de choisir le second de telle façon que le bonus initial soit $\frac{r_{\max}}{1-\gamma}$, c'est-à-dire :

$$\beta \approx \frac{r_{\max}}{1-\gamma}\frac{\beta_0 + \sigma_0^2}{\sigma_0^2} \text{ avec } \sigma_0^2 = \frac{1}{N}\operatorname{trace}(P_{0|0}) \tag{6.31}$$

Nous allons étudier les courbes correspondantes à $\beta_0 = 0.1$ et $\beta_0 = 10$. Nous présentons les résultats figure 6.13, avec une politique ϵ-gloutonne comme référence.

L'heuristique introduite sur ν améliore globalement les résultats. La convergence est moins rapide au départ, mais le comportement asymptotique est bien meilleur, il est équivalent au meilleur que nous ayons obtenu jusqu'ici. Pour la politique basée sur un bonus d'exploration, les performances sont plutôt dégradées par rapport à celles que nous avons déjà obtenues. Si le paramètre β_0 est choisi trop petit, la politique devient rapidement trop confiante et la performance stagne sur un palier assez bas. Si β_0 est trop grand, la politique ne stagne pas mais la convergence est beaucoup plus lente (la réduction de l'exploration est moins rapide). Il serait également possible de jouer sur le rapport entre le bonus initial et le cumul maximal de récompenses (ce qui reviendrait, pour un β_0 donné, à multiplier le β correspondant par une constante).

Ainsi, avec ce type d'approche, on peut obtenir de très bon résultats (politique bonus-gloutonne bien réglée par exemple) comme de plus mauvais (politique bonus-gloutonne mal réglée par exemple). Cela dépend du choix des paramètres, qui dépend du problème d'intérêt et des paramètres choisis pour KTD. Ce sont d'ailleurs ces paramètres que nous étudions à présent.

Influence du bruit d'observation

Dans la section précédente nous avons choisi un bruit d'observation relativement important. Cependant, comme la représentation de la Q-fonction est tabulaire, c'est-à-dire

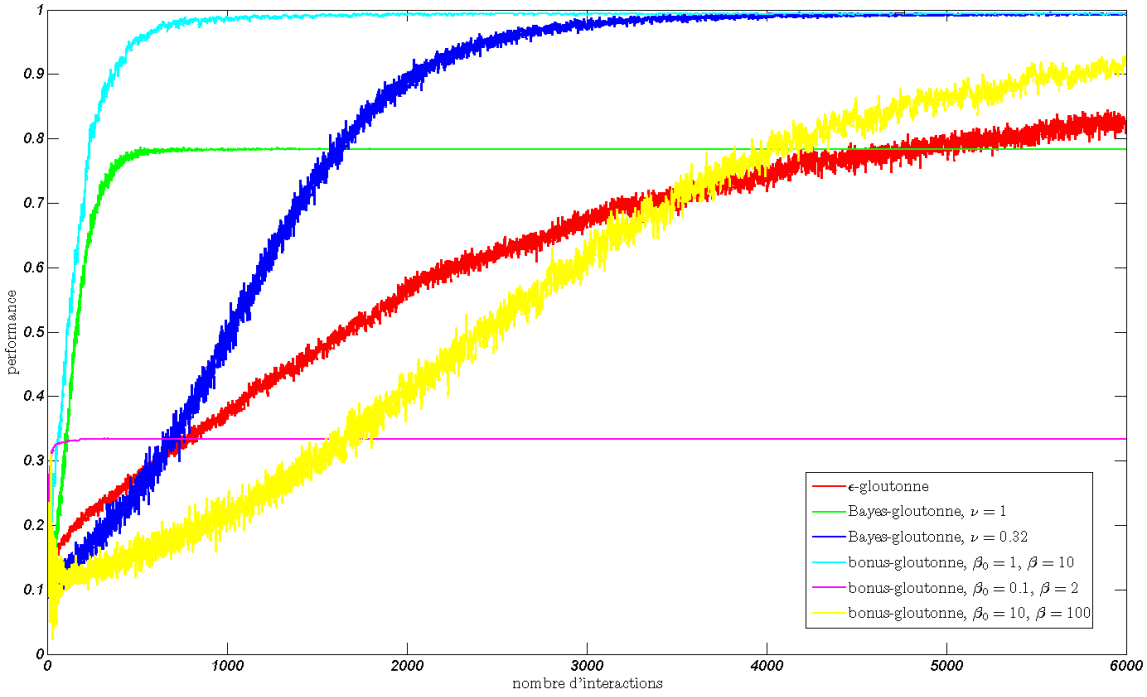


FIG. 6.13 – Bandits, influence des paramètres libres.

comme elle est exacte, nous pouvons choisir un bruit arbitrairement petit (pratiquement il faut éviter les valeurs trop faibles pour des questions de stabilité numérique). De plus, le choix du bruit d'observation a une influence sur la vitesse de convergence de l'algorithme (ou de traque lorsque le cas non-stationnaire est envisagé). Lorsque une représentation exacte est possible, plus le bruit d'observation est petit, plus la convergence est rapide. Cependant, la mise à jour de la variance est proportionnelle à l'inverse de la variance du bruit d'observation, en baissant sa valeur il y a donc un risque que KTD devienne trop confiant, comme nous l'avons déjà observé précédemment. Nous étudions cet aspect ici. Le bruit d'observation est choisi égal à $P_{n_i} = 10^{-3}$, le facteur ν est choisi tel que $\nu = \sqrt{10^{-1}}$, β_0 et β valent respectivement 1 et 10, tous les autres paramètres ne sont pas modifiés. Les résultats sont présentés figure 6.14.

L'influence de la valeur du bruit sur la vitesse de convergence peut s'observer sur les politiques ϵ-gloutonnes. Leur convergence est plus rapide maintenant que le bruit d'observation est plus faible. Informer la politique ϵ-gloutonne améliore toujours un peu les performances dans la première phase de convergence, mais il n'y a plus de différences significative sur la phase asymptotique (ce qui est somme toute logique). Pour toutes les autres politiques proposées, la convergence est rapide dans un premier temps, mais les politiques se retrouvent très rapidement coincées sur un palier, plus ou moins haut selon l'approche. Rappelons que baisser la valeur du bruit d'observation accélère la convergence. Dans ce cas, la variance sur les paramètres tend également plus rapidement vers 0, ce qui a pour effet de rendre KTD trop rapidement trop confiant en ses estimations, ce qui explique ces phénomènes de palier.

Ici encore il est possible d'envisager de jouer sur les paramètres libres pour améliorer le comportement des politiques qui peuvent être réglées de cette façon et nous allons l'illustrer après avoir introduit une nouvelle heuristique. Remarquons d'abord que la mise à jour des

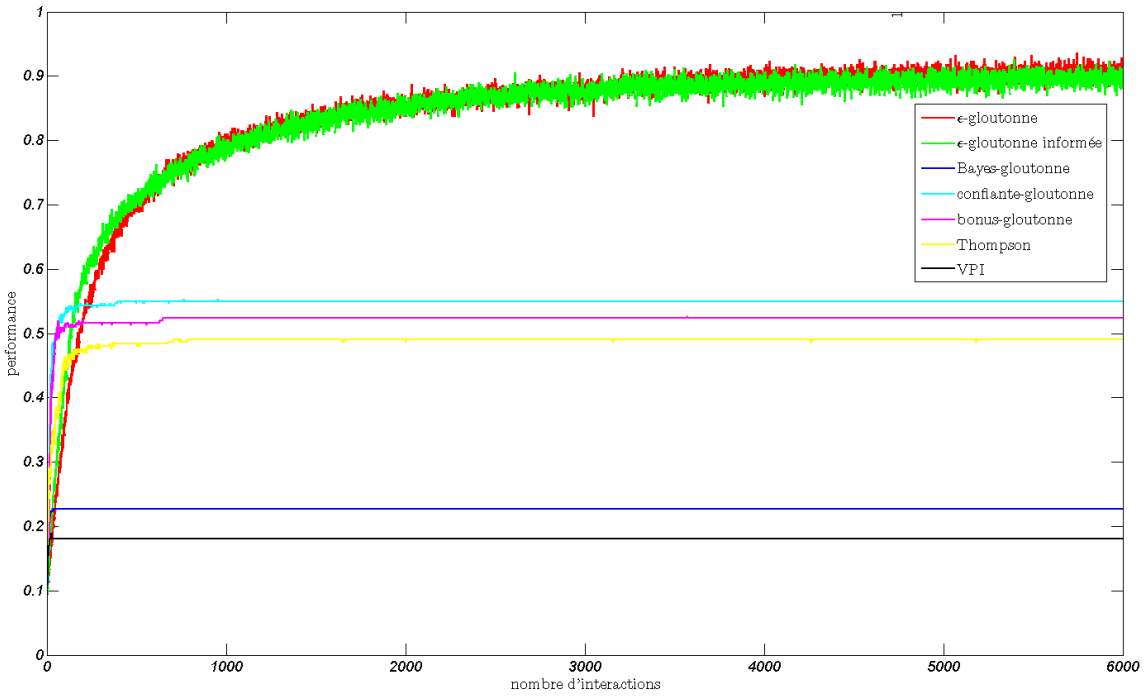


FIG. 6.14 – Bandits, influence du bruit d'observation.

paramètres se fait en fonction du gain de Kalman et de l'erreur de prédiction, alors que la mise à jour de la variance se fait en fonction du gain de Kalman et de la variance de l'innovation. Cette dernière variance se réduit à la variance du bruit d'observation lorsque la variance sur les paramètres est suffisamment faible. Nous proposons donc de diviser le facteur β par la variance du bruit d'observation et de multiplier le facteur ν par l'écart-type de ce même bruit, de façon à ralentir (artificiellement) la convergence de la variance :

$$\nu = \frac{\sigma_0(1-\gamma)}{r_{\max}}\sqrt{P_{n_i}} \text{ avec } \sigma_0 = \sqrt{\frac{1}{N}\operatorname{trace}(P_{0|0})} \tag{6.32}$$

$$\beta \approx \frac{1}{P_{n_i}}\frac{r_{\max}}{1-\gamma}\frac{\beta_0+\sigma_0^2}{\sigma_0^2} \text{ avec } \sigma_0^2 = \frac{1}{N}\operatorname{trace}(P_{0|0}) \tag{6.33}$$

Les résultats correspondants sont présentés figure 6.15.

Il apparaît que les heuristiques introduites sont efficaces dans ce cas. Pour la politique Bayes-gloutonne, la convergence est plus lente que pour la politique ϵ-gloutonne. Cependant, le comportement asymptotique est meilleur. Pour la politique bonus-gloutonne la convergence est légèrement plus lente que la politique ϵ-gloutonne au départ, mais elle atteint un comportement asymptotique très proche de 1 très rapidement. Ainsi un choix judicieux des paramètres libres permet d'obtenir un bon compromis entre exploration et exploitation, ainsi que de bonnes performances asymptotiques. Cependant nous n'affirmons pas que les heuristiques proposées sont toujours adaptées.

Ainsi, nous avons vu que si l'on ne prend pas de précaution, la plupart des heuristiques proposées atteignent très rapidement des paliers sous-optimaux. L'origine de ce problème est le suivant : les heuristiques proposées (sauf la politique ϵ-gloutonne informée et la bonus-gloutonne) utilisent la valeur absolue de la variance et pas une valeur relative comme le

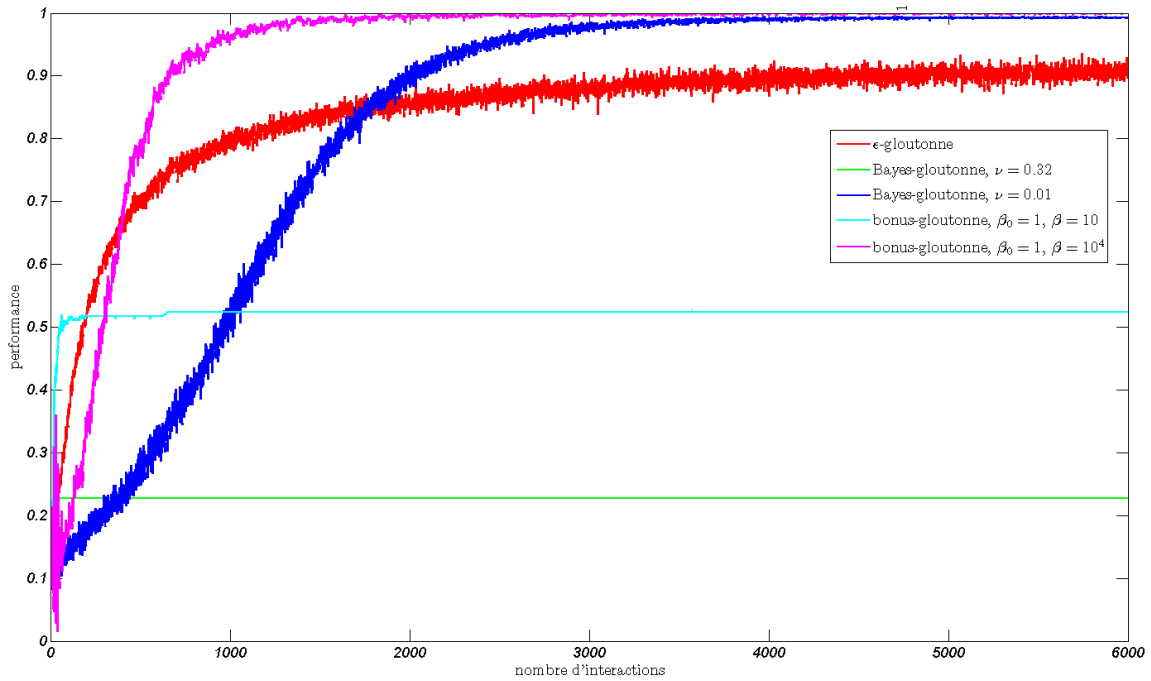


FIG. 6.15 – Bandits, influence du bruit d'observation et des paramètres libres.

schéma d'apprentissage actif par exemple. Or la valeur de cette variance et son évolution dépendent beaucoup des paramètres choisis pour KTD. Pour pouvoir utiliser ces schémas systématiquement, il faudrait pouvoir normaliser l'information d'incertitude disponible, c'est ce que nous avons plus ou moins fait en proposant des heuristiques pour les paramètres libres des politiques qui en acceptent. Une autre approche est d'ajouter du bruit d'évolution, de façon à tempérer la certitude qu'a KTD en ses estimations. On pourrait penser que l'on mesure simplement l'incertitude qu'on ajoute, mais en fait l'expérience montre que cette approche améliore l'apprentissage en évitant une convergence trop rapide de KTD combiné à l'une des politiques présentées, potentiellement vers la mauvaise solution.

Influence du bruit d'évolution

Pour étudier l'influence du bruit d'évolution, nous reprenons les valeurs précédentes pour les différents paramètres (valeurs avec lesquelles la figure 6.14 a été produite, donc sans inclure les heuristiques relatives au bruit d'observation), sauf pour le bruit d'évolution. Nous considérons un bruit adaptatif, comme annoncé, avec $\eta = 10^{-2}$. Pour des questions de stabilité numérique, pour la politique Bayes-gloutonne nous autorisons une variance minimale de 10^{-6}. Les résultats sont présentés figure 6.16.

Avec l'ajout d'un bruit de process, le comportement des politiques ϵ-gloutonnes est le même que sans, excepté des performances asymptotiques légèrement dégradées. La politique Bayes-gloutonne ne profite que très peu de l'ajout de ce bruit (le palier atteint est légèrement plus haut). Cependant, pour les quatre autres politiques, le changement est notable. La convergence est un peu plus lente, mais le comportement asymptotique est bien meilleur (avec un léger avantage de la politique VPI). Ainsi l'ajout du bruit de process réduit la certitude de KTD en ses estimations, ce qui s'avère bénéfique pour les schémas

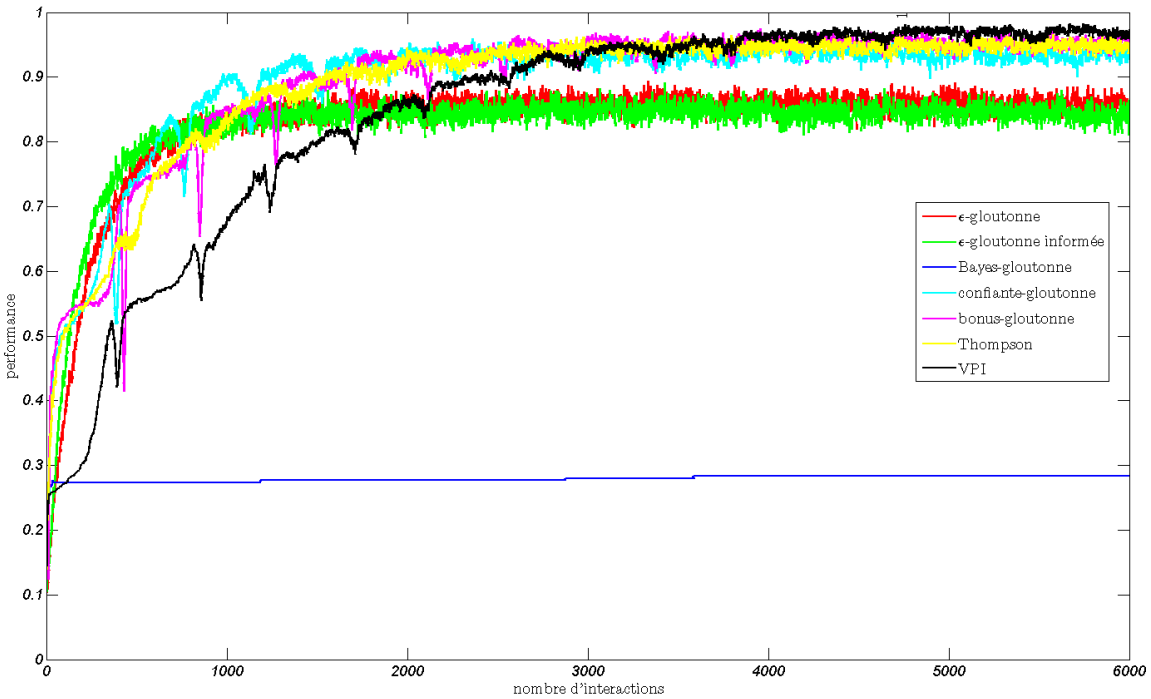


FIG. 6.16 – Bandits, influence du bruit d'évolution.

d'exploration/exploitation que nous avons proposés.

6.5 Bilan

Dans ce chapitre, nous avons montré comment calculer simplement l'incertitude liée aux estimations faites de la fonction de valeur ou de qualité, en utilisant la matrice de variance des paramètres et ceci que la paramétrisation soit linéaire ou non. Nous l'avons tout d'abord utilisée dans une forme d'apprentissage actif, que nous avons montré sur un problème de pendule inversé accélérer l'apprentissage dans le contexte de KTD-Q. Nous avons également adapté un certain nombre de méthodes de l'état de l'art traitant le dilemme entre exploration et exploitation au cadre de travail de KTD, l'idée étant de voir que l'information d'incertitude est utile. Nous avons montré sur un simple problème de bandits qu'effectivement elle permettait d'obtenir de bons résultats, cependant assez fortement dépendants du réglage de l'algorithme (ce que nous avons essayé de tempérer en introduisant quelques heuristiques, visant principalement à "normaliser" cette information d'incertitude). Toutefois, l'idée de ce chapitre n'était pas de résoudre le problème du dilemme entre exploration et exploitation, mais plutôt de montrer que l'information d'incertitude qui peut être obtenue a du sens et peut être utile. De ce point de vue le contrat est rempli, ce que nous pensons ouvrir d'intéressantes perspectives.

Chapitre 7

Un algorithme acteur-critique

Les méthodes que nous avons proposées jusqu'à présent peuvent être qualifiées de "critique pur". Une estimation de la fonction de valeur (ou de qualité) est faite et la politique en est déduite directement (politique gloutonne, ϵ-gloutonne, de Gibbs, *et caetera*). Il n'y a pas de représentation à part entière de la politique. Un inconvénient de ce type d'approche est que l'on ne peut pas forcément garantir la qualité de la politique déduite de la Q-fonction approchée, même dans les cas où l'on peut donner certaines bornes sur l'erreur faite sur la Q-fonction. Un autre problème concerne les grands espaces d'action. Que le schéma considéré soit l'itération de la valeur ou de la politique, il y a forcément un moment où il faut calculer un max (ou éventuellement un argmax), ce qui devient un problème d'optimisation à part entière lorsqu'il y a trop d'actions. Pour l'itération de la valeur, c'est la résolution de l'équation de Bellman qui pose problème, pour l'itération de la politique c'est la phase d'amélioration qui peut s'avérer complexe.

Au contraire, d'autres approches adoptent une représentation de la politique et cherchent directement la politique optimale, sans passer par l'estimation d'une fonction de valeur. Ces méthodes sont généralement qualifiées de "recherche directe dans l'espace des politiques" ou encore "acteurs purs". Un classique de ce type de méthodes est le gradient sur la politique [98, tome 2, chapitre 3]. Une représentation paramétrique de la politique est choisie et les paramètres sont corrigés en suivant une montée de gradient (du cumul espéré des récompenses selon les paramètres de la politique), ce gradient étant généralement estimé par simulation de Monte Carlo. L'avantage de ce type d'approches est que l'on cherche directement dans l'espace des politiques et que les espaces d'action continus ne posent pas particulièrement problème. Un autre avantage est que certaines de ces approches ne nécessitent pas d'hypothèse markovienne, ce que nous ne développerons pas plus. Cependant, l'estimation de la correction à apporter aux paramètres peut être coûteuse en termes d'échantillons voire en temps de calcul.

L'idée des architectures acteur-critique, que nous avons déjà brièvement présentées au chapitre 2, est de mêler ces deux paradigmes. Deux représentations distinctes sont maintenues, une pour la politique (l'acteur) et l'autre pour la fonction de valeur (le critique). Le critique estime la performance de l'acteur, qui utilise cette estimation pour corriger sa représentation. Potentiellement, ce type d'approche pourrait permettre de combiner les avantages des deux paradigmes, notamment l'efficacité en terme d'échantillons que peuvent présenter les méthodes critique pur et la gestion naturelle des grands espaces d'action que présentent généralement les méthodes acteur pur. De plus, ce type de méthode gère habituellement implicitement le dilemme entre exploration et exploitation. Dans ce chapitre,

nous introduisons une méthode acteur-critique dont le critique est KTD(λ) et dont l'acteur est corrigé en utilisant un gradient naturel [1]. Nous appelons cette méthode KNAC (*Kalman-based Natural Actor-Critic*).

7.1 Un algorithme introductif

Nous présentons d'abord l'un des premiers algorithmes acteur-critique à avoir été proposé (présenté notamment dans [105]), traitant le cas tabulaire. Il est relativement simple et l'objectif ici est à travers cet exemple d'illustrer les idées sous-jacentes à ce type d'approches.

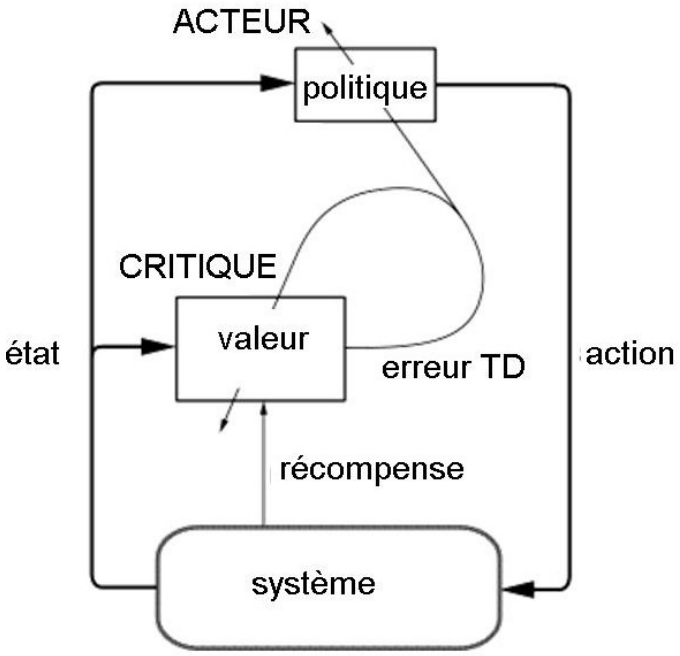


FIG. 7.1 – Architecture acteur-critique générale (bis).

Nous rappelons figure 7.1 l'illustration d'un schéma acteur-critique, déjà donnée figure 2.4. L'agent est composé de deux entités, l'acteur et le critique. L'acteur, qui représente la politique, choisit les actions à appliquer au système. Le critique, qui représente la fonction de valeur, c'est-à-dire la performance de l'acteur, critique les actions prises par ce dernier. Chaque entité maintient sa propre représentation. Les espaces d'état et d'action sont supposés de cardinaux suffisamment faibles pour qu'une représentation tabulaire soit possible. Le critique, dont le but est d'évaluer les choix de l'acteur, est représenté par $|S|$ paramètres, chacun correspondant à l'estimation de la valeur d'un état donné. L'acteur, quant à lui, doit coder la politique, qui est ici considérée stochastique. Il maintient ainsi $|S| \times |A|$ paramètres $\hat{\pi}(s,a)$ qui sont utilisés sous la forme d'une distribution de Gibbs :

$$\hat{\pi}(a|s) = \frac{\exp\left(\hat{\pi}(s,a)\right)}{\sum_{b \in A} \exp\left(\hat{\pi}(s,b)\right)} \tag{7.1}$$

Il est clair que plus le paramètre $\hat{\pi}(s,a)$ sera grand, plus l'action a aura de chance d'être choisie.

Au pas de temps i, le système se trouve dans un état s_i. L'acteur choisit l'action a_i à appliquer selon la politique qu'il maintient, c'est-à-dire selon $\hat{\pi}(.|s_i)$. Le système évolue en suivant sa propre dynamique vers un nouvel état s_{i+1} et le critique reçoit l'information de récompense r_i, qui est l'information locale de la qualité du contrôle. A partir de cette récompense et de son estimation actuelle de la fonction de valeur, le critique calcule l'erreur de différence temporelle :

$$\delta_i = r_i + \gamma \hat{V}(s_{i+1}) - \hat{V}(s_i) \tag{7.2}$$

Cette erreur a deux objectifs, mettre à jour le critique, mais aussi l'acteur. Pour le critique, elle sert à corriger la représentation de la fonction de valeur, c'est l'algorithme TD classique :

$$\hat{V}(s_i) \leftarrow \hat{V}(s_i) + \alpha_i \delta_i \tag{7.3}$$

L'erreur de différence temporelle est également utilisée pour mettre à jour l'acteur. En effet, si cette dernière est positive, cela signifie que la récompense prédite (en utilisant l'estimation de la fonction de valeur, cette prédiction prend donc en compte le long terme) est moindre que la récompense observée et l'action a_i devrait être renforcée. Si au contraire l'erreur de différence temporelle est négative, l'action a_i n'a pas un aussi bon effet qu'escompté et la probabilité de la choisir devrait baisser. La mise à jour suivante de l'acteur est proposée dans [105] :

$$\hat{\pi}(s_i, a_i) \leftarrow \hat{\pi}(s_i, a_i) + \beta_i \delta_i \tag{7.4}$$

Dans cette expression, β_i est un taux d'apprentissage, potentiellement différent de α_i. Il est clair qu'avec ce type d'architecture, il n'y a pas besoin de faire de recherche de maximum, dans la mesure où les paramètres de la politique sont directement modifiés. L'amélioration de la politique ne passe pas par une une phase gloutonne, mais par la mise à jour directe de sa représentation.

Il est assez aisé de comprendre ce qu'il se passe. Pour une politique π donnée, si la fonction de valeur est bien estimée et pour une action choisie dans un état donné, l'erreur de différence temporelle représente l'*avantage* qu'il y a à choisir cette action par rapport à celle préconisée par la politique. En effet, si l'action est choisie selon la politique, alors en moyenne l'erreur de différence temporelle est nulle (on vérifie l'équation d'évaluation de Bellman). Si une action menant à un plus grand cumul de récompenses est choisie, alors en moyenne l'erreur de différence temporelle est positive, d'amplitude d'autant plus grande que l'on améliore la qualité du contrôle. Par contre, si l'action mène à un plus faible cumul, l'erreur sera négative en moyenne. Cette notion d'avantage, qui peut également être vue comme étant la différence entre la qualité d'un couple état-action et la valeur de cet état, va s'avérer importante pour la suite. En effet, pour un état donné, ce n'est pas tant la valeur de chaque action qui est importante que leurs valeurs relatives. Nous y reviendrons.

Nous avons présenté cet algorithme dans un but pédagogique, afin d'illustrer simplement les idées sous-jacentes aux architectures acteur-critique. Historiquement, les premiers algorithmes d'apprentissage par renforcement étaient du type acteur-critique. Il y a deux raisons principales à cela : d'une part, ce type d'architecture est la base neuro-physiologique du renforcement [97] ; d'autre part, avant l'introduction de la fonction de qualité [121], il y avait peu d'autres solutions pour se passer explicitement du modèle du MDP. Après l'introduction de cette Q-fonction, la recherche a eu tendance à se focaliser sur les méthodes de type critique pur, au détriment des architectures acteur-critique. L'un des problèmes de ces

dernières était la difficulté de les combiner de façon satisfaisante à des schémas d'approximation de la fonction de valeur. A la fin des années 90, des travaux comme [71] ou [107] on relancé la recherche dans ce domaine en introduisant un point de vue plus formel de la combinaison de ce type d'architectures avec des schémas d'approximation. L'introduction du gradient naturel pour la phase d'amélioration de la politique [83] a également contribué à populariser ces approches. L'algorithme que nous proposons est basé sur ces travaux (ainsi que sur les différences temporelles de Kalman), nous les résumons maintenant.

7.2 Un peu de théorie

Nous commençons par poser quelques notations et fournir quelques résultats nécessaires, notamment concernant le gradient de la politique, la notion de paramétrisation compatible ainsi que le gradient naturel d'une politique. La première étape est une réécriture de la performance de la politique, nécessaire à l'expression analytique de son gradient.

Supposons qu'il y ait un état initial s_0 (ceci n'est pas restrictif pour notre discours, il suffit de remplacer l'état initial par une distribution sur les états initiaux). Notons $r(s,a)$ la récompense moyenne du couple état-action (s,a) :

$$r(s,a) = E_{s'|s,a}\left[r(s,a,s')\right] = \sum_{s' \in S} p(s'|s,a) r(s,a,s') \tag{7.5}$$

Comme s_0 est l'état initial, la performance d'une politique π est la fonction de valeur prise en cet état initial, c'est-à-dire $V^\pi(s_0)$, que l'on réécrit :

$$\rho(\pi) = E\left[\sum_{i=0}^{\infty} \gamma^i r_i | s_0, \pi\right] \tag{7.6}$$

Nous définissons $d^\pi(s)$, la pondération actualisée de la probabilité de visiter l'état s au cours de la trajectoire en démarrant dans l'état s_0 et en suivant la politique π :

$$d^\pi(s) = (1-\gamma) \sum_{i=0}^{\infty} \gamma^i \Pr(S_i = s | s_0, \pi) \tag{7.7}$$

Le cumul espéré de récompenses partant de s_0 et suivant π, autrement dit la performance de la politique, peut se réécrire en fonction de d^π.

Propriété 7.1 (Réécriture de ρ). *La mesure de performance $\rho(\pi)$ de la politique π peut se réécrire :*

$$\rho(\pi) = \frac{1}{1-\gamma} \sum_{s \in S} d^\pi(s) \sum_{a \in A} \pi(a|s) r(s,a) \tag{7.8}$$

Démonstration. Montrer ce résultat est un simple travail de réécriture [107].

$$\begin{aligned}
\rho(\pi) &= E\left[\sum_{i=0}^{\infty} \gamma^i r_i | s_0, \pi\right] \\
&= \sum_{i=0}^{\infty} \gamma^i E\left[r_i | s_0, \pi\right] \\
&= \sum_{i=0}^{\infty} \gamma^i \sum_{s,a \in S \times A} \Pr(S_i = s, A_i = a | s_0, \pi) r(s,a) \\
&= \sum_{i=0}^{\infty} \gamma^i \left(\sum_{s \in S} \Pr(S_i = s | s_0, \pi) \sum_{a \in A} \pi(a|s) r(s,a)\right) \\
&= \sum_{s \in S} \left(\sum_{i=0}^{\infty} \gamma^i \Pr(S_i = s | s_0, \pi)\right) \sum_{a \in A} \pi(a|s) r(s,a) \qquad (7.9)
\end{aligned}$$

Ceci montre le résultat. Le facteur $1 - \gamma$ est introduit de façon à ce que d^π soit une distribution (il faut pour cela que d^π somme à un sur l'ensemble des états). Etant donné que l'on cherche la politique qui maximise ce critère, l'introduction de ce facteur ne change rien au problème d'optimisation sous-jacent. □

7.2.1 Gradient de la performance

Soit la politique π_ω paramétrée par le vecteur ω. L'objectif est de déterminer le vecteur de paramètres qui maximise la performance de la politique associée, que nous notons $\rho(\omega)$ par abus de notation :

$$\omega^* = \operatorname*{argmax}_\omega \rho(\pi_\omega) = \operatorname*{argmax}_\omega \rho(\omega) \qquad (7.10)$$

Les approches de gradient sur la politique (qui sont du type acteur pur, d'après la dénomination que nous avons donnée précédemment), consistent à corriger les paramètres proportionnellement au gradient de la performance ρ (montée de gradient) :

$$\Delta\omega = \alpha \nabla_\omega \rho(\omega) \qquad (7.11)$$

Les approches varient principalement selon leur façon d'estimer ce gradient. Un résultat important exprime analytiquement ce dernier à l'aide de la fonction de qualité de la politique associée. Le résultat en question est souvent appelé *Policy Gradient Theorem*, ce que nous traduisons par gradient-politique, dans la mesure où c'est le gradient de la mesure de performance que l'on cherche à maximiser par rapport aux paramètres de la politique qui est calculé.

Théorème 7.1 (Gradient-politique). *Soit $b \in \mathbb{R}^S$ une fonction quelconque sur les états à valeurs réelles. Pour le coût γ-pondéré, le gradient de la performance ρ s'écrit :*

$$\nabla_\omega \rho(\omega) = \frac{1}{1-\gamma} \sum_{s \in S} d^{\pi_\omega}(s) \sum_{a \in A} \left(\nabla_\omega \pi_\omega(a|s)\right) \left(Q^{\pi_\omega}(s,a) + b(s)\right) \qquad (7.12)$$

Démonstration. Ce résultat, qui est également valable pour le coût dit moyen (la définition de d^π étant alors différente), est prouvé par [107] dans le cas où $b = 0$. L'extension à $b \in \mathbb{R}^S$

quelconque se retrouve dans [107, 13] sous des formes légèrement différentes (dans le sens où ce résultat est énoncé, mais pas dans le théorème même). Toujours est-il que montrer que la relation est vraie pour tout fonction b est simple. Il suffit de remarquer que :

$$\begin{aligned}\sum_{a\in A}\nabla_\omega\pi_\omega(a|s)b(s) &= b(s)\sum_{a\in A}\nabla_\omega\pi_\omega(a|s)\\ &= b(s)\nabla_\omega\left(\sum_{a\in A}\pi_\omega(a|s)\right)\\ &= b(s)\nabla_\omega(1) = 0 \end{aligned} \tag{7.13}$$

□

Un aspect important de ce gradient est qu'il ne fait pas intervenir de terme de la forme $\nabla_\omega d^{\pi_\omega}$. L'effet du changement de la politique sur la distribution des états n'apparaît donc pas, ce qui est pratique d'un point de vue échantillonnage (pour obtenir des estimateurs non biaisés du gradient). Le problème pour estimer ce gradient est que Q^π n'est pas connu *a priori*. Une première approche pourrait être de l'estimer avec une simulation de Monte-Carlo, ce qui est fait par exemple dans [124] et est une méthode acteur pur.

7.2.2 Gradient et approximation de la valeur

Une autre approche, qui semble plus intéressante, est de remplacer Q^π par une estimation obtenue par exemple avec une méthode de différences temporelles. Cela est possible, sous certaines conditions, grâce à un résultat proposé simultanément par [107] et [71] (cependant seulement pour le coût moyen dans [71]). Avant de l'énoncer, nous définissons la notion de paramétrisation compatible.

Définition 7.1 (Paramétrisation compatible). *Soit π_ω une politique paramétrée par ω, telle que la probabilité de toute action soit strictement positive et f_θ une fonctionnelle paramétrée par le vecteur de paramètres θ. La paramétrisation f_θ est dite* compatible *avec la politique paramétrée π_ω si :*

$$\nabla_\theta f_\theta = \nabla_\omega \ln(\pi_\omega) \tag{7.14}$$

Il est à noter qu'*a priori* la seule façon de vérifier cette condition est que f_θ soit linéaire en ces caractéristiques :

$$f_\theta(s,a) = \left(\nabla_\omega \ln(\pi_\omega)(a|s)\right)^T\theta + \text{cte} \tag{7.15}$$

Dans la suite de ce chapitre, sauf mention du contraire, nous supposerons le terme constant nul. Cette notion de paramétrisation compatible ayant été introduite, nous pouvons énoncer un second résultat, qui permet d'exprimer le gradient de la performance par rapport aux paramètres de la politique en fonction d'une approximation de la fonction de valeur.

Théorème 7.2 (Gradient-politique avec approximation). *Soient $b_1, b_2 \in \mathbb{R}^S$ deux fonctionnelles quelconques. Soit f_θ compatible avec la paramétrisation de la politique π_ω. Supposons que f_θ soit une "bonne" approximation de Q^{π_ω}, dans le sens où θ est un minimum local de l'erreur quadratique entre la fonction de qualité et sont approximation :*

$$\nabla_\theta\left(\sum_{s\in S}d^{\pi_\omega}(s)\sum_{a\in A}\pi_\omega(a|s)\left(Q^{\pi_\omega}(s,a) - b_1(s) - f_\theta(s,a)\right)^2\right) = 0 \tag{7.16}$$

Alors le gradient de ρ par rapport aux paramètres de la politique peut se réécrire en fonction de l'approximation de la Q-fonction :

$$\nabla_\omega \rho(\omega) = \frac{1}{1-\gamma} \sum_{s \in S} d^{\pi_\omega}(s) \sum_{a \in A} \nabla_\omega \pi_\omega(a|s) \left(f_\theta(s,a) + b_2(s) \right) \tag{7.17}$$

Démonstration. Ce résultat est également montré par [107], sous une forme légèrement différente : la condition de gradient sur θ (7.16) est exprimée sous une forme différente (mais équivalente) et il n'y a pas les bases b_1 et b_2. Nous avons déjà montré dans le théorème 7.1 que l'introduction de la base b_2 ne modifiait pas le gradient. La preuve est similaire pour la condition de bonne approximation :

$$\begin{aligned}
&\nabla_\theta \left(\sum_{s \in S} d^{\pi_\omega}(s) \sum_{a \in A} \pi_\omega(a|s) \left(Q^{\pi_\omega}(s,a) - b_1(s) - f_\theta(s,a) \right)^2 \right) = 0 \\
&\Leftrightarrow \sum_{s \in S} d^{\pi_\omega}(s) \sum_{a \in A} \pi_\omega(a|s) \nabla_\theta \left(Q^{\pi_\omega}(s,a) - b_1(s) - f_\theta(s,a) \right)^2 = 0 \\
\Leftrightarrow &\sum_{s \in S} d^{\pi_\omega}(s) \sum_{a \in A} \pi_\omega(a|s) \left(Q^{\pi_\omega}(s,a) - b_1(s) - f_\theta(s,a) \right) \nabla_\theta f_\theta(s,a) = 0
\end{aligned} \tag{7.18}$$

Or f_θ vérifie la condition de compatibilité, donc :

$$\begin{aligned}
\sum_{a \in A} \pi_\omega(a|s) b_1(s) \nabla_\theta f_\theta(s,a) &= b_1(s) \sum_{a \in A} \pi_\omega(a|s) \nabla_\omega \ln(\pi_\omega) \\
&= b_1(s) \nabla_\omega \left(\sum_{a \in A} \pi_\omega(a|s) \right) = 0
\end{aligned} \tag{7.19}$$

Ainsi, nous avons :

$$\begin{aligned}
&\Leftrightarrow \sum_{s \in S} d^{\pi_\omega}(s) \sum_{a \in A} \pi_\omega(a|s) \left(Q^{\pi_\omega}(s,a) - b_1(s) - f_\theta(s,a) \right) \nabla_\theta f_\theta(s,a) = 0 \\
&\Leftrightarrow \sum_{s \in S} d^{\pi_\omega}(s) \sum_{a \in A} \pi_\omega(a|s) \left(Q^{\pi_\omega}(s,a) - f_\theta(s,a) \right) \nabla_\theta f_\theta(s,a) = 0 \\
&\Leftrightarrow \nabla_\theta \left(\sum_{s \in S} d^{\pi_\omega}(s) \sum_{a \in A} \pi_\omega(a|s) \left(Q^{\pi_\omega}(s,a) - f_\theta(s,a) \right)^2 \right) = 0
\end{aligned} \tag{7.20}$$

Notons que l'avant dernière équivalence est la condition de bonne approximation telle qu'exprimée par [107]. On retrouve alors la même formulation que dans cet article, où est montré le résultat. □

Quelques commentaires concernant ce résultat peuvent être faits. Tout d'abord, les bases b_1 et b_2 n'affectent pas les résultats, mais comme noté par [107], ce choix peut influer sur la variance de l'estimation du gradient. Sans perte de généralité, on peut donc supposer $b_1 = 0$. De plus, il est montré dans [14, 13] (pour le critère moyen cependant et non γ-pondéré) que la base b optimale (théorème 7.1) d'un point de vue minimisation de la variance de l'estimateur de la Q-fonction est en fait la fonction de valeur : $b = -V^{\pi_\omega}$. La

fonction f_θ est centrée pour chaque état (respectivement aux actions) :

$$\begin{aligned}\sum_{a\in A}\pi_\omega(a|s)f_\theta(s,a) &= \sum_{a\in A}\pi_\omega(a|s)\left(\nabla_\omega \ln(\pi_\omega(a|s))\right)^T\theta \\ &= \theta^T\sum_{a\in A}\pi_\omega(a|s)\nabla_\omega \ln(\pi_\omega(a|s)) \\ &= \theta^T\sum_{a\in A}\nabla_\omega\pi_\omega(a|s) = 0 \end{aligned} \tag{7.21}$$

Il vaut donc mieux penser à f_θ comme une approximation de la fonction *avantage* [5] qu'à une Q-fonction, cette fonction avantage étant définie par :

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) \tag{7.22}$$

Nous avons introduit cette notion d'avantage dans la section 7.1. Elle quantifie le gain qu'apporte une action donnée par rapport à celle préconisée par la politique, en terme de cumul de récompenses. Le problème de cette fonction avantage, comme nous le verrons par la suite, est qu'elle ne peut pas être définie à partir d'une équation de Bellman. D'autre part, en supposant le résultat de [13] sur la base optimale également correct dans le cas du critère γ-pondéré, la base b_2 peut être considérée comme nulle. Ces remarques concernant ces bases, qui ne changent pas les résultats (excepté éventuellement d'un point de vue variance des estimations), sont intéressantes dans la mesure où elle mettent en avant le fait que ce n'est pas la valeur absolue d'un couple état-action qui est importante, mais la différence de valeur entre les différentes actions, pour un même état, qui l'est.

Il est à noter que ce théorème de gradient avec approximation de la fonction de valeur a également (et parallèlement) été donné par [71, 72] dans le cas du critère moyen. Le résultat n'est pas tout à fait identique, notamment la condition de compatibilité est plus large (il est demandé une inclusion plutôt qu'une égalité), ce qui permet une représentation plus riche pour f_θ, mais basiquement l'idée est la même. A partir de ces résultats il est possible de dériver des algorithmes pratiques, voir par exemple les algorithmes de [71] ou l'algorithme 1 de [14]. L'idée générale est de choisir un approximateur de la fonction de valeur qui joue le rôle de critique, tout en utilisant une représentation compatible, puis de corriger les paramètres de la politique en effectuant une montée de gradient, ce dernier étant estimé à l'aide de l'approximation de la valeur. Cependant, l'approche que nous proposons utilise la notion de gradient naturel plutôt que celle de gradient classique.

7.2.3 Gradient naturel

L'idée du gradient naturel [1] est de suivre la plus grande pente relativement à la métrique de Fisher plutôt que dans l'espace des paramètres, comme cela est fait pour le gradient classique. Le gradient naturel $\tilde{\nabla}$ est défini par :

$$\tilde{\nabla}_\omega\rho(\omega) = G^{-1}(\omega)\nabla_\omega\rho(\omega) \tag{7.23}$$

Dans cette expression, $G(\omega)$ est la matrice d'information de Fisher, elle est définie par (voir par exemple [83] ou [13]) :

$$G(\omega) = E_{s\sim d^{\pi_\omega},a\sim\pi_\omega}\left[\nabla_\omega \ln(\pi_\omega(a|s))\nabla_\omega \ln(\pi_\omega(a|s))^T\right] \tag{7.24}$$

En apprentissage supervisé il a été montré que le gradient naturel (qui est une approche du second ordre, *via* la matrice de Fisher) pouvait permettre d'améliorer significativement les résultats par rapport au gradient classique [1]. L'approche a également été utilisée dans un contexte d'acteur pur par [63] (montée de gradient sur la politique, en utilisant un gradient naturel plutôt que le gradient classique). Dans le contexte qui nous intéresse, le problème qui peut se poser est l'estimation de l'inverse de la matrice de Fisher. Cependant, un résultat important permet de s'en affranchir [83, 82, 81].

Théorème 7.3 (Gradient naturel de la performance). *Soit f_θ une approximation de la fonction avantage vérifiant les hypothèses du théorème 7.2. Le gradient naturel vérifie alors :*

$$\tilde{\nabla}_\omega \rho(\omega) = \theta \tag{7.25}$$

Démonstration. D'après le théorème 7.2, sous ces hypothèses nous avons :

$$\nabla_\omega \rho(\omega) = \frac{1}{1-\gamma} \sum_{s \in S} d^{\pi_\omega}(s) \sum_{a \in A} \nabla_\omega \pi_\omega(a|s) f_\theta(s,a) \tag{7.26}$$

Or d'une part nous avons toujours pour le gradient du logarithme d'une fonctionnelle strictement positive (en l'occurrence la politique ici) :

$$\nabla_\omega \ln(\pi_\omega(a|s)) = \frac{\nabla_\omega \pi_\omega(a|s)}{\pi_\omega(a|s)} \tag{7.27}$$

et d'autre part la condition de compatibilité permet d'écrire :

$$f_\theta(s,a) = \left(\nabla_\omega \ln(\pi_\omega(a|s))\right)^T \theta \tag{7.28}$$

Le gradient peut donc s'écrire :

$$\nabla_\omega \rho(\omega) = \frac{1}{1-\gamma} \left(\sum_{s \in S} d^{\pi_\omega}(s) \sum_{a \in A} \pi_\omega(a|s) \nabla_\omega \ln(\pi_\omega(a|s)) \nabla_\omega \ln(\pi_\omega(a|s))^T \right) \theta \tag{7.29}$$

Il est montré par [83] (entre autre) que c'est en fait la matrice d'information de Fisher :

$$G(\omega) = \sum_{s \in S} d^{\pi_\omega}(s) \sum_{a \in A} \pi_\omega(a|s) \nabla_\omega \ln(\pi_\omega(a|s)) \nabla_\omega \ln(\pi_\omega(a|s))^T \tag{7.30}$$

Ceci prouve le résultat :

$$\tilde{\nabla}_\omega \rho(\omega) = G^{-1}(\omega) \nabla_\omega \rho(\omega) = G^{-1}(\omega) G(\omega) \theta = \theta \tag{7.31}$$

□

Le gradient naturel présente plusieurs avantages. Premièrement, une convergence vers un minimum local est garantie, comme pour le gradient classique. D'autre part, dans la mesure où il choisit un chemin plus "direct" vers la solution, le gradient naturel a empiriquement une convergence plus rapide et moins sensible aux optima locaux. Le gradient naturel peut également être montré "covariant" [81], ce qui signifie qu'il est indépendant des coordonnées choisies pour exprimer les paramètres de la politique. Enfin, d'après les résultats du théorème 7.3, le gradient naturel n'est pas échantillonné, il prend directement en compte le moyennage sur les distributions des actions et des états, ce qui laisse penser qu'il demande moins de données pour être estimé que le gradient classique. Maintenant que nous avons présenté les quelques bases nécessaires (notamment gradient avec approximation de la valeur et gradient naturel), nous pouvons proposer des algorithmes acteur-critique.

7.3 Algorithmes

Le principe des algorithmes que nous présentons ici est le suivant. Une paramétrisation est choisie pour la politique. Une paramétrisation compatible est adoptée pour la fonction avantage. Le rôle du critique est d'apprendre cette fonction avantage pour la paramétrisation donnée, le rôle de l'acteur est de corriger la représentation de la politique, en suivant un gradient naturel et selon les résultats fournis par le critique. Nous considérons des algorithmes en ligne, c'est-à-dire que l'acteur et le critique mettent à jour leur représentation respective à chaque nouvelle observation. Nous présentons deux algorithmes, qui diffèrent par le critique considéré. Le premier algorithme a été proposé par [79] dans le cadre du critère γ-pondéré, puis plus tard par [13] (algorithme 3) dans le cadre du critère moyen (et des prémisses peuvent être trouvées dans [69]). Il consiste à utiliser un algorithme TD classique (TD avec approximation de la fonction de valeur, aussi appelé TD direct) comme critique. Dans ce cas, l'acteur est donc d'ordre 2 et le critique d'ordre 1. Le deuxième algorithme que nous proposons, nouveau celui-là, utilise KTD(λ) comme critique. En conséquence, l'acteur et le critique sont tous les deux d'ordre 2 et basés sur le gradient naturel, plus ou moins directement (nous avons vu que KTD peut être vu comme effectuant une descente de gradient naturel), ce qui devrait permettre *a priori* un apprentissage plus rapide.

7.3.1 TD-NAC

Dans le type d'approche que nous discutons, le problème principal est l'estimation de la fonction avantage, ce qui est le rôle du critique. En effet, puisqu'on ne s'intéresse plus à la valeur absolue d'un couple état-action, mais à la différence de valeur des différentes actions possibles dans un même état, il n'est pas possible d'appliquer directement le principe des différences temporelles à l'estimation de la fonction avantage (il n'y a pas d'équation de Bellman propre à cette dernière, étant donné que l'information propre à l'état, sa valeur, disparaît). Il faut donc d'une façon ou d'une autre se ramener à estimer également la fonction de valeur. Par exemple, on peut exprimer une équation de Bellman qui dépend de la fonction avantage ainsi que de la fonction de valeur (la somme des deux étant égale d'après (7.22) à la fonction de qualité) :

$$A^\pi(s,a) + V^\pi(s) = r(s,a) + \gamma E_{s'|s,a}\left[V^\pi(s')\right] \tag{7.32}$$

De cette équation, on peut déduire une estimation de la fonction avantage qui dépend de l'erreur de différence temporelle relative à la fonction de valeur, comme nous l'avons annoncé section 7.1 :

$$A^\pi(s,a) = r(s,a) + \gamma E_{s'|s,a}\left[V^\pi(s')\right] - V^\pi(s) \tag{7.33}$$

Le principe de l'algorithme TD-NAC (qui n'a pas de nom particulier dans [14, 13] et qui est appelé NTD pour *Natural policy gradient utilizing the Temporal Difference* dans [79]) est de maintenir une paramétrisation propre à la fonction de valeur, d'estimer cette dernière avec un schéma classique de différences temporelles, puis d'utiliser l'erreur de différence temporelle associée comme cible pour la fonction avantage. Plus formellement, supposons la fonction de valeur $\hat{V}_\xi$ associée à la politique π_ω paramétrée par le vecteur ξ. La fonction de valeur est mise à jour classiquement, pour une transition (s_i, s_{i+1}) et une récompense r_i

observées :

$$\xi_i = \xi_{i-1} + \alpha_i \nabla_\xi \hat{V}_{\xi_{i-1}}(s_i)\delta_i \tag{7.34}$$

$$\text{avec } \delta_i = r_i + \gamma \hat{V}_{\xi_{i-1}}(s_{i+1}) - \hat{V}_{\xi_{i-1}}(s_i) \tag{7.35}$$

Il est également nécessaire de mettre à jour la représentation de la fonction avantage. Comme pour l'algorithme TD on utilise la récompense plus une estimation de la valeur en l'état suivant comme estimation de la valeur en l'état courant, pour estimer la fonction avantage on utilise également une forme de bootstrapping en remplaçant l'observation de l'avantage (qui ne peut pas être faite directement, comme ce serait le cas en apprentissage supervisé) par l'erreur de différence temporelle. Le taux d'apprentissage utilisé est le même. Ceci donne la mise à jour suivante (en rappelant que pour la paramétrisation de la fonction avantage la condition de compatibilité est vérifiée) :

$$\begin{aligned}\theta_i &= \theta_{i-1} + \alpha_i \nabla_\theta f_{\theta_{i-1}}(s_i, a_i)\left(\delta_i - f_{\theta_{i-1}}(s_i, a_i)\right) \\ &= \theta_{i-1} + \alpha_i \nabla_\omega \ln(\pi_{\omega_{i-1}}(a_i|s_i))\left(\delta_i - \nabla_\omega \ln(\pi_{\omega_{i-1}}(a_i|s_i))^T \theta_{i-1}\right)\end{aligned} \tag{7.36}$$

Ceci clôt la mise à jour du critique. Une fois cette dernière faite, il ne reste plus qu'à mettre à jour l'acteur. Pour cela, nous appliquons la mise à jour (7.11), en remplaçant le gradient classique par le gradient naturel et en utilisant le résultat du théorème 7.3 :

$$\omega_i = \omega_{i-1} + \beta_i \theta_i \tag{7.37}$$

Le taux d'apprentissage β_i est propre à l'acteur. De plus, nous suivons [13] et adoptons deux échelles de temps différentes pour l'acteur et le critique. L'idée naturelle derrière cela est que du point de vue du critique, l'acteur doit paraître stationnaire. Autrement dit le critique doit converger plus vite que l'acteur. Concernant les taux d'apprentissage, cela se traduit par :

$$\sum_{i=0}^{\infty} \alpha_i = \sum_{i=0}^{\infty} \beta_i = \infty \quad , \quad \sum_{i=0}^{\infty} \alpha_i^2, \sum_{i=0}^{\infty} \beta_i^2 < \infty \quad , \quad \lim_{i\to\infty} \frac{\beta_i}{\alpha_i} = 0 \tag{7.38}$$

Si le critère considéré est le critère moyen (ce qui ne change que l'expression de l'erreur de différence temporelle) et si la fonction de valeur est paramétrée linéairement, alors il est possible de montrer que l'algorithme converge [14]. Ceci joue en faveur de l'approche, mais nous ne proposons pas de preuve de convergence pour le critère γ-pondéré. L'approche TD-NAC proposée est résumée dans l'algorithme 7.1.

7.3.2 KNAC

Nous avons baptisé l'approche présentée maintenant KNAC pour *Kalman Natural Actor-Critic*. La mise à jour de l'acteur reste la même, mais au lieu d'utiliser TD pour mettre à jour le critique nous utilisons KTD(λ). Il y a plusieurs raisons à cela. Premièrement, comme KTD traque la solution plutôt qu'il ne converge vers elle, il n'y a pas de nécessite *a priori* de considérer deux échelles de temps distinctes (il serait d'ailleurs difficile de spécifier directement une échelle de temps pour KTD). D'autre part, KTD est une méthode du second ordre, que nous avons dit être proche d'une descente de gradient naturel (voir le chapitre 3). On peut ainsi espérer un apprentissage plus rapide.

Algorithme 7.1 : TD-NAC

Initialisations;
Paramètres pour la politique ω_0 ;
Paramètres pour la fonction de valeur ξ_0 ;
Paramètres pour la fonction avantage θ_0 ;
Etat initial s_1 ;

pour $i \leftarrow 1, 2, \ldots$ **faire**

Exécution;
Echantillonnage d'une action $a_i \sim \pi_{\omega_{i-1}}(.|s_i)$;
Observer la transition $s_{i+1} \sim p(.|s_i, a_i)$ ainsi que la récompense associée r_i ;

Mise à jour du critique;
$\delta_i = r_i + \gamma \hat{V}_{\xi_{i-1}}(s_{i+1}) - \hat{V}_{\xi_{i-1}}(s_i)$;
$\xi_i = \xi_{i-1} + \alpha_i \nabla_\xi \hat{V}_{\xi_{i-1}}(s_i)\delta_i$;
$\theta_i = \theta_{i-1} + \alpha_i \nabla_\omega \ln(\pi_{\omega_{i-1}}(a_i|s_i)) \left(\delta_i - \nabla_\omega \ln(\pi_{\omega_{i-1}}(a_i|s_i))^T \theta_{i-1}\right)$;

Mise à jour de l'acteur ;
$\omega_i = \omega_{i-1} + \beta_i \theta_i$;

Rappelons l'équation de Bellman dans laquelle apparaissent les fonctions avantage et de valeur :

$$A^\pi(s,a) + V^\pi(s) = r(s,a) + \gamma E_{s'|s,a}\left[V^\pi(s')\right] \tag{7.39}$$

L'objectif du critique est d'estimer la fonction avantage, ce qui passe comme nous l'avons vu par l'estimation de la fonction de valeur. Une première approche pourrait être de considérer deux mises à jour successives, comme pour TD-NAC. On aurait ainsi deux filtres : le premier, dont l'état caché serait l'ensemble des paramètres de la fonction de valeur, serait KTD-V(λ), l'observation étant classiquement la récompense ; le second, dont l'état caché serait l'ensemble des paramètres de la fonction avantage, disposerait d'une équation d'observation liant l'erreur de différence temporelle (qui peut être vue comme une pseudo-observation) à l'estimation de la fonction avantage. Cependant, avec deux filtres séparés, toute information de corrélation entre les paramètres de la fonction de valeur et ceux de la fonction avantage serait perdue. Nous préférons considérer le problème d'un bloc.

L'algorithme

Pour cela, il faut exprimer le problème de l'approximation jointe de la fonction de valeur et de la fonction avantage sous la forme d'un filtre, comme nous l'avons déjà fait pour l'estimation de la fonction de valeur ou de qualité dans les chapitres précédents. Nous considérons l'ensemble des paramètres ξ et θ comme faisant partie de l'état caché, pour lesquels nous adoptons comme d'habitude un modèle de marche aléatoire. Reste à adopter un modèle d'observation. Comme auparavant, l'observation est la récompense. Cependant, l'équation d'observation est modifiée. Pour la déterminer, nous réécrivons l'équation (7.39) pour une transition donnée (s_i, a_i, r_i, s_{i+1}), en tenant compte des paramétrisations et en introduisant un terme de bruit n_i, modélisant à la fois la stochasticité du MDP et le biais inductif (que nous rappelons être l'incapacité potentielle de l'approximateur à représenter

exactement la fonction d'intérêt) :

$$f_\theta(s_i, a_i) + \hat{V}_\xi(s_i) = r_i + \gamma \hat{V}_\xi(s_{i+1}) + n_i \tag{7.40}$$

En isolant la récompense r_i dans cette dernière équation, on obtient un processus d'observation. Le bruit n_i est donc le bruit d'observation et nous le choisissons coloré, paramétré par le facteur d'éligibilité λ, tel que décrit dans le chapitre 5. Il faut donc étendre le vecteur d'état avec les composantes de bruit *ad hoc*. Tout ceci suggère le modèle espace-d'état suivant :

$$\begin{cases} \mathbf{x}_i = F(\lambda)\mathbf{x}_{i-1} + v'_i \\ r_i = f_{\theta_i}(s_i, a_i) + \hat{V}_{\xi_i}(s_i) - \gamma \hat{V}_{\xi_i}(s_{i+1}) + n_i \end{cases} \tag{7.41}$$

Dans ce nouveau modèle espace-d'état, le vecteur étendu de paramètres $\mathbf{x}_i$ prend en compte la paramétrisation de la fonction avantage θ_i, la paramétrisation de la fonction de valeur ξ_i ainsi que la structure du bruit d'observation $\begin{pmatrix} b_i & n_i \end{pmatrix}^T$ si l'on reprend les notations du chapitre 5 :

$$\mathbf{x}_i^T = \begin{pmatrix} \theta_i^T & \xi_i^T & b_i & n_i \end{pmatrix} \tag{7.42}$$

Le bruit d'observation est donc estimé en même temps que les autres paramètres. La matrice d'évolution $F(\lambda)$ rend compte principalement de la structure du bruit d'observation, les autres paramètres étant toujours supposés suivre une marche aléatoire (ce qui correspond à une matrice d'évolution égale à l'identité). Si l'on note p la dimension de θ et q la dimension de ξ, la matrice d'évolution est donc donnée par :

$$F(\lambda) = \begin{pmatrix} I_{p+q} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \gamma\lambda & 0 \\ \mathbf{0}^T & -\gamma(1-\lambda) & 0 \end{pmatrix} \tag{7.43}$$

Le bruit d'évolution v'_i est la concaténation du bruit v_i sur les paramètres de la fonction avantage et de la fonction de valeur (qui doit être choisi par le praticien) et du bruit propre à la structure colorée du bruit d'observation, fonction du bruit blanc centré u_i de variance σ_i^2 :

$$v'_i = \begin{pmatrix} v_i \\ u_i \\ u_i \end{pmatrix} \sim \left(\begin{pmatrix} \mathbf{0} \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} P_{v_i} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \sigma_i^2 & \sigma_i^2 \\ \mathbf{0}^T & \sigma_i^2 & \sigma_i^2 \end{pmatrix} \right) \tag{7.44}$$

Enfin, l'équation d'observation lie la récompense observée aux fonctions avantage et de valeur *via* la paramétrisation choisie (que nous rappelons devoir être compatible en ce qui concerne la fonction avantage). Nous réécrivons le modèle espace d'état complet :

$$\begin{cases} \begin{pmatrix} \theta_i \\ \xi_i \\ b_i \\ n_i \end{pmatrix} = \begin{pmatrix} I_{p+q} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \gamma\lambda & 0 \\ \mathbf{0}^T & -\gamma(1-\lambda) & 0 \end{pmatrix} \begin{pmatrix} \theta_{i-1} \\ \xi_{i-1} \\ b_{i-1} \\ n_{i-1} \end{pmatrix} + \begin{pmatrix} v_i \\ u_i \\ u_i \end{pmatrix} \\ r_i = \nabla_\omega \ln(\pi_{\omega_{i-1}}(a_i|s_i))^T \theta_i + \hat{V}_{\xi_i}(s_i) - \gamma \hat{V}_{\xi_i}(s_{i+1}) + n_i \end{cases} \tag{7.45}$$

Une fois ce modèle espace d'état posé, l'algorithme KTD(λ) peut aisément être adapté, il suffit de remplacer l'équation d'observation usuelle par celle que nous proposons. En combinant cette évaluation du critique avec la correction de l'acteur utilisant le gradient naturel, nous obtenons KNAC (*Kalman Natural Actor Critic*), résumé dans l'algorithme 7.2.

Algorithme 7.2 : KNAC

Initialisations;
Paramètres pour la politique ω_0 ;
a priori $\mathbf{x}_{0|0}$ et $P_{0|0}$;
Etat initial s_1 ;

pour $i \leftarrow 1, 2, \ldots$ **faire**

Exécution;
Echantillonnage d'une action $a_i \sim \pi_{\omega_{i-1}}(.|s_i)$;
Observer la transition $s_{i+1} \sim p(.|s_i, a_i)$ ainsi que la récompense associée r_i ;

Mise à jour du critique;
début

Phase de prédiction;
$\hat{\mathbf{x}}_{i|i-1} = F(\lambda)\hat{\mathbf{x}}_{i-1|i-1}$;
$P_{i|i-1} = F(\lambda)P_{i-1|i-1}F(\lambda)^T + P_{v'_{i-1}}$;

Calcul des sigma-points ;
$\mathbf{X}_{i|i-1} = \left\{\hat{\mathbf{x}}^{(j)}_{i|i-1}, \quad 0 \leq j \leq 2(p+q)\right\}$ (en utilisant $\hat{\mathbf{x}}_{i|i-1}$ et $P_{i|i-1}$);
$\mathcal{W} = \{w_j, \quad 0 \leq j \leq 2(p+q) \quad \}$;
`/* notons que` $(\hat{\mathbf{x}}^{(j)}_{i|i-1})^T = \left((\hat{\theta}^{(j)}_{i|i-1})^T \quad (\hat{\xi}^{(j)}_{i|i-1})^T \quad \hat{b}^{(j)}_{i|i-1} \quad \hat{n}^{(j)}_{i|i-1}\right)$ `*/`
$\mathcal{R}_{i|i-1} = \left\{\hat{r}^{(j)}_{i|i-1} = \nabla_\omega \ln(\pi_{\omega_{i-1}}(a_i|s_i))^T\hat{\theta}^{(j)}_{i|i-1} + \hat{V}_{\hat{\xi}^{(j)}_{i|i-1}}(s_i) - \gamma\hat{V}_{\hat{\xi}^{(j)}_{i|i-1}}(s_{i+1}) + \right.$
$\left.\hat{n}^{(j)}_{i|i-1}, 0 \leq j \leq 2(p+q)+4\right\}$;

Calcul des statistiques d'intérêt;
$\hat{r}_{i|i-1} = \sum_{j=0}^{2(p+q)+4} w_j \hat{r}^{(j)}_{i|i-1}$;
$P_{\mathbf{x}r_i} = \sum_{j=0}^{2(p+q)+4} w_j(\hat{\mathbf{x}}^{(j)}_{i|i-1} - \hat{\mathbf{x}}_{i|i-1})(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})$;
$P_{r_i} = \sum_{j=0}^{2(p+q)+4} w_j \left(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1}\right)^2$;

Phase de correction;
$K_i = P_{\mathbf{x}r_i}P_{r_i}^{-1}$;
$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_{i|i-1} + K_i\left(r_i - \hat{r}_{i|i-1}\right)$;
$P_{i|i} = P_{i|i-1} - K_iP_{r_i}K_i^T$;

fin

Mise à jour de l'acteur ;
$\omega_i = \omega_{i-1} + \beta_i\hat{\theta}_{i|i}$;

Un point de vue alternatif

Jusqu'à présent, nous avons suivi d'assez près la littérature sur le sujet. Cependant, il est possible d'adopter un point de vue alternatif qui permet de déduire plus naturellement des critiques. En effet, pour tous les résultats que nous avons présentés dans la section 7.2 (et qui sont très proches de leur forme d'origine), c'est la paramétrisation f_θ qui est centrale et qui sous les hypothèses faites est une approximation de la fonction avantage. Ceci pose problème pour la déduction d'un critique, dans la mesure où la fonction avantage ne peut pas être (directement) donnée par une équation de Bellman.

Cependant, un autre point central de tous ces résultats est qu'ils restent vrais à une fonction de base $b \in \mathbb{R}^S$ près, du moment que cette dernière dépend uniquement des états et non des actions. On pourrait introduire la paramétrisation suivante de la Q-fonction, qui dépend d'un ensemble de paramètres θ pour la partie relative au couple état-action et d'un autre ensemble de paramètres ξ pour la partie relative à l'état seul :

$$\hat{Q}_{\theta,\xi}(s,a) = f_\theta(s,a) + \hat{V}_\xi(s) \tag{7.46}$$

Un point important est que, par rapport aux paramètres θ, $\hat{V}_\xi(s)$ est une constante (autrement dit $\nabla_\theta(\hat{V}_\xi(s))$=0). Dès lors, on pourrait étendre la notion de paramétrisation compatible aux expressions du type (7.46) en imposant l'égalité suivante :

$$\nabla_\theta \hat{Q}_{\theta,\xi}(s,a) = \nabla_\theta f_\theta(s,a) = \nabla_\omega \ln(\pi_\omega(a|s)) \tag{7.47}$$

La paramétrisation (7.46) étant posée et la notion de compatibilité adaptée, tous les résultats de la section 7.2 peuvent se réécrire en remplaçant f_θ par $\hat{Q}_{\theta,\xi}$ (nous ne le faisons pas, mais remarquons tout de même que l'élément central de cette réécriture est l'indépendance des différents résultats à l'ajout d'une base $b \in \mathbb{R}^S$). La seconde paramétrisation, contrairement à la première, peut vraiment être une approximation de la Q-fonction et est donc compatible avec les équations de Bellman.

Dès lors, le rôle du critique se résume à estimer la fonction de qualité, la paramétrisation discutée étant adoptée. En conséquence, on peut appliquer n'importe quel algorithme d'approximation de la Q-fonction, cette dernière étant paramétrée par le vecteur $[\theta^T, \xi^T]^T$. Par exemple, pour KTD-SARSA(λ), l'équation d'observation serait :

$$\begin{aligned} r_i &= \hat{Q}_{\theta_i,\xi_i}(s_i,a_i) - \gamma \hat{Q}_{\theta_i,\xi_i}(s_{i+1},a_{i+1}) + n_i \\ &= f_{\theta_i}(s_i,a_i) + \hat{V}_{\xi_i}(s_i) - \gamma \hat{V}_{\xi_i}(s_{i+1}) - \gamma f_{\theta_i}(s_{i+1},a_{i+1}) + n_i \end{aligned} \tag{7.48}$$

La différence avec l'équation d'observation du critique de KNAC est la présence du terme $-\gamma f_{\theta_i}(s_{i+1},a_{i+1})$. Nous rappelons que pour tout état s, $E_{a|s,\pi}[f_\theta(s,a)] = 0$, donc l'équation d'observation de KNAC peut être vue comme une moyenne selon les actions prises en l'état vers lequel transite le système de l'équation d'observation (7.48). Remplacer le terme $Q(s_{i+1},a_{i+1})$ par $E_{a|s_{i+1},\pi}[Q(s_{i+1},a]$ dans les algorithmes de type SARSA n'est pas nouveau, voir par exemple [123]. Ce point de vue alternatif permet donc d'obtenir un algorithme très proche et plus naturellement.

Le même raisonnement peut être appliqué pour obtenir une alternative à TD-NAC. En appliquant directement TD avec approximation de la valeur à $\hat{Q}_{\theta,\xi}$ paramétré par $[\theta^T, \xi^T]^T$, nous avons la mise à jour suivante :

$$\begin{pmatrix} \theta_i \\ \xi_i \end{pmatrix} = \begin{pmatrix} \theta_{i-1} \\ \xi_{i-1} \end{pmatrix} + \alpha_i \nabla_{\theta,\xi} \hat{Q}_{\theta_{i-1},\xi_{i-1}}(s_i,a_i) \left(r_i + \gamma \hat{Q}_{\theta_{i-1},\xi_{i-1}}(s_{i+1},a_{i+1}) - \hat{Q}_{\theta_{i-1},\xi_{i-1}}(s_i,a_i) \right) \tag{7.49}$$

Etant donnée la paramétrisation, cette mise à jour peut se réécrire de la façon suivante :

$$\begin{pmatrix}\theta_i\\ \xi_i\end{pmatrix} = \begin{pmatrix}\theta_{i-1}\\ \xi_{i-1}\end{pmatrix} + \alpha_i \begin{pmatrix}\nabla_\omega \ln(\pi_{\omega_{i-1}}(a_i|s_i))\\ \nabla_\xi \hat{V}_{\xi_{i-1}}(s_i)\end{pmatrix} \delta_i \tag{7.50}$$

$$\text{avec } \delta_i = r_i + \gamma \hat{V}_\xi(s_{i+1}) + \gamma f_\theta(s_{i+1}, a_{i+1}) - f_\theta(s_i, a_i) - \hat{V}_\xi(s_i) \tag{7.51}$$

On retrouve donc un algorithme très similaire à TD-NAC, la seule différence étant l'erreur de différence temporelle considérée, pour laquelle la même discussion que sur KNAC et son alternative peut être faite. Nous avons ainsi présenté un point de vue alternatif sur les architectures acteur-critique combinées à l'approximation de la valeur, le pensant intéressant. Cependant, nous n'expérimenterons que TD-NAC et KNAC dans la section suivante, pas leurs alternatives.

7.4 Expérimentation

Dans cette section nous comparons les algorithmes TD-NAC et KNAC sur le problème de pendule inversé déjà considéré. Nous en rappelons le principe. Cette tâche nécessite de balancer un pendule de longueur et masse inconnues de façon à ce qu'il reste vertical en appliquant des forces au chariot sur lequel il est fixé. Trois actions sont possibles : pousser à gauche (-1), pousser à droite ($+1$), ou ne rien faire (0). L'état du système est donné par la position angulaire φ et la vitesse angulaire $\dot{\varphi}$. Les transitions déterministes sont calculées grâce à la dynamique du système physique :

$$\ddot{\varphi} = \frac{g\sin(\varphi) - \beta ml\dot{\varphi}^2 \sin(2\varphi)/2 - 50\beta\cos(\varphi)a}{4l/3 - \beta ml\cos^2(\varphi)} \tag{7.52}$$

où g est la constante de gravitation, m et l sont la masse et la longueur du pendule, M la masse du chariot et $\beta = \frac{1}{m+M}$. L'épisode se termine dès que la position angulaire quitte l'intervalle $[-\frac{\pi}{2}, \frac{\pi}{2}]$. La récompense instantanée est le cosinus de la position angulaire vers laquelle transite le pendule : $r_i = \cos(\varphi_{i+1})$.

Nous choisissons comme paramétrisation pour la politique une distribution de Gibbs. Soit $\phi(s,a) = (\phi_i(s,a))_{1\leq i\leq p}$ le vecteur composé des fonctions de bases, la politique est donnée par :

$$\pi_\omega(a|s) = \frac{\exp\left(\phi(s,a)^T\omega\right)}{\sum_{b\in A}\exp\left(\phi(s,b)^T\omega\right)} \tag{7.53}$$

La condition de compatibilité se traduit alors par :

$$\begin{aligned}\nabla_\theta f_\theta(s,a) &= \nabla_\omega \ln(\pi_\omega(a|s))\\ &= \phi(s,a) - \sum_{b\in A}\pi_\omega(b|s)\phi(s,b)\end{aligned} \tag{7.54}$$

Nous choisissons comme fonctions de base un terme constant et un ensemble de 9 noyaux gaussiens équi-répartis (centrés en $\{-\frac{\pi}{4}, 0, \frac{\pi}{4}\} \times \{-1, 0, 1\}$ et d'écart-type 1), cela pour chaque action. Il y a donc 30 fonctions de base. Pour la fonction de valeur nous choisissons une paramétrisation linéaire (rappelons cependant que ce n'est pas une obligation, la représentation de $\hat{V}_\xi$ peut être choisie non-linéaire) :

$$\hat{V}_\xi(s) = \psi(s)^T\xi \tag{7.55}$$

Comme fonctions de base pour $\psi(s) = (\psi_i(s))_{1 \leq i \leq q}$ nous choisissons également un terme constant et un ensemble de 9 noyaux gaussiens équi-répartis (centrés en $\{-\frac{\pi}{4}, 0, \frac{\pi}{4}\} \times \{-1, 0, 1\}$ et d'écart-type 1), pour un total de 10 fonctions de base pour l'approximation de la fonction de valeur.

Restent à déterminer les paramètres pour les deux algorithmes. Ceux que nous donnons ont été choisis après quelques essais visant à obtenir les meilleurs résultats possibles. En testant plus systématiquement tous les jeux de paramètres possibles, de meilleurs résultats pourraient éventuellement être obtenus pour l'un ou l'autre des algorithmes. Cependant, les ordres de grandeur sont corrects. Pour TD-NAC et KNAC les vecteurs de paramètres initiaux sont choisis nuls. Pour TD-NAC, le taux d'apprentissage du critique est choisi égal à :

$$\alpha_i = \alpha_0 \frac{\alpha_c}{\alpha_c + i^{\frac{2}{3}}} \tag{7.56}$$

avec $\alpha_0 = 10^{-2}$ et $\alpha_c = 10^4$. Pour l'acteur, le taux d'apprentissage est choisi égal à :

$$\beta_i = \beta_0 \frac{\beta_c}{\beta_c + i} \tag{7.57}$$

avec $\beta_0 = 10^{-3}$ et $\beta_c = 10^4$. Le critique converge donc plus vite que l'acteur. En ce qui concerne KNAC, les mêmes paramètres sont choisis pour l'acteur. Pour le critique, le facteur d'éligibilité $\lambda = 1$ est choisi (ce qui correspond au KTD classique, le problème étant déterministe), l'*a priori* est choisi tel que $P_{0|0} = I$. La variance des résidus est $\sigma_i^2 = 10^{-1}$. Pour le bruit de process, nous choisissons le même bruit qu'habituellement. Si l'on note P_{θ_i, ξ_i} le bloc supérieur gauche de taille $(p+q) \times (p+q)$ de la matrice $P_{i|i}$ (la partie de la matrice de variance correspondant aux paramètres des fonctions de valeur et avantage), le bruit est choisi tel que :

$$P_{v_i} = \eta P_{\theta_{i-1}, \xi_{i-1}} \tag{7.58}$$

Le facteur d'oubli est choisi ici tel que $\eta = 10^{-6}$.

Nous comparons la capacité des deux algorithmes à apprendre la politique optimale. Pour chaque épisode, l'agent est initialisé dans un état aléatoire proche de l'équilibre $(0, 0)$. Nous mesurons la performance avec le nombre de pas de temps où le pendule est maintenu dans la zone admissible en fonction du nombre d'épisodes où l'apprentissage a eu lieu. Un maximum de 3000 pas est autorisé, ce qui correspond à un balancement du pendule de 5 minutes. Les résultats présentés sur la figure 7.2 sont moyennés sur 100 essais indépendants.

Il apparaît clairement que KNAC permet un apprentissage beaucoup plus rapide que TD-NAC. Le premier algorithme nécessite une centaine d'épisodes pour apprendre une politique optimale (avec une variance sur les résultats nulle, ce qui mérite d'être noté), contre environ 1600 pour TD-NAC. Ceci était prévisible, dans la mesure où KNAC est basé sur un critique du second ordre. Cependant, ces résultats sont également dus au fait que KTD traque la solution plutôt que de chercher la convergence, ce qui permet de s'affranchir des deux échelles de temps nécessaires pour les approches basées sur TD [72, 14].

7.5 Bilan

Dans ce chapitre, nous avons discuté les inconvénients des méthodes acteur pur et critique pur, ce qui a motivé une extension du cadre des différences temporelles de Kalman (qui entre dans la première catégorie) à une architecture acteur-critique. Nous avons présenté des concepts importants pour la combinaison de ce type d'architectures à un schéma

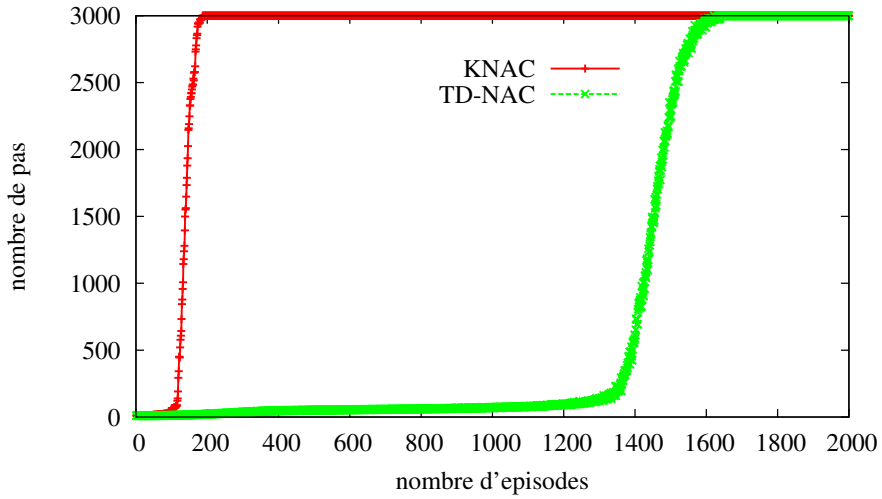


FIG. 7.2 – Pendule inversé, comparaison d'algorithmes acteur-critique.

d'approximation, à savoir la notion de compatibilité de la paramétrisation entre l'acteur et le critique et le théorème gradient-politique. Nous avons également présenté la notion de gradient naturel, basé sur la matrice d'information de Fisher et plus efficace qu'un gradient classique. Ce socle nous a servi à introduire un nouvel algorithme, KNAC, dont l'acteur est basé sur une montée de gradient naturel et le critique sur une variante de KTD(λ), permettant d'estimer conjointement la fonction avantage et la fonction de valeur. Cet algorithme est en ligne et d'ordre 2 tant pour l'acteur que pour le critique. De plus, nous rappelons qu'en vertu de ce qui a été discuté au chapitre 3.6, le critique peut également être vu comme basé sur un gradient naturel. Notons que KNAC n'est pas le seul algorithme avec un acteur basé sur un gradient naturel et un critique d'ordre 2. Il existe un certain nombre d'approches utilisant une variante de LSTD pour le critique, comme [83, 81]. Cependant, comme LSTD converge vers la solution plutôt que de traquer cette dernière (mauvaise prise en compte de la non-stationnarité, comme nous l'avons déjà souvent discuté), ils sont hors-ligne (les mises à jour n'interviennent pas après chaque interaction entre l'agent et l'environnement, mais après des trajectoires complètes). L'algorithme proposé a été expérimenté sur un problème de pendule inversé et comparé à TD-NAC, dont le critique est d'ordre 1. Il montre un apprentissage rapide de la politique optimale. Ces résultats, bien qu'insuffisants pour conclure à l'avantage de cette technique sur d'autres, sont encourageants et ouvrent d'intéressantes perspectives quant à l'utilisation du cadre de travail des différences temporelles de Kalman dans un contexte acteur-critique. D'une part, KTD peut s'avérer bénéfique aux méthodes acteur-critique de par sa nature du second ordre et sa capacité à traquer la fonction de valeur plutôt qu'à converger vers elle (et fournir une alternative à LSTD si l'on souhaite un apprentissage en ligne, voire une paramétrisation non-linéaire de la fonction de valeur). D'autre part, les méthodes acteur-critique peuvent apporter des bénéfices à KTD, grâce à leur capacité à gérer implicitement le dilemme entre exploration et exploitation ainsi qu'à leur capacité à s'affranchir du problème du calcul du maximum, ce qui est primordial pour les grands espaces d'action.

Chapitre 8

Gestion de flux de gaz dans un complexe sidérurgique

Nous rappelons que l'objectif de la thèse est pour ArcelorMittal de développer un outil d'optimisation le plus générique possible, à l'origine aucune application industrielle concrète n'avait encore été choisie. Plusieurs ont été envisagées ensuite, notamment une concernant la gestion d'un parc à cylindres au niveau du laminage à froid, pour laquelle certains résultats ont été obtenus. Cependant, pour diverses raisons (notamment un échange plus simple avec les équipes en charge du problème), une autre application lui a été préférée, à savoir la gestion des flux de gaz dans un complexe sidérurgique. C'est cette application, décrite section 8.1, que nous présentons dans ce manuscrit. En interaction avec une équipe travaillant sur le même sujet, nous avons mis au point un simulateur, dont le modèle est décrit section 8.2. Notons que si le paradigme général de l'apprentissage par renforcement est très générique, la transcription d'un problème réel en un problème d'apprentissage par renforcement n'est généralement pas triviale. En effet, il faut choisir ce que sera l'état (certaines composantes pouvant être cachées, ce qui risque de briser le caractère markovien), ce que seront les actions, simplifier ce qui peut l'être (par exemple supprimer une redondance éventuelle dans la description du système, ou faire certaines hypothèses simplificatrices), *et caetera*. Cette phase n'est pas forcément évidente, mais elle est nécessaire à l'exploitation des outils que nous proposons. Ce chapitre illustre cet état de fait. Nous comparons sur ce simulateur les résultats obtenus par KTD à ceux obtenus par une méthode de Monte Carlo, maximisant la récompense immédiate, qui correspond à ce qui est fait en pratique, aux heuristiques près. En effet, les outils d'optimisation utilisés sont statiques et les effets "long terme" sont pris en compte par des heuristiques. Ceci sera développé dans la section 8.3.

8.1 Description du problème

Le problème de la gestion de flux des gaz consiste à optimiser le flux de gaz entre des installations productrices (aciérie, cokerie...) et des installations consommatrices (four de recuit, usine de production d'électricité...) de gaz dans un complexe sidérurgique. Le routage devrait être fait de telle façon que chaque consommateur reçoive la quantité de gaz requise au bon moment. Si la production est trop basse, il peut être nécessaire d'en acheter à un réseau extérieur. Si elle est trop importante, il peut être nécessaire de torcher (brûler) le gaz (à certains endroits, cet excès peut également servir à produire de l'électricité). Le contrôle du réseau implique le routage de flux de gaz provenant d'une ou plusieurs sources

vers une ou plusieurs destinations. Il y a également des accumulateurs de gaz (*tankers*) de capacité limitée qui peuvent être remplis ou vidés lorsque c'est nécessaire (stockage du gaz excédentaire en prévision d'une pénurie, par exemple). Il existe certaines contraintes du réseau qui doivent être vérifiées. Par exemple, certains tuyaux ont des contraintes (de flux) inférieure (flux minimal) et supérieure (flux maximal) et certains consommateurs ont des contraintes de capacité calorifique minimale (voir maximale) et de puissance pour le gaz les alimentant. Cela est important étant donné que les gaz provenant de différentes sources n'ont pas forcément les mêmes capacités calorifiques.

Les caractéristiques importantes pour traiter ce problème de gestion des flux de gaz peuvent être résumées comme suit :

contraintes du problème d'optimisation :
- infrastructure du réseau (position des mélangeurs et des switchs, tuyaux entre les différentes installations) et contraintes associées (limites de flux, capacités calorifiques minimales...),
- quantité de gaz produite par chaque installation et capacité calorifique associée,
- gaz consommé par chaque installation (puissance requise et capacité calorifiques minimale associée) ;

degrés de liberté pour l'optimisation :
- configuration du réseau (état des diviseurs, utilisation des accumulateurs, *et caetera*, mais on s'interdit d'en modifier la topologie) ;

fonction de coût :
- prix de la quantité de gaz achetée minorée du prix de la quantité de gaz revendue après transformation en électricité ;

autres informations :
- différentes mesures dans le réseau (pressions, flux, capacités calorifiques...).

Ce problème d'optimisation de la gestion des flux de gaz est traité par une autre équipe d'ArcelorMittal qui utilise un logiciel nommé `Aspen+` pour optimiser le réseau. Le moteur d'optimisation associé à ce programme permet de chercher une solution à la minimisation de la fonction de coût sous les contraintes du réseau. La solution obtenue a été comparée au contrôle des flux effectué par les opérateurs et des améliorations significatives peuvent être obtenues par rapport aux heuristiques employées en pratique. Cependant, `Aspen+` n'est pas tout à fait adapté à ce problème. Les résultats obtenus sont parfois plus mauvais que la solution de l'opérateur (minimum local et corruption des données peuvent être des explications). Ce logiciel ne permet pas de prendre en compte des switchs du réseau, seulement les mélangeurs et diviseurs continus. Mais surtout, l'optimisation dans le cadre d'`Aspen+` est statique. D'une part, cela ne permet pas de prendre en compte les accumulateurs de gaz. D'autre part, les opérateurs effectuent un contrôle dynamique des flux, la comparaison n'est pas totalement équitable. L'apprentissage par renforcement, quant à lui, peut prendre naturellement en compte cette composante dynamique du problème.

Etant donné le problème à traiter, il a été jugé nécessaire d'implémenter les structures suivantes :

structure du réseau implémentant différentes caractéristiques :
- producteurs et consommateurs,
- contraintes du réseau,
- torches et connexions au réseau de gaz extérieur,
- différents nœuds (mélangeurs, diviseurs, accumulateurs...),

– degrés de liberté de l'utilisateur ;

réponse du réseau (gaz torché et acheté) à un ensemble de contraintes (flux, puissance consommée) et à la configuration choisie par l'utilisateur ;

l'algorithme d'optimisation est un agent interagissant avec le système :
– une action (une configuration du réseau) est choisie,
– la réponse du réseau est observée et une récompense est récupérée,
– suivant la réponse du système, une nouvelle action est choisie ;

Ces différentes étapes ne sont pas spécifiques au problème de la gestion des flux, à part la forme particulière des états, actions et récompenses.

Notons que la modélisation d'un tel environnement peut être un problème très complexe, ce qui justifie le prix important de logiciels comme `Aspen+`. Nous allons en proposer une modélisation relativement simple, afin que sa conception et son implémentation soient compatibles avec les contraintes de programme et scientifiques de la thèse, mais qui met tout de même en avant la problématique de la gestion des accumulateurs (ce qui pose problème pour les solutions actuelles).

8.2 Modélisation du problème

Le type de réseau que nous considérons possède des contraintes topologiques et fonctionnelles. Les contraintes topologiques peuvent être par exemple un flux maximal admissible dans un certain tuyau du réseau et de façon plus générale l'architecture du réseau. Les contraintes fonctionnelles sont par exemple la puissance et la capacité calorifique minimale requises par certains consommateurs.

Une première solution, assez intuitive, serait de chercher une modélisation qui permette la propagation et la prise en compte des contraintes du réseau. Cependant, ce type d'approche est difficile de façon conceptuelle (autrement dit formalisation de ce type de problème) et à implémenter. Nous proposons une modélisation plus simple, bien que moins fondée physiquement. Nous considérons le réseau comme un ensemble de nœuds reliés par des tuyaux. Dans la modélisation adoptée, il est possible à chaque nœud d'acheter et de torcher du gaz, ce qui permet de respecter les différentes contraintes du réseau. Soit il existe vraiment un accès au réseau extérieur (de gaz) et/ou une torche et la modélisation est correcte. Soit il n'est possible physiquement ni d'acheter ni de torcher du gaz et nous les considérerons comme virtuels. On achète et on torche du gaz "virtuel" pour respecter les contraintes de réseau, le caractère virtuel se traduit par une pénalisation de la fonction de coût (le coût étant compris ici comme étant l'opposé de la récompense).

8.2.1 Nœuds élémentaires

Une première étape dans la description du réseau est la définition des nœuds élémentaires qui le composent. De façon générale, nous noterons f les flux (sauf le flux de gaz naturel, acheté sur le réseau extérieur, que nous notons G), c les capacités calorifiques et P les puissances. Pour chaque type de nœud nous décrivons son principe avant de le résumer dans une définition.

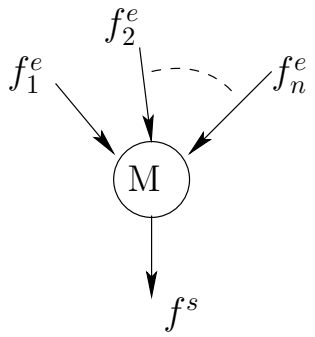


FIG. 8.1 – Nœud mélangeur.

Nœud mélangeur

Le premier type de nœud que nous considérons est le nœud mélangeur, illustré figure 8.1 (remarquons que lorsqu'on nous notons un flux, il y a toujours une concentration associée). Il a n entrées, une seule sortie et il n'y a pas de degré de liberté. En effet, comme le flux ne peut *a priori* être ni torché ni routé ailleurs, on devrait avoir conservation des flux. On vérifie donc le lien suivant entre les flux d'entrée et le flux de sortie (les flux et les puissances sont additifs, une puissance étant le produit d'un flux par une capacité calorifique) :

$$\begin{cases} f^s = \sum_{i=1}^{n} f_i^e \\ c^s = \frac{1}{f^s} \sum_{i=1}^{n} c_i^e f_i^e \end{cases} \tag{8.1}$$

Les contraintes de réseau peuvent être violées dans deux cas, si la somme des flux d'entrée est inférieure au flux minimal de sortie $f^s_{\min}$, ou si elle est supérieure au flux maximal de sortie $f^s_{\max}$.

Dans le premier cas, c'est-à-dire si l'on vérifie $f^s < f^s_{\min}$, il faut acheter le flux de gaz manquant $G_n = f^s_{\min} - f^s$. Dans le second cas, c'est-à-dire si $f^s > f^s_{\max}$, il faut torcher le flux de gaz excédentaire $f_T = f^s - f^s_{\max}$. Nous rappelons que la torche et la possibilité d'acheter du gaz peuvent être virtuels, dans ce cas ils sont pénalisés dans la fonction de coût. Nous y reviendrons en définissant la fonction de récompense.

Ce type de nœud a juste fonction de routage, il ne correspond ni a un degré de liberté ni à une composante d'état du système (cependant il contribue à la fonction de récompense). Pour la définition suivante, nous rappelons que la notation $(.)_+$ correspond au max entre la valeur et 0 : $(x)_+ = \max(x, 0)$.

Définition 8.1 (Nœud mélangeur). *Un nœud mélangeur (représenté figure 8.1) comporte n entrées et 1 sortie. Les flux et capacité calorifique de sortie se calculent en fonction des flux et capacités calorifiques d'entrée :*

$$\begin{cases} f^s = \max\left(\min\left(\sum_{i=1}^{n} f_i^e, f^s_{max}\right), f^s_{min}\right) \\ c^s = \frac{1}{\sum_{i=1}^{n} f_i^e} \sum_{i=1}^{n} c_i^e f_i^e \end{cases} \tag{8.2}$$

Les flux acheté au réseau extérieur et torché sont respectivement :

$$\begin{cases} G_n = \left(f^s_{min} - f^s\right)_+ \\ f_T = \left(f^s - f^s_{max}\right)_+ \end{cases} \tag{8.3}$$

Nœud consommateur

FIG. 8.2 – Nœud consommateur.

Un nœud consommateur a une entrée et pas de sortie, comme illustré sur la figure 8.2. Notons qu'on peut facilement considérer un nœud consommateur à n entrées en le couplant avec un nœud mélangeur. Le nœud va consommer une puissance $P = f^e c^e$. Nous distinguons deux types de consommateurs, les libres et les contraints.

Consommateur libre : les consommateurs libres n'ont pas de contraintes de puissance consommée, comme par exemple l'usine de production d'électricité. Ils ne présentent aucun degré de liberté et ne participent pas à la description d'état. Cependant, ils contribuent au coût total (par exemple pour l'usine de production d'électricité le gaz est transformé en électricité qui est vendue).

Consommateur contraint : Les consommateurs contraints (par exemple cokerie ou aciérie) ont des contraintes de puissance consommée de type

$$\begin{cases} P = fc \\ c \geq c_{\min} \end{cases} \tag{8.4}$$

qui ne sont pas forcément vérifiées dans ce formalisme, étant donné que nous ne modélisons pas les propagations de contraintes. Si le flux et la capacité calorifique d'entrée ne vérifient pas les contraintes, il faut acheter et/ou torcher une certaine quantité de gaz. Les calculs *ad hoc* sont développés ci-après. Ils déterminent la quantité *minimale* de gaz à acheter pour vérifier les contraintes. Il n'y a pas de degré de liberté et un tel consommateur correspond à deux composantes d'état (puissance et capacité calorifique requises).

Le calcul des flux de gaz à acheter et à torcher (ou flux résiduels) au niveau d'un consommateur contraint est un problème d'optimisation sous contraintes. Cependant, nous allons voir que sous une hypothèse forte mais physiquement justifiée (le gaz acheté sur le réseau extérieur est le plus énergétique), le problème peut être analytiquement résolu.

En entrée nous avons un mélange de gaz de flux f_e et concentration c_e. Le nœud, lorsqu'il est consommateur, demande de vérifier les contraintes (8.4). Une partie f_r (flux résiduel) du flux f_e défini précédemment peut être ignoré (par exemple si la capacité énergétique est trop faible) et si le flux n'est pas assez important (ou pas assez énergétique), il est possible d'acheter le flux G_{n_0} de gaz naturel (de capacité associée c_n). Le flux et la concentration fournis au consommateur sont donc

$$\begin{cases} f_e^* = f_e - f_r + G_{n_0} \\ c_e^* = \frac{1}{f_e^*} \left(c_e(f_e - f_r) + c_n G_{n_0} \right) \end{cases} \tag{8.5}$$

En injectant ces dernières équations dans les contraintes, nous obtenons

$$\begin{aligned}
&\begin{cases} P = c_e^* f_e^* \\ c_e^* \geq c_{\min} \end{cases} \\
\Leftrightarrow &\begin{cases} P = c_e(f_e - f_r) + c_n G_{n_0} \\ \frac{P}{f_e^*} \geq c_{\min} \end{cases} \\
\Leftrightarrow &\begin{cases} P = c_e(f_e - f_r) + c_n G_{n_0} \\ \frac{P}{c_{\min}} \geq f_e - f_r + G_{n_0} \end{cases}
\end{aligned} \tag{8.6}$$

Il y a encore deux contraintes, relativement évidentes mais qui méritent d'être mentionnées :

$$\begin{cases} G_{n_0} \geq 0 \\ 0 \leq f_r \leq f_e \end{cases} \tag{8.7}$$

La première signifie que le flux de gaz acheté doit être positif, la seconde que le flux résiduel doit être positif et au plus égal au flux entrant f_e. Si le système physique est bien modélisé, il devrait y avoir au moins une solution. Cependant, il y en a le plus souvent plusieurs. On pourrait alors choisir la valeur de c_e^* qui vérifie ces contraintes tout en minimisant la quantité de gaz acheté G_{n_0}. On obtient alors un problème d'optimisation sous contraintes.

Nous faisons à présent comme hypothèse que le gaz acheté sur le réseau extérieur est plus énergétique que tous les autres gaz. Autrement dit, sa capacité calorifique est plus importante :

$$\forall c, \quad c_n \geq c \tag{8.8}$$

et donc plus particulièrement :

$$c_n \geq c_e \tag{8.9}$$

Dans les raisonnements qui suivent nous allons nous placer dans $\mathbb{R}^2$, sur un graphe (f_r, G_{n_0}) (c'est-à-dire G_{n_0} fonction de f_r).

Nous obtenons donc à partir des contraintes (8.4) deux équations. La première est :

$$G_{n_0} = \frac{P - c_e f_e}{c_n} + \frac{c_e}{c_n} f_r \tag{8.10}$$

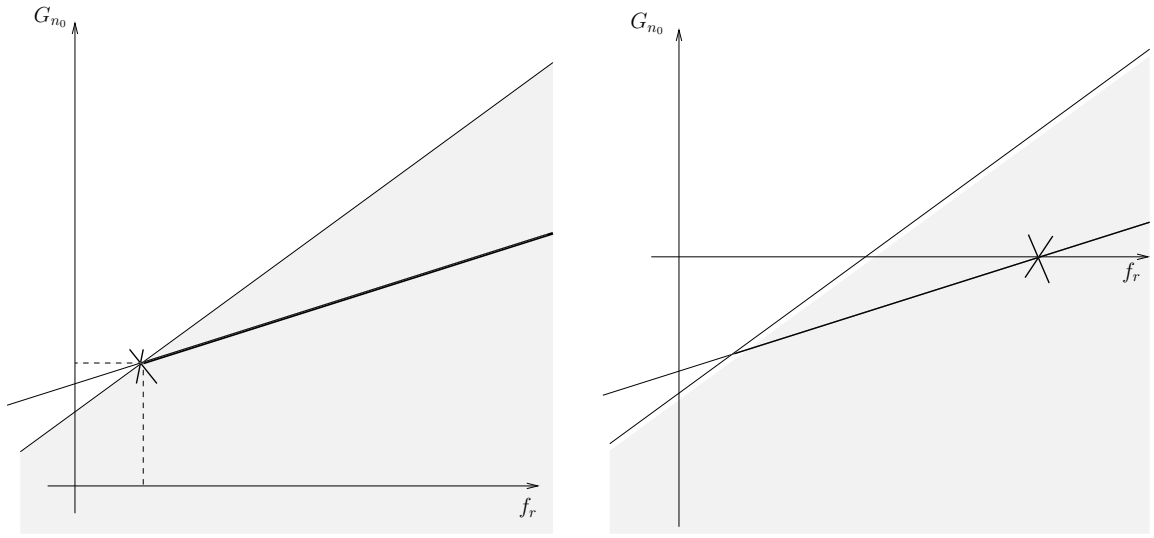
Elle définit une droite d'origine $\frac{P - c_e f_e}{c_n}$ et de pente $\frac{c_e}{c_n} < 1$ (d'après l'hypothèse que le gaz acheté sur le réseau extérieur est le plus énergétique). La seconde équation est :

$$G_{n_0} \leq \frac{P}{c_{\min}} - f_e + f_r \tag{8.11}$$

Elle définit le demi-espace inférieur dont la limite supérieure est donnée par la droite d'origine $\frac{P}{c_{\min}} - f_e$ et de pente 1.

Nous distinguons quatre cas, selon que l'intersection se fasse à flux résiduel positif ou négatif et à flux G_{n_0} positif ou négatif (autrement dit, selon le cadran où se fait l'intersection). Premièrement, si l'origine de la première droite est supérieure à celle de la seconde, c'est-à-dire si :

$$\begin{aligned}
&\frac{P - c_e f_e}{c_n} > \frac{P}{c_{\min}} - f_e \\
\Leftrightarrow\ &f_e > \frac{P}{c_{\min}} \frac{c_n - c_{\min}}{c_n - c_e}
\end{aligned} \tag{8.12}$$



a. Intersection à $G_{n_0} > 0$. b. Intersection à $G_{n_0} < 0$.

FIG. 8.3 – Intersection à $f_r > 0$

alors le flux résiduel sera positif. Cela est illustré sur la figure 8.3. L'autre cas (flux résiduel négatif, *i.e.* demi plan gauche) est illustré sur la figure 8.4.

Supposons être dans le cas de la figure 8.3.a. L'espace des solutions est l'intersection de la droite avec le demi-plan défini par la seconde équation (demi-droite dont le trait est épais sur les figures). De plus parmi ces solutions il ne faut garder que celles qui vérifient les différentes contraintes, puis enfin la solution qui minimise le flux acheté. Dans ce premier cas, les deux droites s'intersectent dans le cadrant positif et on peut calculer les flux :

$$\begin{aligned} \frac{P - c_e f_e}{c_n} + \frac{c_e}{c_n} f_r &= \frac{P}{c_{\min}} - f_e + f_r \\ \Leftrightarrow f_r &= f_e - \frac{P}{c_{\min}} \frac{c_n - c_{\min}}{c_n - c_e} \end{aligned} \tag{8.13}$$

et alors à partir de n'importe laquelle des deux équations on a :

$$G_{n_0} = \frac{P}{c_{\min}} \frac{c_{\min} - c_e}{c_n - c_e} \tag{8.14}$$

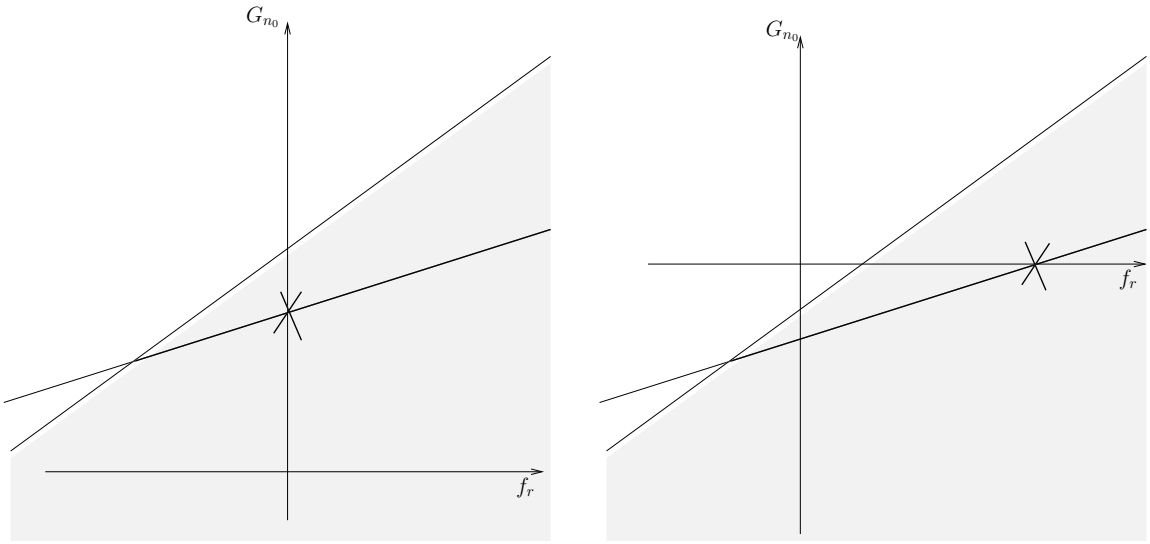
Il est clair qu'alors on vérifie $0 < f_r < f_e$.

Si par contre cette solution correspond à un flux acheté négatif, c'est-à-dire que les concentrations vérifient

$$c_{\min} < c_e \tag{8.15}$$

alors il faut considérer la solution illustrée sur la figure 8.3.b. Le flux acheté est nul, $G_{n_0} = 0$: à partir de la première équation on obtient le flux résiduel :

$$f_r = f_e - \frac{P}{c_e} \tag{8.16}$$



a. Intersection à $G_{n_0} > 0$. b. Intersection à $G_{n_0} < 0$.

FIG. 8.4 – Intersection à $f_r < 0$

Les deux autres cas correspondent à un flux résiduel négatif, ce qui advient lorsque :

$$\begin{aligned} &\frac{P - c_e f_e}{c_n} > \frac{P}{c_{\min}} - f_e \\ \Leftrightarrow\ &f_e < \frac{P}{c_{\min}} \frac{c_n - c_{\min}}{c_n - c_e} \end{aligned} \tag{8.17}$$

Le premier cas, illustré sur la figure 8.4.a, correspond à :

$$\begin{cases} f_r = 0 \\ G_{n_0} = \frac{P - c_e f_e}{c_n} \end{cases} \tag{8.18}$$

Par contre si $P < c_e f_e$ alors la solution est :

$$\begin{cases} G_{n_0} = 0 \\ f_r = f_e - \frac{P}{c_e} \end{cases} \tag{8.19}$$

comme illustré sur la figure 8.4.b.

Ces résultats sont résumés par le tableau 8.1, qui pour des contraintes (inégalités vérifiées par les paramètres du système) donne la solution. Notons qu'on vérifie bien dans tous les cas que $G_{n_0} \geq 0$ et que $0 \leq f_r \leq f_e$.

Définition 8.2 (Nœud consommateur). *Nous distinguons deux types de nœuds consommateurs (voir figure 8.2). Pour chacun, il y a un flux d'entrée f^e et la capacité calorifique associée c^e. Le premier type, dit* consommateur libre*, n'est pas contraint et ne participe pas à la description de l'état du système. Cependant, il participe à la fonction coût (un exemple type de consommateur libre est l'usine d'électricité à laquelle peut être vendu le gaz excédentaire). Le second type, dit* consommateur contraint*, doit vérifier des conditions sur la puissance consommée et la capacité calorifique du gaz considéré :*

$$\begin{cases} P = fc \\ c \geq c_{min} \end{cases} \tag{8.20}$$

Contraintes	G_{n_0}	f_r
$f_e > \frac{P}{c_{\min}} \frac{c_n - c_{\min}}{c_n - c_e}$ et $c_{\min} > c_e$	$G_{n_0} = \frac{P}{c_{\min}} \frac{c_{\min} - c_e}{c_n - c_e}$	$f_r = f_e - \frac{P}{c_{\min}} \frac{c_n - c_{\min}}{c_n - c_e}$
$f_e > \frac{P}{c_{\min}} \frac{c_n - c_{\min}}{c_n - c_e}$ et $c_{\min} < c_e$	$G_{n_0} = 0$	$f_r = f_e - \frac{P}{c_e}$
$f_e < \frac{P}{c_{\min}} \frac{c_n - c_{\min}}{c_n - c_e}$ et $P > c_e f_e$	$G_{n_0} = \frac{P - c_e f_e}{c_n}$	$f_r = 0$
$f_e < \frac{P}{c_{\min}} \frac{c_n - c_{\min}}{c_n - c_e}$ et $P < c_e f_e$	$G_{n_0} = 0$	$f_r = f_e - \frac{P}{c_e}$

TAB. 8.1 – Nœud consommateur : résumé des résultats.

Les flux acheté au réseau extérieur et torché sont donnés dans le tableau 8.1.

Nœud diviseur

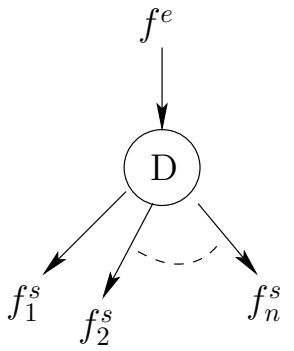


FIG. 8.5 – Nœud diviseur.

Le nœud diviseur comporte une entrée et n sorties, comme illustré figure 8.5. Il ne participe pas à la description du système (il ne contribue pas au vecteur d'état), mais il possède n degrés de liberté, un par sortie.

Chaque degré de liberté est associé à une sortie et correspond à la proportion de flux maximal du tuyau associé :

$$f_i^s = u_i(f_{i,\max}^s - f_{i,\min}^s) + f_{i,\min}^s \tag{8.21}$$

Dans cette équation, $u_i \in [0,1]$ est le degré de liberté associé à la $i^{\text{ème}}$ sortie. Cependant, il faut vérifier la loi de Kirchhoff (ce qui rentre doit être égal à ce qui sort) et donc on ajoute une torche (éventuellement virtuelle) ainsi qu'une arrivée de gaz (éventuellement virtuelle). C'est-à-dire que si $f^s = \sum_i f_i^s > f^e$, on achète le flux de gaz nécessaire $G_n = f^s - f^e$ (à un coût prohibitif si l'arrivée de gaz est virtuelle), si $f_s - f^e < 0$, on torche l'excédent $f_T = f^e - f^s$ (ce qui a également un coût si la torche est virtuelle).

Définition 8.3 (Nœud diviseur). *Un nœud diviseur (voir figure 8.5) comporte 1 entrée et n sorties. Il ne participe pas à la description de l'état du système, mais il correspond à n degrés de liberté. Soit* $u_i \in [0,1]$ *le pourcentage de flux possible que l'on choisit de faire*

passer dans le tuyau i :

$$f_i^s = u_i(f_{i,max}^s - f_{i,min}^s) + f_{i,min}^s \tag{8.22}$$

La capacité calorifique du gaz pour chaque sortie est égale à celle d'entrée :

$$c_i^s = c^e \tag{8.23}$$

Si les contraintes de conversion des flux ne sont pas vérifiées, il faut acheter ou torcher du gaz :

$$\begin{cases} G_n = (\sum_{i=1}^n f_i^s - f^e)_+ \\ f_T = (f^e - \sum_{i=1}^n f_i^s)_+ \end{cases} \tag{8.24}$$

Nœud producteur pur

FIG. 8.6 – Nœud producteur.

Les nœuds producteurs purs n'ont pas d'entrée et une seule sortie, comme illustré sur la figure 8.6. Comme pour le nœud consommateur, on peut obtenir un nœud producteur avec plusieurs sorties à partir d'un nœud producteur couplé à un diviseur. Un producteur (haut-fourneau par exemple) produit du gaz à flux et concentration f_0 et c_0 fixés. Ce flux et cette concentration font partie de la description de l'état du système, mais il n'y a pas de degré de liberté (donc pas de composante d'action). On peut noter que certaines installations sont à la fois productrices et consommatrices. Il suffit de les modéliser par l'ensemble d'un consommateur et d'un producteur, qui sont deux nœuds élémentaires.

Définition 8.4 (Nœud producteur). *Un nœud producteur (voir figure 8.6) pur a une sortie. Il produit du gaz à flux et capacité calorifique f_0 et c_0. Il participe donc à la description de l'état du système.*

Nœud accumulateur

Le dernier type de nœud que l'on considère est l'accumulateur de gaz, qui comporte une entrée et une sortie, comme illustré sur la figure 8.7. L'accumulateur dispose de son propre flux f^a de capacité c^a. Ces deux quantités participent à la description de l'état du système. Notons qu'en toute rigueur, il faudrait considérer le volume de gaz contenu dans l'accumulateur et non le flux. Ces deux quantités diffèrent par une unité de temps. Dans le mesure où l'on considère un simulateur à événements discrets (et équi-répartis sur l'axe temporel), choisir l'une ou l'autre revient au même.

Le fonctionnement que nous choisissons est le suivant. Le gaz d'entrée sert à alimenter l'accumulateur et une portion du flux propre de l'accumulateur (plus le gaz d'entrée) sert à

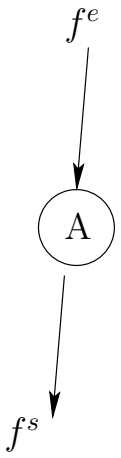


FIG. 8.7 – Nœud accumulateur.

alimenter la sortie. Soit f^e le flux d'entrée. Le degré de liberté est la proportion $u_a \in [0,1]$ du flux f^e+f^a qui va en sortie alimenter le reste du réseau. On obtient alors un nouveau flux $(1-u_a)(f^e+f^a)$ dans l'accumulateur qui doit vérifier la contrainte de capacité maximale $f^a_{\max}$ (le flux minimal est considéré nul, ce qui correspond à un accumulateur vide).

Ainsi, le nouveau flux de l'accumulateur et la capacité calorifique associée sont :

$$\begin{cases} f^a_{\text{new}} = \min(f^a_{\max}, (1-u_a)(f_e+f_A)) \\ c^a_{\text{new}} = \frac{f^e c^e + f^a c^a}{f^e + f^a} \end{cases} \tag{8.25}$$

En conséquence, le flux à torcher est le flux excédentaire (s'il y en a) :

$$f_T = ((1-u_a)(f_e+f_A) - f^a_{\max})_+ \tag{8.26}$$

En sortie, nous avons :

$$\begin{cases} f^s = u_a(f^e+f^a) \\ c^s = \frac{f^e c^e + f^a c^a}{f^e + f^a} \end{cases} \tag{8.27}$$

Ce type de nœud présente deux composantes d'état (flux f^a et capacité calorifique c^a de l'accumulateur), ainsi qu'un degré de liberté (la proportion u_a du flux f^e+f^a qui sert à remplir l'accumulateur).

Définition 8.5 (Nœud accumulateur). *Un nœud accumulateur (voir figure 8.7) dispose d'une entrée et d'une sortie. La proportion $u_a \in [0,1]$ de la somme des flux d'entrée et propre à l'accumulateur, f^e+f^a, sert à alimenter la sortie. Le flux de gaz à torcher est :*

$$\left\{ f_T = ((1-u_a)(f_e+f_A) - f^a_{max})_+ \right. \tag{8.28}$$

Le flux de l'accumulateur et la capacité calorifique associée sont modifiés de la façon suivante :

$$\begin{cases} f^a_{new} = \min(f^a_{max}, (1-u_a)(f_e+f_A)) \\ c^a_{new} = \frac{f^e c^e + f^a c^a}{f^e + f^a} \end{cases} \tag{8.29}$$

Le flux de sortie et la capacité calorifique associée sont calculés de la façon suivante :

$$\begin{cases} f^s = u_a(f^e+f^a) \\ c^s = \frac{f^e c^e + f^a c^a}{f^e + f^a} \end{cases} \tag{8.30}$$

8.2.2 Modélisation du réseau

Ainsi nous avons introduit un certain nombre de nœuds élémentaires, qui devraient permettre de modéliser toutes sortes de réseaux. Par exemple une usine qui est à la fois productrice et consommatrice et qui a plusieurs entrées et plusieurs sorties peut être modélisée par un diviseur, un mélangeur, un consommateur et un producteur. La modélisation proposée est ainsi très générale, mais elle ne permet pas de respecter les contraintes topologiques du réseau réel car on permet d'acheter et de torcher du gaz à tous les nœuds. Nous rappelons que c'est un choix qui permet de simplifier les calculs de propagation des contraintes, qui sont intégrées dans la fonction de récompense, ce qui permettra éventuellement de trouver une solution qui correspond au vrai réseau (sans violation des structures topologiques).

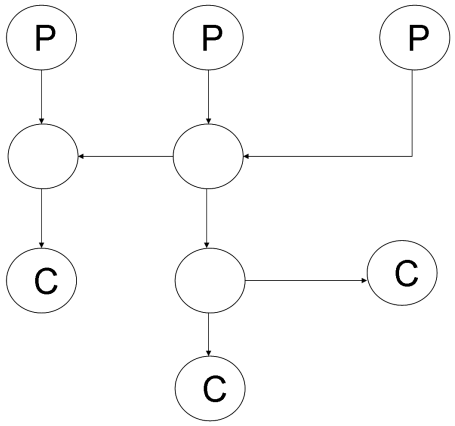


FIG. 8.8 – Structure d'arbre pour le calcul du réseau.

Reste à voir le fonctionnement du réseau dans son ensemble. En effet, le flux d'entrée d'un nœud est nécessaire au calcul du flux de sortie, lui-même nécessaire au calcul du nœud suivant. Il faut commencer par calculer les flux de sortie des producteurs purs, pour lesquels il n'existe pas d'entrée. Il faut ensuite propager ces flux, la sortie d'un producteur étant l'entrée d'un autre nœud, jusqu'à arriver aux consommateurs purs, pour lesquels il n'y a pas de sortie. Pour pouvoir calculer ces flux simplement, il faut organiser le réseau en structure d'arbre, comme illustré figure 8.8, avec aux racines les producteurs purs et aux feuilles les consommateurs purs. Certaines installations sont à la fois productrice et consommatrice, dans ce cas il faut les séparer en deux nœuds, un correspondant à la consommation et l'autre à la production. Une fois la structure d'arbre obtenue, il faut calculer l'ensemble des nœuds de profondeur zéro, puis ceux de profondeur un, jusqu'à la dernière feuille. Lorsqu'on calcule les nœuds de profondeur i, les nœuds de profondeur $i-1$ auront déjà été calculés, les flux d'entrée seront donc connus et le nœud est calculable.

Une question pertinente est de savoir s'il est toujours possible de mettre le réseau sous forme d'arbre (fini). Il est envisageable que ce soit impossible, à cause par exemple d'une structure de boucle. Si le réseau ne peut se mettre que sous la forme de graphe (et pas d'arbre), les calculs de réseaux sont plus complexes. Cependant, il y a une réalité physique derrière, qui conditionne la réponse à cette question. Nous verrons que ce problème ne se pose pas pour le réseau que nous proposons d'étudier dans la suite.

8.2.3 Modélisation conforme à l'apprentissage par renforcement

Nous avons ainsi présenté une modélisation possible du problème de la gestion des flux de gaz. Il reste encore à modéliser ce problème en termes adéquats au formalisme des processus décisionnels de Markov, c'est-à-dire spécifier ce qui doit être état, action et récompense.

Etats : l'état du système est composé :
- des contraintes de puissance et de capacité calorifique minimale des consommateurs contraints,
- du flux produit par les producteurs et de la capacité calorifique associée,
- du flux et de la capacité calorifique courants des accumulateurs.

L'ensemble des contributions des nœuds au vecteur d'état est résumé par le tableau 8.2. Si on note N_C le nombre de consommateurs contraints, N_P le nombre de producteurs, et N_A le nombre d'accumulateurs, alors on a $(2N_c + 2N_P + 2N_A)$ composantes d'état (chaque composante étant continue) ;

Actions : l'ensemble des actions du système est composé :
- des degrés de liberté au niveau des diviseurs,
- des degrés de liberté au niveau des accumulateurs.

L'ensemble des contributions des nœuds au vecteur d'action est résumé dans le tableau 8.2. Si l'on note N_S le nombre total de sorties (en cumulant tous les diviseurs), on a $(N_S + N_A)$ composantes continues ;

Récompenses : la récompense est égale au prix de l'électricité vendue (pour la partie du gaz transformé en électricité) minoré du prix de la quantité de gaz acheté. De plus, comme nous l'avons déjà souligné, il faut pénaliser l'achat et le torchage de gaz là où ce n'est pas possible physiquement. On a donc une fonction de récompense de la forme :

$$r(s,a) = \sum_{i=1}^{N} \left(\alpha_T^{(i)} f_T^{(i)} + \alpha_G^{(i)} G_n^{(i)} \right) \tag{8.31}$$

Dans cette expression, N est le nombre total de nœuds et les différents termes ont les significations suivantes (ici $f_T^{(i)}$ est la quantité de gaz torché au nœud i et $G_n^{(i)}$ est le flux de gaz acheté pour ce même nœud) :
- si le nœud i dispose d'une torche, alors $\alpha_T^{(i)} = 0$. S'il ne dispose pas de torche et ne permet pas de produire de l'électricité, alors c'est un terme pénalisant (violation de la contrainte topologique) et $\alpha_T^{(i)} < 0$. Enfin, si le nœud produit de l'électricité, c'est le prix de revient par watt produit (par exemple) et $\alpha_T^{(i)} > 0$. Notons que comme c'est la puissance qui est vendue et non le flux, ce coefficient devrait dépendre de la capacité calorifique. Nous choisissons donc dans ce cas $\alpha_T^{(i)} = \frac{c}{c_n}\alpha_{\text{PP}}$, où α_{PP} est une constante et c la capacité calorifique du gaz vendu ;
- si le nœud dispose d'un accès sur le réseau extérieur, alors $\alpha_G^{(i)}$ est le négatif du prix d'achat (par unité de flux) du gaz au réseau extérieur. Si le nœud ne dispose pas d'accès au réseau extérieur, il y a violation de la topologie du réseau, et $\alpha_G^{(i)}$ est une pénalisation (très) supérieure au prix d'achat du gaz.

Notons que pour obtenir une solution qui satisfasse les contraintes topologiques du réseau, il faut bien dimensionner ces différentes grandeurs (ce que nous ferons de façon empirique, notamment pour les coûts des violations).

Probabilités de transition : les algorithmes développés dans le cadre de la thèse sont *model-free*, ils ne nécessitent pas un modèle de transition. Cependant, il est important de noter que s'il y a indépendance de la transition à l'action, c'est-à-dire

$$p(s'|s,a) = p(s'|s)$$

alors maximiser le cumul de récompense sur le long terme revient rigoureusement à maximiser la récompense immédiate. Ce n'est plus un problème d'apprentissage par renforcement. Si on applique les algorithmes développés, cela va revenir à régresser la fonction de récompense et à en chercher le maximum. On arriverait au même résultat avec du Monte Carlo par exemple (et de façon plus simple, étant donné qu'on dispose d'un modèle de simulation). Cela concerne uniquement le cas statique, sans accumulateur. Dans le cas dynamique, avec accumulateur, au moins les composantes d'état correspondant au contenu des accumulateurs dépendent de l'action choisie. L'évolution des grandeurs d'état relatives à l'accumulateur sont données dans la définition 8.5. Pour les autres composantes d'état du système, nous avons choisi des modèles d'évolution temporelle en accord avec ArcelorMittal, de façon à obtenir quelque chose de proche du vrai système. Nous les détaillons par la suite, lorsque nous décrivons plus précisément le réseau traité.

Il faut noter que si cette modélisation est plus simple en terme de propagation des contraintes à travers le réseau, elle est plus complexe en terme d'espace état-action. En effet, certaines actions qui seraient impossibles avec une modélisation plus fine (mais bien plus complexe à mettre en œuvre) sont simplement pénalisées ici.

nœud	composantes d'état	composantes d'action
mélangeur	x	x
consommateur libre	x	x
consommateur contraint	(P, c_{min})	x
diviseur	x	$(u_1, \ldots, u_n)$
producteur	(f_0, c_0)	x
accumulateur	(f^a, c^a)	u^a

TAB. 8.2 – Contribution des nœuds en terme d'AR.

8.3 Application de KTD

Dans cette section, nous présentons l'application de KTD à la gestion des flux de gaz sur un réseau particulier. Nous commençons par décrire ce réseau et donnons également plus de précisions sur l'évolution des états choisie. Pour traiter ce problème, nous introduisons d'abord une variation de KTD : nous proposons une paramétrisation semi-automatique de la fonction de qualité basée sur l'utilisation de noyaux et d'une méthode de dictionnaire [29]. Cela peut-être vu comme une adaptation de la méthode qui permet de rendre la représentation non-paramétrique de la fonction de valeur parcimonieuse dans le cadre de GPTD [28] au cadre de KTD (la seule différence étant que la représentation parcimonieuse se construit en ligne pour GPTD, alors qu'elle doit être construite en pré-traitement pour KTD). Nous présentons enfin les résultats de l'application de KTD à l'optimisation de ce réseau. Etant

donné que les méthodes d'optimisation utilisées en pratique sont statiques (voire combinées à des heuristiques pour prendre en compte les accumulateurs), nous comparons notre approche à une méthode de Monte Carlo maximisant la récompense immédiate.

8.3.1 Un réseau particulier

Le réseau que nous proposons d'étudier peut sembler relativement simple. Cependant, il est intéressant pour plusieurs raisons. Tout d'abord, il représente d'un méta-point de vue assez fidèlement le type de réseaux que l'on peut rencontrer dans la pratique (les nœuds de production ou de consommation correspondants en fait à des sous-réseaux). C'est d'ailleurs également ce type de réseau qui est considéré par l'autre équipe travaillant sur le sujet. D'autre part, comme nous le verrons, il permet de mettre en avant la gestion de l'accumulateur de gaz, qui est le point qui pose problème pour les approches utilisées en pratique.

Description du réseau

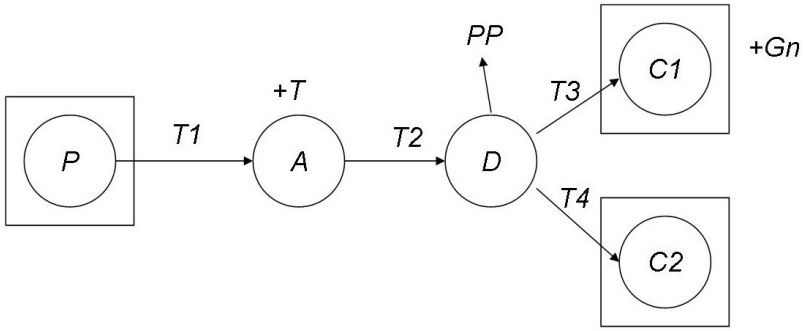


FIG. 8.9 – Réseau considéré.

Nous considérons le réseau illustré sur la figure 8.9, qui est composé :

- d'un producteur P ;
- d'un accumulateur A qui est relié à une torche ;
- d'un diviseur D qui alimente deux consommateurs et dont le gaz torché alimente en fait une usine d'électricité ;
- d'un consommateur C_1 qui est connecté au réseau de gaz naturel ;
- d'un consommateur C_2 qui n'est pas connecté au réseau de gaz naturel.

D'après le parallèle avec l'apprentissage de la section 8.2.3, le producteur correspond à 2 composantes d'état, chaque consommateur également, le diviseur à deux composantes d'action et l'accumulateur à deux composantes d'état et une d'action. Finalement, on obtient un vecteur d'état de dimension 8 et un vecteur d'action de dimension 3. Pour résumer,

nous avons comme vecteurs d'état et d'action :

$$\begin{aligned} s &= [\underbrace{s_1, s_2}_{P}, \underbrace{s_3, s_4}_{A}, \underbrace{s_5, s_6}_{C_1}, \underbrace{s_7, s_8}_{C_2}]^T \in \mathbb{R}^8 \\ &= [\underbrace{f_0, c_0}_{P}, \underbrace{f^a, c^a}_{A}, \underbrace{P_1, c_{1,\min}}_{C_1}, \underbrace{P_2, c_{2,\min}}_{C_2}]^T \qquad (8.32) \\ a &= [\underbrace{a_1}_{A}, \underbrace{a_2, a_3}_{D}]^T \in \mathbb{R}^3 \\ &= [\underbrace{u_a}_{A}, \underbrace{u_1, u_2}_{D}]^T \qquad (8.33) \end{aligned}$$

Il reste deux choses pour obtenir un simulateur opérationnel. La première est de préciser le modèle d'évolution. La seconde est de préciser les différents paramètres du simulateur.

Modèles d'évolution

Nous avons déjà donné le modèle d'évolution des composantes d'état correspondant à l'accumulateur dans la définition 8.5, il dépend de l'action appliquée. Cependant il faut également déterminer un modèle d'évolution pour le reste des composantes du vecteur d'état. En pratique, ces différents paramètres varient en fonction d'un certain nombre de données, comme le carnet de commande des clients (quel type d'acier produire, pour quand) ou des contraintes physiques (on ne peut pas produire n'importes quels aciers l'un après l'autre) voire logistiques. Les modèles d'évolution que nous proposons ont été mis au point au cours de discussions avec l'autre équipe travaillant sur ce sujet et devraient refléter la réalité, au moins jusqu'à un certain point.

Nous distinguons deux modèles d'évolution (hormis l'accumulateur), l'un pour le flux au niveau du producteur, l'autre pour les autres quantités. Au niveau du producteur, le flux n'est pas continu, il y a des pics de production, réguliers mais non équi-espacés. Pour modéliser cela, nous introduisons deux constantes $t_{\min}, t_{\max} \in \mathbb{N}$ correspondant aux temps minimum et maximum entre deux pics de production, ainsi qu'un minuteur t. A chaque pas de temps, le minuteur décroît d'une unité. Lorsqu'il arrive à zéro, un pic de production est émis et le minuteur est réinitialisé aléatoirement (tirage uniforme) entre $t_{\min}$ et $t_{\max}$. Nous avons donc :

$$t_{i+1} = \begin{cases} t_i - 1 & \text{si } t_i > 0 \\ t_{\text{rand}} & \text{avec } t_{\text{rand}} \sim \mathcal{U}_{[t_{\min}, t_{\max}]} \text{ sinon} \end{cases} \qquad (8.34)$$

$$\text{et } f_P = \begin{cases} f_{P,\max} & \text{si } t = 0 \\ 0 & \text{sinon} \end{cases} \qquad (8.35)$$

Notons que ce minuteur est interne au simulateur, l'agent n'y a pas accès ; il observe uniquement le flux de production[1].

[1] Dès lors, on peut se demander si du point de vue de l'agent le système est markovien. Nous verrons dans la section 8.3.3 que cela ne pose pas particulièrement problème pour apprendre un bon contrôle, probablement car les situations pour lesquelles il y a un repliement perceptuel (deux états différents du système correspondant à une même observation pour l'agent) n'impliquent pas de contrôles fondamentalement différents.

Nous choisissons de modéliser toutes les autres quantités (capacités calorifiques en production, en consommations, puissances) par des processus auto-régressifs stationnaires d'ordre 1, dont la définition suit.

Définition 8.6 (Processus auto-régressif stationnaire d'ordre 1). *Nous disons que le processus aléatoire X_i est auto-régressif stationnaire d'ordre 1, de moyenne μ et de variance σ^2. Nous le notons $X_i \sim \mathrm{AR1}(\varrho, \mu, \sigma)$ avec $|\varrho| < 1$ si sa loi d'évolution est :*

$$X_i = \varrho X_{i-1} + (1-\varrho)\mu + \sigma(1-\varrho)^{\frac{1}{2}} n_i \tag{8.36}$$

où n_i est un bruit blanc gaussien centré réduit.

Choix des paramètres du réseau

Les choix que nous faisons pour les paramètres du réseau sont sensiblement arbitraires (ils n'ont pas été calibrés sur des données réelles), d'un point de vue physique, mais ils sont fait de façon à garantir une évolution temporelle intéressante. Ainsi, les valeurs absolues des différentes quantités du simulateur ne reflètent pas la réalité, par contre leur évolution devrait en être assez proche. De plus nous faisons l'hypothèse que les différentes capacités calorifiques (du gaz produit et nécessaires pour les consommateurs) sont constantes. D'après nos interactions avec l'équipe travaillant sur le sujet, elle n'est pas très restrictive. Les paramètres à préciser sont :

- pour chaque tuyau les flux minimal et maximal acceptés ;
- pour le producteur, le flux produit lors des pics, la capacité calorifique associée, ainsi que les temps minimal et maximal entre deux pics ;
- pour l'accumulateur, sa capacité maximale ;
- pour les consommateurs, les puissances minimale et maximale consommables, ainsi que les capacités calorifiques minimales requises ;
- la capacité calorifique du gaz naturelle (qui est par hypothèse la plus grande) ;
- les différents coûts, voir l'équation (8.31).

De plus, pour chacune des quantités concernées, nous précisons le modèle AR1 utilisé. Notons également que nous forçons chaque composante à rester dans les bornes choisies, le processus aléatoire que nous utilisons est donc plutôt :

$$X_i = \max\left(\min\left(\varrho X_{i-1} + (1-\varrho)\mu + \sigma(1-\varrho)^{\frac{1}{2}} n_i, X_{\max}\right), X_{\min}\right) \tag{8.37}$$

D'autre part, les valeurs du modèle AR1 que nous donnons sont pour des composantes d'état normalisées entre 0 et 1, c'est-à-dire qu'on considère plutôt le processus X'_i défini par $X'_i = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}}$ (uniquement pour la définition des paramètres AR1).

Pour les différents tuyaux (notés T_1 à T_4 sur la figure 8.9), nous choisissons les flux minimaux et maximaux suivants :

$$\begin{cases} f^{T_1}_{\min} = 0 kNm^3.h^{-1} \\ f^{T_1}_{\max} = 300 kNm^3.h^{-1} \end{cases}, \begin{cases} f^{T_2}_{\min} = 0 kNm^3.h^{-1} \\ f^{T_2}_{\max} = 300 kNm^3.h^{-1} \end{cases} \tag{8.38}$$

$$\begin{cases} f^{T_3}_{\min} = 0 kNm^3.h^{-1} \\ f^{T_3}_{\max} = 300 kNm^3.h^{-1} \end{cases}, \begin{cases} f^{T_4}_{\min} = 0 kNm^3.h^{-1} \\ f^{T_4}_{\max} = 300 kNm^3.h^{-1} \end{cases} \tag{8.39}$$

Nous rappelons que ces flux conditionnent les flux minimaux et maximaux au niveau des nœuds. Pour le producteur, nous choisissons les valeurs suivantes (rappelons que s_1 correspond au flux produit et s_2 à la capacité calorifique associée, constante d'après l'hypothèse) :

$$s_{1,\max} = 300kNm^3.h^{-1}, s_2 = 10Mcal.kNm^{-3} \tag{8.40}$$

Les temps minimum et maximum entre deux pics de productions doivent également être choisis :

$$t_{\min} = 7, \quad t_{\max} = 15 \tag{8.41}$$

Pour l'accumulateur, il suffit de choisir sa capacité maximale (pas de modèle AR1). Nous rappelons que s_3 correspond au flux propre à l'accumulateur et s_4 à la capacité calorifique associée :

$$s_{3,\max} = 300kNm^3.h^{-1} \tag{8.42}$$

Pour le consommateur C_1, il faut choisir les puissances et capacités calorifiques extrêmes possibles. Nous rappelons que s_5 correspond à la puissance et s_6 à la capacité calorifique (minimale nécessaire) :

$$\begin{cases} s_{5,\min} = 15Mcal.h^{-1} \\ s_{5,\max} = 20Mcal.h^{-1} \end{cases}, s_6 = 10Mcal.kNm^{-3} \tag{8.43}$$

Il faut également choisir le modèle AR1 correspondant :

$$s_5 \sim \text{AR1}(0.4, 0.5, 0.2) \tag{8.44}$$

Il faut définir les mêmes paramètres pour le consommateur C_2 :

$$\begin{cases} s_{7,\min} = 10Mcal.h^{-1} \\ s_{7,\max} = 15Mcal.h^{-1} \end{cases}, s_8 = 10Mcal.kNm^{-3} \tag{8.45}$$

Il faut également choisir le modèle AR1 correspondant :

$$s_7 \sim \text{AR1}(0.7, 0.5, 0.4) \tag{8.46}$$

Nous choisissons pour le gaz naturel une valeur plus grande que les autres :

$$c_n = 15Mcal.kNm^{-3} \tag{8.47}$$

Il ne reste plus qu'à définir les composantes de la fonction récompense. Rappelons son expression (8.31) :

$$r(s,a) = \sum_{i=1}^{5} \left(\alpha_T^{(i)} f_T^{(i)} + \alpha_G^{(i)} G_n^{(i)} \right) \tag{8.48}$$

On peut remarquer que les coefficients peuvent se diviser en cinq catégories : torchage réel, noté α_T, torchage virtuel (pas de torche physiquement possible), noté α_{T_v}, vente à l'usine d'électricité, noté α_{PP} (et nous rappelons que la récompense associée est proportionnelle à $\alpha_{PP}\frac{c}{c_n}$), achat de gaz naturel, noté α_G et achat de gaz naturel virtuel, noté α_{G_v}. Nous choisissons pour ces différents facteurs les valeurs suivantes :

$$\begin{cases} \alpha_T = 0€.h.kNm^{-3} \\ \alpha_{T_v} = -5€.h.kNm^{-3} \\ \alpha_{PP} = 0.1€.h.kNm^{-3} \\ \alpha_G = -30€.h.kNm^{-3} \\ \alpha_{G_v} = -2€.h.kNm^{-3} \end{cases} \tag{8.49}$$

8.3.2 Semi-automatisation de la paramétrisation

Un choix qui peut s'avérer difficile et que nous avons peu abordé jusqu'à présent est celui de la paramétrisation de la fonction de valeur, ou plus généralement de la fonction de qualité. Il existe des méthodes visant à adapter les fonctions de base au problème traité, on peut citer [67], [78] ou encore [125] par exemple. Dans cette section, nous proposons de semi-automatiser une représentation par noyaux [96, 118], la méthode étant originellement proposée par [29] dans un contexte légèrement différent. Etant donné le choix *a priori* d'un type de noyau, elle permet de choisir de façon automatique les fonctions de base *ad hoc*.

Un noyau peut être vu comme la généralisation fonctionnelle de la notion de matrice positive.

Définition 8.7 (Noyau). *Soit X un espace mesurable, un noyau K défini par :*

$$\begin{aligned} K : X \times X &\to \mathbb{R} \\ x, y &\to K(x, y) \end{aligned} \tag{8.50}$$

est une fonction continue, symétrique et positive, c'est-à-dire que pour tout sous-ensemble fini $\{x_1, \ldots, x_n\}$ de X et tout ensemble de nombre réels $\{c_1, \ldots, c_n\}$, on a :

$$\sum_{i,j} K(x_i, x_j) c_i c_j \geq 0 \tag{8.51}$$

L'exemple qui est peut-être le plus classique et que nous avons utilisé souvent jusqu'à présent est le noyau Gaussien (Σ étant définie positive) :

$$K(x, y) = \exp\left(-\frac{1}{2}(x - y)^T \Sigma^{-1} (x - y)\right) \tag{8.52}$$

Un autre type de noyau classique est le noyau polynomial :

$$K(x, y) = \left(x^T y + c\right)^q \tag{8.53}$$

Il existe beaucoup d'autres noyaux, nous ne les décrirons pas tous. L'intérêt et le succès des méthodes à noyaux repose sur le théorème de Mercer, qui stipule que le noyau $K(x, y)$ correspond en fait au produit scalaire des images de x et y dans un espace préhilbertien[2] potentiellement de plus grande dimension.

Théorème 8.1 (Mercer). *Soit $K : X^2 \to \mathbb{R}$ un noyau, il existe une application $\varphi : X \to \mathcal{F}$, où $\mathcal{F}$ est un espace préhilbertien appelé espace de redescription (feature space en anglais), telle que $K(x, y)$ soit le produit scalaire de $\varphi(x)$ et $\varphi(y)$, pour tout x et y de X :*

$$K(x, y) = \langle \varphi(x), \varphi(y) \rangle \tag{8.54}$$

Ce résultat est très important, dans la mesure où il est à la base de l'astuce dite du noyau (*kernel trick* en anglais) : tout algorithme linéaire s'exprimant uniquement en terme de produits scalaires peut être utilisé pour résoudre un problème non-linéaire en remplaçant ces produits scalaires par des noyaux, ce qui a pour effet de transformer implicitement l'espace de représentation des données en un espace généralement de plus grande dimension. Un problème linéaire dans l'espace de redescription de plus grande dimension (pour un noyau

[2] C'est un espace vectoriel muni d'un produit scalaire.

gaussien par exemple, cet espace est de dimension infinie) correspondra à un problème non-linéaire dans l'espace d'origine.

Revenons plus précisément au problème qui nous intéresse, à savoir la représentation paramétrique d'une fonction d'intérêt. Nous nous abstrayons cependant un peu de l'apprentissage par renforcement et considérons un fonction $\hat{f}_\theta(x)$ paramétrée par un vecteur θ, x n'étant pas forcément scalaire. La représentation paramétrique la plus simple possible est linéaire :

$$\hat{f}_\theta(x) = \theta^T x = \langle \theta, x \rangle \tag{8.55}$$

Si cette représentation est simple, elle n'est pas représentative d'une grande classe de fonctions. Une paramétrisation par noyau, quant à elle, est de la forme suivante :

$$\hat{f}_\theta(x) = \sum_{i=1}^{p} w_i K(x, x_i) \text{ avec } \theta = (w_1, \ldots, w_p)^T \tag{8.56}$$

En appliquant le théorème de Mercer, on peut se ramener à une forme de produit scalaire dans l'espace de redescription :

$$\begin{aligned} \hat{f}_\theta(x) &= \sum_{i=1}^{p} w_i K(x, x_i) \\ &= \sum_{i=1}^{p} w_i \langle \varphi(x), \varphi(x_i) \rangle \\ &= \left\langle \varphi(x), \sum_{i=1}^{p} w_i \varphi(x_i) \right\rangle \end{aligned} \tag{8.57}$$

Ceci justifie le choix de la représentation, mais ne permet pas de répondre aux questions importantes que sont le choix du type de noyau, le choix du nombre de noyaux (paramètre p) et le choix de leurs positions (paramètres x_i). En supposant le choix du type de noyau fait, la méthode de dictionnaire que nous présentons maintenant permet de déterminer de façon automatique le nombre de noyaux et leurs positions. Elle est due à [29] et repose également sur le théorème de Mercer.

Rappelons la forme de la régression (8.57). La fonction paramétrée est le produit scalaire des vecteurs $\varphi(x)$ et $\sum_{i=1}^{p} w_i \varphi(x_i)$. Si $(\varphi(x_1), \ldots, \varphi(x_p))$ est une famille liée, il y a de la redondance. Si elle est libre, mais non génératrice de $\varphi(X)$, il y a une perte du pouvoir de représentation de la fonction paramétrée. L'idée est donc de construire une base (approximative) de $\varphi(X)$ et ce sans utiliser explicitement la fonction φ. Même si l'espace de redescription est de grande dimension, le sous-espace $\varphi(X)$ peut être de dimension plus raisonnable. Nous cherchons donc un ensemble minimal de p points $\tilde{x}_1, \ldots, \tilde{x}_p$ de X tels que

$$\varphi(X) \approx \text{Vect}\,\{\varphi(\tilde{x}_1), \ldots, \varphi(\tilde{x}_p)\} \tag{8.58}$$

Cette méthode est en-ligne par nature. Supposons que les échantillons $x_1, x_2, \ldots$ soient séquentiellement observés (soit qu'ils sont disponibles à l'avance, soit qu'ils sont échantillonnés selon une loi uniforme sur X par exemple). Au temps t un dictionnaire $\mathcal{D}_{t-1}$ de m_{t-1} éléments est disponible :

$$\mathcal{D}_{t-1} = (\tilde{x}_j)_{j=1}^{m_{t-1}} \subset (x_j)_{j=1}^{t-1} \tag{8.59}$$

Par construction, les vecteurs $\varphi(\tilde{x}_i)$ sont approximativement linéairement indépendants dans l'espace $\mathcal{F}$ de redescription. Un échantillon x_t de X est alors observé. Si $\varphi(x_t)$ est approximativement linéairement indépendant de $\varphi(\mathcal{D})$ alors il est ajouté au dictionnaire. Pour tester la dépendance, il faut résoudre le problème d'optimisation suivant, en notant a le vecteur $(a_1, \ldots, a_{m_{t-1}})^T$:

$$\delta_t = \min_{a \in \mathbb{R}^{m_{t-1}}} \left\| \sum_{j=1}^{m_{t-1}} a_j \varphi(\tilde{x}_j) - \varphi(x_t) \right\|^2 \tag{8.60}$$

Formellement, si $\delta_t = 0$, alors on a une dépendance linéaire, sinon non. En pratique, une dépendance approchée est autorisée et δ_t est comparée à un seuil ν (choisi par l'utilisateur). De ce choix du seuil dépend la qualité de l'approximation et la parcimonie de la représentation (il s'agit de trouver un compromis entre les deux). On considère donc qu'il y a dépendance si $\delta_t < \nu$ et alors le dictionnaire reste inchangé. Dans le cas contraire l'élément x_t y est ajouté.

L'équation (8.60) peut être réécrite comme un produit scalaire, développée et les produits scalaires remplacés par les noyaux :

$$\begin{aligned} \delta_t &= \min_{a \in \mathbb{R}^{m_{t-1}}} \left\langle \sum_{j=1}^{m_{t-1}} a_j \varphi(\tilde{x}_j) - \varphi(x_t), \sum_{j=1}^{m_{t-1}} a_j \varphi(\tilde{x}_j) - \varphi(x_t) \right\rangle \\ &= \min_{a \in \mathbb{R}^{m_{t-1}}} \left\{ \sum_{i,j=1}^{m_{t-1}} a_i a_j K(\tilde{x}_i, \tilde{x}_j) - 2 \sum_{j=1}^{m_{t-1}} a_j K(\tilde{x}_j, x_t) + K(x_t, x_t) \right\} \end{aligned} \tag{8.61}$$

En définissant la matrice $\tilde{K}_{t-1}$ et le vecteur $\tilde{k}_{t-1}(x)$ de tailles respectives $m_{t-1} \times m_{t-1}$ et $m_{t-1} \times 1$ par :

$$\left(\tilde{K}_{t-1}\right)_{i,j} = K(\tilde{x}_i, \tilde{x}_j) \text{ et } \left(\tilde{k}_{t-1}(x)\right)_i = K(x, \tilde{x}_i) \tag{8.62}$$

il est possible de réécrire l'équation (8.60) sous forme matricielle :

$$\delta_t = \min_{a \in \mathbb{R}^{m_{t-1}}} \left\{ a^T \tilde{K}_{t-1} a - 2 a^T \tilde{k}_{t-1}(x_t) + K(x_t, x_t) \right\}$$

Ce problème quadratique peut être résolu analytiquement et sa solution est donnée par :

$$\begin{cases} a_t = \tilde{K}_{t-1}^{-1} \tilde{k}_{t-1}(x_t) \\ \delta_t = K(x_t, x_t) - \tilde{k}_{t-1}(x_t)^T a_t \end{cases} \tag{8.63}$$

Si $\delta_t \leq \nu$, $\varphi(x_t)$ est approximativement linéairement dépendant de $\varphi(\mathcal{D}_{t-1})$ et peut être réécrit comme :

$$\begin{aligned} \varphi(x_t) &= \sum_{i=1}^{m_{t-1}} a_i \varphi(\tilde{x}_i) + \varphi_t^{res}, \quad \|\varphi_t^{res}\| \leq \sqrt{\nu} \\ &\approx \sum_{i=1}^{m_{t-1}} a_i \varphi(\tilde{x}_i) \end{aligned} \tag{8.64}$$

Sinon, $\delta_t > \nu$ et $x_t = \tilde{x}_{m_t}$ est ajouté au dictionnaire.

Il est à noter que la matrice $\tilde{K}_t^{-1}$ peut être calculée efficacement. Plutôt que de faire une inversion matricielle de complexité $O(m_t^3)$, il est possible de réduire la complexité à $O(m_t^2)$. Si $\delta_t < \nu$ aucun point n'est ajouté au dictionnaire et $\tilde{K}_t^{-1} = \tilde{K}_{t-1}^{-1}$. Si x_t est ajouté au dictionnaire, la matrice $\tilde{K}_t^{-1}$ peut être écrite par bloc et son inverse peut être déterminée analytiquement en utilisant la formule de l'inverse de la matrice de covariance partionnée (voir [27]) :

$$\tilde{K}_t = \begin{pmatrix} \tilde{K}_{t-1} & \tilde{k}_{t-1}(x_t) \\ \tilde{k}_{t-1}(x_t)^T & K(x_t, x_t) \end{pmatrix} \Leftrightarrow \tilde{K}_t^{-1} = \frac{1}{\delta_t} \begin{pmatrix} \delta_t \tilde{K}_{t-1}^{-1} + a_t a_t^T & -a_t \\ -a_t^T & 1 \end{pmatrix} \tag{8.65}$$

L'approche permet donc de calculer séquentiellement une approximation de la base de $\varphi(X)$. Pour une analyse de l'approche (notamment qualité de l'approximation et bornes sur la taille du dictionnaire en fonction du facteur de parcimonie ν), voir [29, 27]. Nous résumons la construction du dictionnaire dans l'algorithme 8.1.

Algorithme 8.1 : Calcul du dictionnaire

entrées : un ensemble de N points $x_1, \ldots, x_N$ de X, facteur de parcimonie ν
sortie : un dictionnaire $\mathcal{D}$

Initialisation;
$\mathcal{D}_1 = \{x_1\}$;
$\tilde{K}_1^{-1} = \frac{1}{K(x_1,x_1)}$;
$m = 1$;

Calcul du dictionnaire;
pour $t = 2, \ldots N$ **faire**
  Observer l'échantillon x_t;
  Calculer $\tilde{k}_{t-1}(x_t)$, voir équation (8.62);
  Calcul de la liberté;
  $a_t = \tilde{K}_{t-1}^{-1} \tilde{k}_{t-1}(x_t)$;
  $\delta_t = K(x_t, x_t) - \tilde{k}_{t-1}(x_t)^T a_t$;
  si $\delta_k > \nu$ **alors**
    $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{x_t\}$;
    Calculer $\tilde{K}_t^{-1}$, voir équation (8.65);
    $m \leftarrow m + 1$
  sinon
    $\tilde{K}_t^{-1} = \tilde{K}_{t-1}^{-1}$;
    $\mathcal{D}_t = \mathcal{D}_{t-1}$;

Pour utiliser cette méthode de dictionnaire dans le cadre de KTD, nous supposons connaître *a priori* les espaces d'état et d'action. Nous choisissons un noyau K, un facteur de parcimonie ν et nous appliquons la méthode de dictionnaire en pré-traitement sur un ensemble de N couples état-action (cas de la paramétrisation de la Q-fonction) tirés uniformément dans $S \times A$. C'est cette méthode que nous utilisons dans les publications [39, 38, 45], en paramétrant cependant également les hyper-paramètres (variance associée à un noyau gaussien par exemple). Rappelons que cette approche peut être vue comme une adaptation à KTD de la représentation non-paramétrique du cadre de travail GPTD [27].

8.3.3 Résultats

Ici nous nous proposons de comparer les politiques obtenues par KTD et Monte Carlo sur le simulateur considéré, la seconde approche étant, à l'utilisation d'heuristiques pour prendre en compte les accumulateurs près, le type d'approche qui est utilisé en pratique (c'est-à-dire maximisation de la récompense immédiate, plutôt que maximisation du cumul de récompenses sur le long terme). Nous commençons par présenter les résultats de Monte Carlo sur la figure 8.10. La politique est déterminée en choisissant, parmi un grand nombre d'actions tirées aléatoirement, celle qui maximise la récompense immédiate.

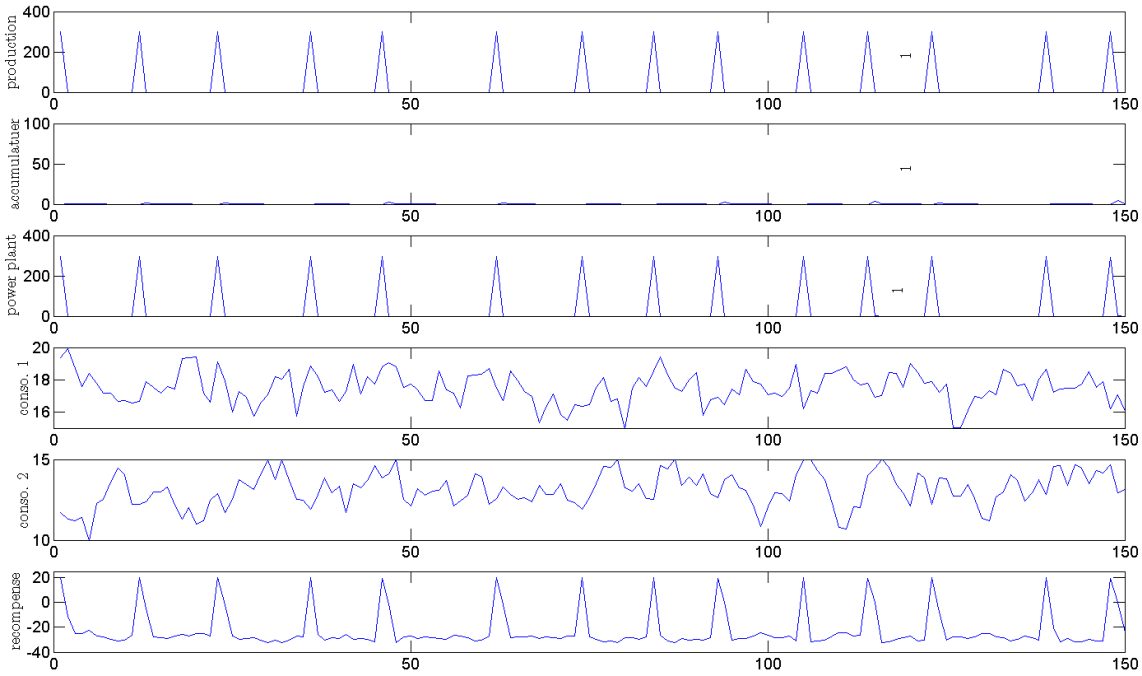


FIG. 8.10 – Politique obtenue avec Monte Carlo.

Ce résultat, composé de six graphiques, représente une évolution typique du système lorsque la politique maximise la récompense immédiate et s'interprète comme suit. La première ligne représente le flux de gaz produit au cours du temps. On y observe bien la structure temporelle annoncée, à savoir des pics de production, réguliers mais non identiquement espacés. La seconde ligne représente la quantité de gaz contenue par l'accumulateur au cours du temps. Il apparaît clairement qu'il n'est pas utilisé. La troisième ligne représente le gaz vendu à l'usine de production d'électricité. On y voit que la quasi-totalité du flux injecté dans le réseau lors des pics de production est vendu. Ce comportement est cohérent, puisque c'est la récompense immédiate qui est maximisée. Les quatrième et cinquième lignes représentent respectivement l'évolution des puissances requises par les consommateurs un et deux. La dernière ligne représentent la récompense instantanée. Lorsqu'elle est négative, c'est qu'il n'y a pas assez de gaz pour alimenter les consommateurs, qui doivent en acheter sur le réseau extérieur. Lorsqu'elle est positive, c'est que du gaz est vendu l'usine de production d'électricité. On voit ici aussi que l'ensemble du gaz est vendu dès que possible (moins le gaz utilisé pour l'alimentation des consommateurs au moment du pic de production) et qu'il est ensuite nécessaire d'acheter du gaz. Notons que ce n'est pas tout à fait le type de politique employée en pratique. Des heuristiques sont utilisées pour prendre en compte

les accumulateurs, ce qui pourrait s'interpréter comme l'ajout d'un terme de récompense relatif à leur utilisation et qui permettrait d'obtenir un meilleur contrôle en maximisant la récompense immédiate. Nous nous intéressons maintenant au type de politique obtenu grâce à l'apprentissage par renforcement.

L'algorithme considéré est KTD-SARSA combiné à une politique de Thompson (voir section 6.3.6). Nous faisons certains choix pour la représentation de la fonction de valeur. Les capacités calorifiques étant constantes, il n'est pas nécessaire de considérer les composantes d'état associées. Au niveau du diviseur le choix des actions correspondant au routage du gaz n'a aucune influence sur l'évolution du système, uniquement sur la récompense immédiate. Les actions optimales correspondantes consistent donc à maximiser la récompense immédiate, ce qui peut être fait de manière analytique. C'est ce que nous faisons et les deux actions correspondantes ne sont pas considérées dans la Q-fonction. Sur un système réel, cela revient à dire que si une décision locale n'a aucune influence sur l'évolution du système et qu'il existe un algorithme efficace pour la prendre, autant l'utiliser. Au niveau des consommateurs, l'information de la puissance requise est redondante. On la retrouve dans les composantes d'état associées, mais également dans la récompense. En effet, si trop peu de gaz est fourni aux consommateurs, cela créé une contribution négative au terme de récompense instantané. Nous ignorons donc ces composantes dans la fonction de qualité. Du point de vue de l'agent, cela rend l'information de récompense stochastique au lieu de déterministe. Le contrôle résultant peut être légèrement moins fin, mais cependant suffisant pour une validation du concept. Enfin, c'est la somme du flux de production et du flux propre à l'accumulateur qui est importante pour la prise de décision. Plutôt que de considérer les deux composantes d'état séparées pour la Q-fonction, nous en considérons la somme. Enfin, nous choisissons des noyaux gaussiens pour la représentation de la fonction de qualité et nous utilisons la méthode décrite section 8.3.2 pour automatiser la paramétrisation attenante. Les actions étant continues, nous utilisons Monte Carlo lorsqu'il est nécessaire de calculer un maximum. Le type de politique obtenu par cette approche est représentée sur la figure 8.11.

La signification des différentes lignes est la même que pour la figure 8.10. On remarque que, cette fois-ci, la politique utilise bien l'accumulateur (deuxième ligne). Elle le remplit avec suffisamment de gaz pour assurer les besoins de consommation durant une période (période que nous rappelons ne pas être constante) et vend l'excédent à l'usine de production d'électricité (troisième ligne). Le taux de remplissage de l'accumulateur correspond environ à l'intégrale des besoins moyens entre deux pics de production. Si l'on observe la courbe de récompense (sixième ligne), il apparaît qu'elle est rarement négative, ce qui signifie que suffisamment de gaz est fourni aux consommateurs presque tout le temps. La récompense obtenue lors des pics de production (correspondant à la vente de gaz excédentaire donc) est plus faible que pour Monte Carlo. Ce type de comportement est typique des algorithmes d'apprentissage par renforcement et c'est ce que l'on souhaite observer. Il peut être nécessaire de choisir initialement une action moins récompensée qu'une autre pour obtenir un plus grand gain sur le long terme.

Nous supposons que ce type de résultat aurait pu être obtenu avec d'autres algorithmes d'apprentissage par renforcement et que le problème traité ne met pas forcément en avant l'intérêt de KTD par rapport à l'état de l'art, mais plutôt de l'apprentissage par renforcement par rapport à des méthodes d'optimisation statiques (qui ne prennent pas en compte le long terme). Ce chapitre a pour but de montrer que ce type de méthodologie peut avoir des applications pratiques, ce qui est un point intéressant en soi. D'autre part, les poli-

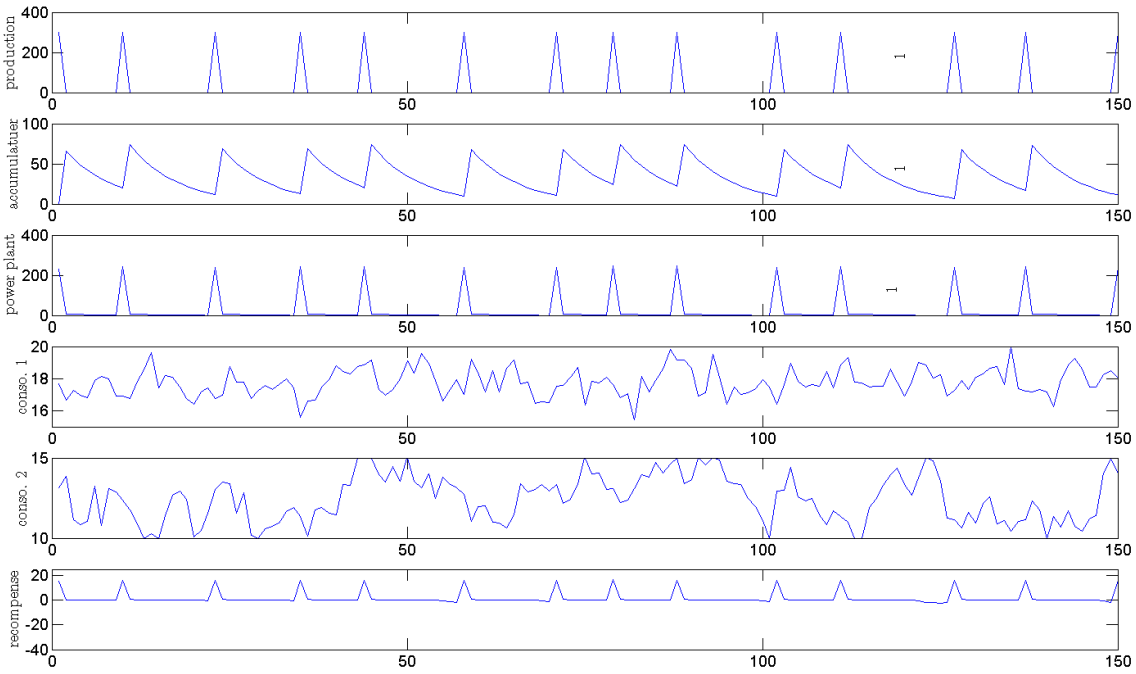


FIG. 8.11 – Politique obtenue avec KTD.

tiques utilisées sur les chaînes de production sont bien sûr meilleures que celle que nous illustrons sur la figure 8.10, dans la mesure où, comme nous l'avons déjà dit, elles utilisent des heuristiques pour prendre en compte les accumulateurs. Nous pensons que l'utilisation de méthodes telle que celle que nous proposons y est une alternative intéressante, viable et que nous pensons plus fondée.

8.3.4 Perspectives

Ainsi nous avons montré sur un problème simplifié de gestion des flux de gaz que considérer une optimisation tenant compte de la dynamique du système et dont la fonction de coût est un cumul espéré sur le long terme (nos algorithmes de renforcement en pratique) permettent d'obtenir de meilleurs résultats qu'une approche par optimisation statique (Monte Carlo dans ce cas). L'avantage de KTD sur Monte Carlo était éminemment prévisible. Cependant, cette application a le mérite de montrer que ce type de méthodologie peut trouver des applications intéressantes pour l'industriel. Dans la forme actuelle du simulateur, d'autres algorithmes de renforcement auraient pu convenir. Cependant, l'apprentissage bénéficie d'avantages de KTD comme l'efficacité en termes d'échantillons. De plus, des évolutions intéressantes du simulateur impliquent une dynamique non-stationnaire du système : la fréquence des pics de production par période peut changer (ainsi que l'intégrale des besoins sur une période) et ni le prix d'achat de gaz ni le prix de vente d'électricité ne sont stationnaires. Dans ce cas, l'avantage de KTD serait encore plus marqué.

Il y a plusieurs perspectives intéressantes, du point de vue du simulateur et des algorithmes. En ce qui concerne le simulateur, le modèle sous-jacent a été construit en interaction avec ArcelorMittal. Il sert de démonstrateur de faisabilité et les caractéristiques jugées essentielles par les praticiens (accumulateurs, structures de pics, *etc.*) ont été respectées pour que l'obtention de résultats corrects convainquent l'industriel d'aller plus loin dans

la démarche, ce qui semble être le cas. Une prochaine étape pourrait être de caler les paramètres du modèle à partir des données usine (ce qui permettrait également de voir si le modèle retranscrit bien la réalité). Une autre approche pourrait être d'apprendre directement la politique à partir de données, sans passer par un simulateur. Nos algorithmes sont *model-free*, ils le permettent. En pratique, même si les moteurs d'optimisation utilisés par ArcelorMittal sont statiques, ils sont combinés à des heuristiques, concernant notamment le remplissage des accumulateurs de gaz, de façon à prendre en compte le long terme. Il pourrait être intéressant de coupler un module de renforcement en lieu et place de ces heuristiques. Cela pourrait éventuellement permettre d'obtenir une meilleure gestion des accumulateurs. Cependant, la mise en œuvre pratique peut s'avérer difficile, les logiciels utilisés actuellement étant fermés.

Concernant les améliorations algorithmiques, le fait que les actions soient continues peut s'avérer problématique. Il faudrait pouvoir s'affranchir du calcul du maximum (ou de l'argument qui maximise, ce qui revient au même) pour faciliter le passage à l'échelle. C'est justement le cas de KNAC, la méthode acteur-critique présentée dans le chapitre 7. Il pourrait donc être intéressant de l'essayer sur ce problème, voire des problèmes plus importants. Le choix d'une paramétrisation plus compacte, comme un réseau de neurones par exemple, pourrait également aider le passage à l'échelle.

Chapitre 9

Contributions et perspectives

Ce chapitre clôt le manuscrit. Nous proposons un résumé du travail présenté, ainsi que les contributions attenantes. Nous proposons également un certain nombre de perspectives qu'il pourrait être intéressant de développer.

9.1 Résumé du travail présenté

Nous avons présenté dans le **chapitre 2** les **bases** de l'apprentissage par renforcement en général et du problème de l'approximation de la fonction de valeur en particulier. Notamment, nous avons unifié un certain nombre d'approches paramétriques usuelles (algorithmes directs, résiduels, LSTD, GPTD) sous un même formalisme, à savoir la mise à jour du vecteur de paramètres caractérisant la représentation de la fonction de valeur en fonction de l'erreur de différence temporelle faite sur la transition courante et d'un gain, l'erreur de différence temporelle dépendant du type d'équation de Bellman qui est considéré (évaluation ou optimalité) et le gain de l'algorithme. Nous avons également mis en avant un certain nombre de caractéristiques que nous pensons souhaitables, voire importantes, pour un algorithme d'approximation de la fonction de valeur. Nous souhaitions que cet algorithme permette un apprentissage en ligne et gère les non-linéarités. Nous souhaitions aussi qu'il soit efficace en termes d'échantillons, c'est-à-dire l'apprentissage d'une bonne fonction de valeur (ou mieux d'un bon contrôle) avec aussi peu d'interactions agent-système que possible. Nous le souhaitions également capable de prendre en compte la non-stationnarité. Le cas d'application le plus évident est un environnement non-stationnaire. Mais même si le système à contrôler est stationnaire, il existe un certain nombre de bonnes raisons pour préférer traquer la solution plutôt que de chercher à converger vers elle. Un certain nombre de ces raisons est donné par [106]. Cependant, celle qui nous intéresse le plus est la non-stationnarité de la fonction de valeur que l'on cherche à apprendre induite par les améliorations successives de la politique dans un cadre d'itération de la politique généralisée. Un autre aspect que nous pensons important est la capacité de l'algorithme à fournir une information d'incertitude concernant les valeurs estimées. Ce type de donnée peut s'avérer très utile pour le dilemme entre exploration et exploitation. Enfin, le coût computationnel doit rester le plus faible possible pour que l'algorithme soit utilisable.

Ces propriétés souhaitées ayant été définies, nous avons introduit un cadre de travail général dans le **chapitre 3**, que nous appelons les **différences temporelles de Kalman** (KTD). A l'origine, le filtrage de Kalman sert à traquer l'état caché d'un système dynamique non-stationnaire à partir d'observations indirectes de cet état, modélisé par une variable

aléatoire. Ce paradigme présente plusieurs avantages : algorithme d'ordre 2 et donc efficacité en termes d'échantillons, prise en compte de la non-stationnarité, gestion de l'incertitude, apprentissage en ligne et coût quadratique, ce qui correspond assez bien au cahier des charges fixé. L'idée sous-jacente à KTD est d'exprimer le problème d'approximation de la fonction de valeur comme un problème de filtrage. Les paramètres sont l'état caché à inférer, l'observation étant la récompense, liée aux paramètres via la transition effectuée et l'une des équations de Bellman. Nous avons donné l'algorithme général, qui est obtenu en minimisant l'erreur quadratique moyenne des paramètres conditionnée aux récompenses observées et qui n'accepte pas d'expression analytique excepté dans le cas linéaire (qui ne se vérifie que si la paramétrisation est linéaire et que l'équation d'observation n'est pas l'équation d'optimalité de Bellman). Pour cela, un schéma d'approximation nommé transformation non-parfumée a été utilisé, ce qui a donné naissance à trois algorithmes, à savoir KTD-V, KTD-SARSA et KTD-Q selon que l'objectif soit l'évaluation de la fonction de valeur d'une politique donnée, l'évaluation de la fonction de qualité ou l'optimisation directe de la Q-fonction (autrement dit l'évaluation directe de la politique optimale). Nous avons proposé une analyse de convergence de cette famille d'algorithmes et montré qu'ils minimisent un résidu quadratique de Bellman. Il a également été mis en avant le fait que ces algorithmes effectuent en fait une forme de descente de gradient naturel. Le cadre de travail proposé a été expérimenté sur un certain nombre de tests de référence, sur lesquels il a été montré que KTD exhibe les propriétés qu'on lui souhaitait et se compare favorablement à l'état de l'art.

Cependant, comme tout algorithme minimisant un résidu quadratique de Bellman (ce que KTD fait sous l'hypothèse d'un bruit d'observation blanc), l'estimateur correspondant à KTD est biaisé dans le cas de transitions stochastiques. C'est pour traiter ce problème que dans le **chapitre 4** nous avons introduit une **extension des différences temporelles de Kalman** (XKTD). Le principe de XKTD est d'utiliser un bruit coloré en lieu et place du bruit blanc usuel (et nécessaire à l'obtention des équations de Kalman). Ce bruit coloré, à moyenne mobile plus précisément, est obtenu en menant un raisonnement similaire sur le processus aléatoire dont la fonction de valeur est l'espérance à celui permettant d'obtenir l'équation de Bellman. En séparant ce processus aléatoire en deux parties, la fonction de valeur qui correspond à sa moyenne et un résidu aléatoire centré, nous obtenons un modèle de bruit coloré. Pour cela, les résidus sont supposés blancs (le bruit coloré étant une somme des résidus à deux instants consécutifs), ce qui est une hypothèse très forte. Cependant, malgré cela, nous pouvons montrer que XKTD minimise une erreur quadratique associant aux valeurs le retour cumulé de récompenses qui serait estimé par Monte Carlo. C'est exactement le coût minimisé par LSTD(1), qui est un estimateur non-biaisé de la fonction de valeur. XKTD est donc également un estimateur non-biaisé, au moins dans le cas linéaire. Nous avons également présenté le problème qui se posait pour un apprentissage *off-policy*. Basiquement, de la même façon que pour les traces d'éligibilité dans les algorithmes plus classiques, le bruit coloré induit un effet mémoriel qui n'est pas compensé et pose problème lorsque la politique apprise n'est pas la politique suivie. Ceci explique que nous ne présentons pas d'algorithme XKTD-Q. Nous avons également proposé une version pondérée de XKTD, potentiellement utile dans le cas de politiques aléatoires et non-stationnaires et permettant de réduire l'impact de l'effet mémoriel dans le cas d'une politique apprise non-stationnaire (ϵ-gloutonne par exemple). Cette version pondérée de l'algorithme peut être interprétée comme l'introduction d'un nouveau bruit coloré. Ces variations algorithmiques de KTD ont été expérimentées sur des tests de référence en apprentissage par renforcement.

XKTD présente globalement les mêmes caractéristiques que KTD, mais il permet en plus de supprimer le biais que rencontre KTD dans le cas des transitions stochastiques. Cependant, avec XKTD, on perd d'une part la capacité à faire de l'apprentissage *off-policy*, à cause de la non-localité des mises à jour et d'autre part les résultats sont légèrement moins bons pour XKTD que pour KTD, lorsque apprentissage et contrôle sont entrelacés (politique ϵ-gloutonne dans nos résultats expérimentaux). Nous pensons ce dernier aspect également causé par l'effet mémoriel induit par les transitions stochastiques. Comme la mise à jour ne se fait pas seulement en fonction de l'erreur de différence temporelle courante, mais (implicitement) en fonction de toutes les erreurs de la trajectoire, l'adaptation de l'algorithme est moins rapide et peut-être moins efficace.

KTD permet donc des mises à jour locales, mais est biaisé, alors que XKTD effectue des mises à jour plus globales, sans être biaisé. De plus, d'après notre résultat de convergence, on peut rapprocher XKTD de l'estimation de la fonction de valeur par des méthodes de Monte Carlo. Ainsi, dans le **chapitre 5**, comme les traces d'éligibilité permettent pour les algorithmes de différences temporelles classiques de jeter un pont entre les méthodes les plus locales (TD par exemple) et Monte Carlo, nous introduisons une **extension des différences temporelles de Kalman aux traces d'éligibilité**, KTD(λ). L'approche que nous adoptons est cependant sensiblement différente de celle utilisée à l'origine pour introduire ce concept. Plutôt que de considérer une moyenne géométrique des différences temporelles d'ordre k (vue avant des traces d'éligibilité), nous considérons un nouveau bruit d'observation qui est une moyenne géométrique de bruits d'ordre k. Nous introduisons ainsi toute une nouvelle classe de bruits dépendants du facteur d'éligibilité $\lambda \in [0, 1]$. Cette nouvelle classe d'algorithmes KTD(λ) jette en pont KTD et XKTD dans la mesure où ils en sont des cas extrêmes (respectivement pour $\lambda = 1$ et $\lambda = 0$, contrairement à la littérature où $\lambda = 1$ correspond plutôt à la version Monte Carlo). Nous proposons également une analyse de convergence pour cette nouvelle classe d'algorithmes, en montrant qu'elle minimise une fonction de coût quadratique particulière. Dans les cas extrêmes, elle se simplifie en un résidu quadratique de Bellman ($\lambda = 1$) ou en un coût quadratique associant les valeurs aux retours de Monte Carlo ($\lambda = 0$). Cependant, nous ne pouvons rien dire sur le biais de l'estimateur résultant. Nous le testons donc expérimentalement, à nouveau sur des problèmes usuels de l'apprentissage par renforcement. Empiriquement, KTD(λ) est non biaisé pour tout λ strictement inférieur à 1 ($\lambda = 1$ correspond à KTD), ce qui est le comportement que nous souhaitions. De plus, certaines valeurs de λ réduisent empiriquement la variance de l'estimateur. Enfin, croître λ améliore les résultats du contrôle dans le cadre d'une itération optimiste de la politique, ce qui tend à montrer que KTD(λ) permet effectivement un compromis entre estimateur non biaisé et localité de la mise à jour, cela en jetant un pont entre KTD et XKTD.

Nous avons annoncé la capacité à fournir une **information d'incertitude** sur les valeurs estimées comme une caractéristique désirable d'un algorithme d'approximation de la fonction de valeur. Dans le **chapitre 6**, nous avons montré comment une telle information pouvait naturellement être déduite du cadre de travail de KTD. Nous avons également introduit une forme d'**apprentissage actif** utilisant l'information d'incertitude des valeurs estimées permettant d'accélérer l'apprentissage dans le contexte de l'apprentissage *off-policy* du Q-learning. Nous avons aussi adapté un certain nombre d'approches visant à **balancer entre exploration et exploitation**, toutes utilisant une information d'incertitude et la plupart n'ayant été développées que dans le cas tabulaire (les approches permettant à la fois d'approcher la fonction de valeur et d'estimer une incertitude associée

étant particulièrement rares). Il a été montré expérimentalement que l'apprentissage actif accélérait en effet la convergence et les différents schémas visant à traiter le dilemme entre exploration et exploitation ont été testés sur un problème de bandits. Certains fonctionnent très bien, d'autre moins. Nous pensons que le problème vient principalement du fait que c'est l'information absolue d'incertitude qui est la plupart du temps utilisée et pas une information relative, ce qui suggère la nécessité de normalisation. Cependant, cela ouvre d'intéressantes perspectives.

La dernière de nos contributions théoriques est présentée au **chapitre 7**. Elle concerne **un algorithme acteur-critique**, KNAC, dont l'acteur est une politique mise à jour avec un gradient naturel et le critique est une variation de KTD(λ), qui nous le rappelons peut également être vu comme une forme de gradient naturel. Il y a plusieurs avantages à utiliser KTD comme critique. Premièrement, comme KTD traque la solution plutôt qu'il ne converge vers elle, il n'y a pas besoin de considérer deux échelles de temps ou d'oublier une partie des paramètres. D'autre part, comme KTD est un algorithme d'ordre 2, proche dans l'idée d'un gradient naturel, il laisse espérer un apprentissage rapide si combiné avec un acteur basé sur un gradient naturel. Il y a également plusieurs avantages à considérer KTD dans un contexte acteur-critique. Tout d'abord, cela permet de profiter des avantages des approches acteur-critique par rapport aux approches critique pur, comme cela a déjà été discuté auparavant. De plus, les espaces d'actions continus ne sont plus un problème dans ce cas, dans la mesure où il n'y a pas de recherche de maximum, les paramètres de la politique étant directement modifiés. Cet aspect est particulièrement important du point de vue des applications industrielles pour ArcelorMittal, dans la mesure où KTD souffre des espaces d'action continus et où ce cas se rencontre finalement fréquemment en pratique. Si les premiers résultats expérimentaux sont encourageants, avec un apprentissage sensiblement plus rapide lorsque le critique est KTD que lorsqu'il est TD, KNAC nécessite encore du travail, tant d'un point de vue théorique qu'expérimental.

Enfin, le **chapitre 8** présente une **application industrielle** qui consiste à router les gaz entre les différentes usines d'un complexe sidérurgique de façon à fonctionner le plus possible de façon autonome (*i.e.* sans acheter de gaz à l'extérieur) tout en respectant les contraintes du réseau. Ce problème peut bénéficier de la maximisation du cumul sur le long terme de récompenses, notamment grâce à la présence d'accumulateurs qui permettent de stocker le gaz en prévision de consommations futures par exemple. Nous avons proposé une modélisation générique et générale de ce type de réseau, ainsi qu'un réseau particulier dont nous avons dimensionné les paramètres de façon relativement arbitraire (mais de façon à obtenir une dynamique intéressante, fidèle à celle du système réel) et sur lequel nous avons testé KTD. Pour cela, nous avons introduit un schéma de semi-automatisation du choix de la paramétrisation en prétraitement. Nous avons comparé notre approche à une méthode de Monte Carlo maximisant la récompense immédiate, ce qui est (aux heuristiques utilisées pour prendre en compte les accumulateurs près) le type de méthode utilisée en pratique. De façon prévisible (étant donné l'importance de l'action sur le long terme pour ce type de problème), les meilleurs résultats sont obtenus avec KTD.

9.2 Perspectives

L'idée lors de l'introduction du cadre des différences temporelles de Kalman était de proposer un algorithme d'approximation de la fonction de valeur qui permette de prendre en compte d'autres aspects que nous pensons importants pour le paradigme de l'apprentissage

par renforcement, comme la gestion de la non-stationnarité ou de l'incertitude par exemple. De fait, nous pensons que cela ouvre quelques portes.

Une première perspective est de continuer l'analyse théorique de KTD et de ses variations. Par exemple, il pourrait être intéressant de quantifier l'influence du bruit d'évolution autrement qu'empiriquement. Le cadre de travail proposé pourrait également bénéficier de plus de validations expérimentales, notamment sur des problèmes du monde réel.

Nous avons vu qu'il y a un certain nombre de paramètres à choisir. Si l'*a priori* traduit la connaissance du système que l'on peut avoir à l'avance, le choix des bruits d'observation et surtout d'évolution sont moins évidents. Il existe une vaste littérature sur le filtrage adaptatif, où ces quantités sont déterminées automatiquement en fonction des données et KTD pourrait en profiter.

Concernant le bruit d'observation justement, nous en avons proposé toute une famille (dépendant du facteur d'éligibilité). Une question ouverte est de savoir quel est le "meilleur" choix de bruit et quels bruits ont un sens dans ce contexte.

Une autre perspective d'importance est le cas de l'observabilité partielle. Une approche courante consiste à estimer un état de croyance, en fait une distribution sur la probabilité de l'état courant en fonction des observations faites. C'est donc un problème de filtrage (inférer l'état du système à partir des observations), qui devrait pouvoir assez simplement être ajouté au paradigme proposé (d'autant plus que grâce à la transformation non-parfumée, il est tout à fait possible d'y considérer la fonction de valeur comme fonction d'une distribution sur l'état plutôt que comme une fonction de l'état, ce qui est déjà le cas pour les paramètres). Cependant, cela supposerait *a priori* de disposer du modèle d'évolution des états et du modèles de génération des observations à partir de ces mêmes états.

Les différences temporelles de Kalman permettent également de prendre en compte les non-linéarités. Nous l'avons principalement exploité pour KTD-Q, qui est un algorithme du type itération de la valeur, mais il pourrait aussi être intéressant de considérer des paramétrisations non-linéaires, comme des réseaux de neurones par exemple, potentiellement plus compactes. Cet aspect pourrait également s'avérer utile dans le cadre d'une recherche de bases optimales parmi une famille paramétrée.

Nous avons adapté de façon relativement directe un certain nombre de schémas pour traiter le dilemme entre exploration et exploitation. Cependant, on pourrait envisager des schémas propres à KTD, voire s'inspirer de ce qui est appelé l'exploration bayésienne et qui est un pan important du dilemme entre exploration et exploitation [26, 85, 120, 70]. Basiquement, le principe est de construire un méta-MDP comprenant l'incertitude du modèle, cette incertitude étant mise à jour en utilisant de l'inférence bayésienne. Résoudre le problème de planification sur le méta-MDP résout implicitement le dilemme entre exploration et exploitation sur le MDP original, le goulot d'étranglement étant que le méta-MDP est bien plus complexe à résoudre que l'original. Nous citons ces approches car il est possible de tracer des parallèles entre l'exploration bayésienne, l'incertitude maintenue par KTD et certaines des heuristiques que nous proposons, comme la politique de Thompson (voir par exemple [120]). Les formalismes étant relativement proches, du moins conceptuellement, il pourrait être intéressant de développer cet axe (qui se raccrocherait à quelque chose de plus général que serait l'exploration bayésienne dite *model-free*, qui n'a pas fait l'objet de publication à notre connaissance).

Nous avons vu que KTD(λ) ne permettait pas de faire d'apprentissage *off-policy* lorsque le facteur d'éligibilité est différent de 1. Cependant, ce type d'apprentissage est d'intérêt, pour les algorithmes du type itération de la valeur, ou encore pour apprendre une politique

à partir de données imposées (dans un contexte industriel, il est peu probable que l'apprentissage se fasse en ligne, directement sur les machines). Une perspective intéressante pourrait être d'utiliser le principe de l'échantillonnage préférentiel (basiquement la détermination de quantités liée à une distribution alors que les échantillons sont générés selon une distribution différente).

Un autre point d'amélioration concerne KNAC. Nous avons posé certaines bases préliminaires et proposé un premier algorithme fonctionnel, pour lequel les résultats expérimentaux sont assez prometteurs. Au vu des avantages potentiels de ce type d'approche, l'utilisation de KTD dans une architecture acteur-critique mérite plus de travail, tant du point de vue théorique qu'expérimental.

Enfin, l'aspect non-stationnaire de KTD pourrait s'avérer intéressant dans un contexte multi-agent, notamment du point de vu adaptation. En effet, pour les modèles multi-agent dans lesquels il n'y a pas de coordinateur centralisé (DEC-(PO)MDP), l'environnement d'un agent peut-être considéré comme rendu non-stationnaire par l'ensemble des autres agents. La faculté de KTD à s'adapter à cette non-stationnarité pourrait dès lors être exploitée dans ce contexte.

Bibliographie

[1] Shun-Ichi Amari : Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[2] András Antos, Csaba Szepesvári et Rémi Munos : Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *In Conference On Learning Theory (COLT 06)*, pages 574–588, Pittsburgh, USA, 2006.

[3] András Antos, Csaba Szepesvári et Rémi Munos : Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008.

[4] Peter Auer et Ronald Ortner : Logarithmic Online Regret Bounds for Undiscounted Reinforcement Learning. *In* B. Schölkopf, J. Platt et T. Hoffman, éditeurs : *Advances in Neural Information Processing Systems (NIPS 19)*, pages 49–56. MIT Press, Cambridge, MA, 2007.

[5] Leemon C. Baird : Advantage Updating. Rapport technique WL-TR-93-1146, Wright Laboratory, Wright-Pattersin Air Force Base, 1993.

[6] Leemon C. Baird : Residual Algorithms : Reinforcement Learning with Function Approximation. *In Proceedings of the International Conference on Machine Learning (ICML 95)*, pages 30–37, 1995.

[7] Richard Bellman : A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

[8] Richard Bellman : *Dynamic Programming*. Dover Publications, sixth édition, 1957.

[9] Richard Bellman, Robert Kalaba et Bella Kotkin : Polynomial approximation - a new computational technique in dynamic programming : allocation processes. *Mathematical Computation*, 17:155–161, 1973.

[10] D.S. Bernstein, R. Givan, N. Immerman et S. Zilberstein : The complexity of decentralized control of Markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[11] Dimitri P. Bertsekas : *Dynamic Programming and Optimal Control*. Athena Scientific, 3ème édition, 1995.

[12] Dimitri P. Bertsekas et John N. Tsitsiklis : *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*. Athena Scientific, 1996.

[13] Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh et Mark Lee : Incremental natural actor-critic algorithms. *In Conference on Neural Information Processing Systems (NIPS)*, Vancouver, Canada, 2007.

[14] Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh et Mark Lee : Natural Actor-Critic Algorithms. *Automatica*, 2009.

[15] Christopher M. Bishop : *Neural Networks for Pattern Recognition*. Oxford University Press, New York, USA, 1995.

[16] Vivek S. Borkar : Controlled diffusion processes. *Probability Surveys*, 2:213–244, 2005.

[17] Justin A. Boyan : Least-squares temporal difference learning. *In Proceedings of the 16th International Conference on Machine Learning (ICML 99)*, pages 49–56. Morgan Kaufmann, San Francisco, CA, 1999.

[18] Justin A. Boyan : Technical Update : Least-Squares Temporal Difference Learning. *Machine Learning*, 49(2-3):233–246, 1999.

[19] Steven J. Bradtke et Andrew G. Barto : Linear Least-Squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

[20] Ronen I. Brafman et Moshe Tennenholtz : R-max - A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

[21] David Choi et Benjamin Van Roy : A Generalized Kalman Filter for Fixed Point Approximation and Efficient Temporal-Difference Learning. *Discrete Event Dynamic Systems*, 16:207–239, 2006.

[22] Richard Dearden, Nir Friedman et David Andre : Model-Based Bayesian Exploration. *In Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 150–15, San Francisco, CA, 1999. Morgan Kaufmann.

[23] Richard Dearden, Nir Friedman et Stuart J. Russell : Bayesian Q-Learning. *In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI)*, pages 761–768, 1998.

[24] Thomas Degris, Olivier Sigaud et Pierre-Henri Wuillemin : Chi-square Tests Driven Method for Learning the Structure of Factored MDPs. *In Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI 06)*, pages 122–129, Massachusetts Institute of Technology, Cambridge, MA, USA, 2006. AUAI Press.

[25] Christos Dimitrakakis et Samy Bengio : Estimates of Parameter Distributions for Optimal Action Selection. Rapport technique IDIAP-RR 04-72, Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP), Martigny, Valais, Suisse, 2005.

[26] Michael O'Gordon Duff : *Optimal learning : Computational preocedures for Bayes-adaptive Markov decision processes*. Thèse de doctorat, University of Massachusetts Amherst, 2002.

[27] Yaakov Engel : *Algorithms and Representations for Reinforcement Learning*. Thèse de doctorat, Hebrew University, Avril 2005.

[28] Yaakov Engel, Shie Mannor et Ron Meir : Bayes Meets Bellman : The Gaussian Process Approach to Temporal Difference Learning. *In Proceedings of the International Conference on Machine Learning (ICML 03)*, pages 154–161, 2003.

[29] Yaakov Engel, Shie Mannor et Ron Meir : The Kernel Recursive Least Squares Algorithm. *IEEE Transactions on Signal Processing*, 52:2275–2285, 2004.

[30] Yaakov Engel, Shie Mannor et Ron Meir : Reinforcement Learning with Gaussian Processes. *In Proceedings of the International Conference on Machine Learning (ICML 05)*, 2005.

[31] Damien Ernst, Pierre Geurts et Louis Wehenkel : Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[32] L.A. Feldkamp, T.M. Feldkamp et D.V. Prokhorov : Neural network training with the nprkf. *In International Joint Conference on Neural Networks (IJCNN 01)*, volume 1, pages 109–114, 2001.

[33] L.A. Feldkamp et G.V. Puskorius : A signal processing framework based on dynamic neural networks with application to problems in adaptation, filtering, and classification. *In Proceedings of the IEEE*, volume 86, pages 2259–2277, 1998.

[34] Norman Ferns, Prakash Panangaden et Doina Precup : Metrics for Finite Markov Decision Processes. *In Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI 04)*, pages 162–16, Arlington, Virginia, 2004. AUAI Press.

[35] Norman Ferns, Prakash Panangaden et Doina Precup : Metrics for Markov Decision Processes with Infinite State Spaces. *In Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI 05)*, page 201, Arlington, Virginia, 2005. AUAI Press.

[36] Norman F. Ferns, Pablo S. Castro, Doina Precup et Prakash Panangaden : Methods for computing state similarity in Markov Decision Processes. *In Proceedings of the 22nd Conference on Uncertainty in Artificial intelligence (UAI 06)*, Cambridge, MA, USA, 2006.

[37] Matthieu Geist, Olivier Pietquin et Gabriel Fricout : A Sparse Nonlinear Bayesian Online Kernel Regression. *In Proceedings of the Second IEEE International Conference on Advanced Engineering Computing and Applications in Sciences (AdvComp 2008)*, pages 199–204, Valencia (Espagne), Octobre 2008.

[38] Matthieu Geist, Olivier Pietquin et Gabriel Fricout : Bayesian Reward Filtering. *In* S. Girgin et al., éditeur : *Proceedings of the European Workshop on Reinforcement Learning (EWRL 2008)*, volume 5323 de *Lecture Notes in Artificial Intelligence*, pages 96–109. Springer Verlag, Lille (France), Juin 2008.

[39] Matthieu Geist, Olivier Pietquin et Gabriel Fricout : Filtrage bayésien de la récompense. *In actes des Journées Francophones de Planification, Décision et Apprentissage pour la conduite de systèmes (JFPDA 2008)*, pages 113–122, Metz (France), Juin 2008.

[40] Matthieu Geist, Olivier Pietquin et Gabriel Fricout : Kalman Temporal Differences : Uncertainty and Value Function Approximation. *In NIPS Workshop on Model Uncertainty and Risk in Reinforcement Learning*, Vancouver (Canada), Décembre 2008.

[41] Matthieu Geist, Olivier Pietquin et Gabriel Fricout : Online Bayesian Kernel Regression from Nonlinear Mapping of Observations. *In Proceedings of the 18th IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2008)*, Cancun (Mexico), 2008.

[42] Matthieu Geist, Olivier Pietquin et Gabriel Fricout : Différences Temporelles de Kalman. *In actes des Journées Francophones de Planification, Décision et Apprentissage pour la conduite de systèmes (JFPDA 2009)*, Paris, Juin 2009.

[43] Matthieu Geist, Olivier Pietquin et Gabriel Fricout : Différences temporelles de Kalman : cas déterministe. *Revue d'Intelligence Artificielle*, 2009. acceptée pour publication.

[44] Matthieu Geist, Olivier Pietquin et Gabriel Fricout : Différences Temporelles de Kalman : le cas stochastique. *In actes des Journées Francophones de Planification, Décision et Apprentissage pour la conduite de systèmes (JFPDA 2009)*, Paris, 2009.

[45] Matthieu Geist, Olivier Pietquin et Gabriel Fricout : From Supervised to Reinforcement Learning : a Kernel-based Bayesian Filtering Framework. *International Journal On Advances in Software*, 2(1), 2009.

[46] Matthieu Geist, Olivier Pietquin et Gabriel Fricout : Kalman Temporal Differences : the deterministic case. *In IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2009)*, page 8 pages, Nashville, TN, USA, April 2009.

[47] Matthieu Geist, Olivier Pietquin et Gabriel Fricout : Kernelizing Vector Quantization Algorithms. *In European Symposium on Artificial Neural Networks (ESANN 09)*, Bruges, Belgique, 2009.

[48] Matthieu Geist, Olivier Pietquin et Gabriel Fricout : Tracking in Reinforcement Learning. *In Proceedings of the 16th International Conference on Neural Information Processing (ICONIP 2009)*, Bangkok (Thailande), 2009. Springer.

[49] Matthieu Geist, Olivier Pietquin et Gabriel Fricout : Astuce du Noyau & Quantification Vectorielle. *In Actes du 17ème colloque sur la Reconnaissance des Formes et l'Intelligence Artificielle (RFIA'10)*, Caen (France), January 2010.

[50] Alborz Geramifard, Michael Bowling et Richard S. Sutton : Incremental Least-Squares Temporal Difference Learning. *In 21st Conference of American Association for Artificial Intelligence (AAAI 06)*, pages 356–361, 2006.

[51] Zoubin Ghahramani : Learning dynamic bayesian networks. *In Adaptive Processing of Sequences and Data Structures, International Summer School on Neural Networks, "E.R. Caianiello"-Tutorial Lectures*, pages 168–197, London, UK, 1998. Springer-Verlag.

[52] Geoffrey Gordon : Stable Function Approximation in Dynamic Programming. *In Proceedings of the International Conference on Machine Learning (IMCL 95)*, 1995.

[53] Mohinder S. Grewal et Angus P. Andrew : *Kalman Filtering : Theory and Practice*. Prentice Hall, 1993.

[54] Hirotaka Hachiya, Takayuki Akiyama, Masashi Sugiayma et Jan Peters : Adaptive Importance Sampling for Value Function Approximation in Off-policy Reinforcement Learning. *Neural Network*, 2009.

[55] Ronald A. Howard : *Dynamic Programming and Markov Processes*. The MIT Press, Cambridge, Massachusetts, 3ème édition, 1960.

[56] Sung Eun Jo et Sang Woo Kim : Consistent Normalized Least Mean Square Filtering with Noisy Data Matrix. *IEEE Transactions on Signal Processing*, 53(6):2112–2123, juin 2005.

[57] S. J. Julier et J. K. Uhlmann : Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[58] Simon J. Julier : The scaled unscented transformation. *In American Control Conference*, volume 6, pages 4555–4559, 2002.

[59] Simon J. Julier et Jeffrey K. Uhlmann : A new extension of the Kalman filter to nonlinear systems. *In Int. Symp. Aerospace/Defense Sensing, Simul. and Controls 3*, 1997.

[60] Tobias Jung et Peter Stone : Feature Selection for Value Function Approximation Using Bayesian Model Selection. *In Proceedings of European Conference on Machine Learning (ECML 2009)*, Bled, Slovenia, 2009.

[61] Leslie Pack Kaelbling : *Learning in embedded systems*. MIT Press, 1993.

[62] Leslie Pack Kaelbling, Michael L. Littman et Anthony R. Cassandra : Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[63] Sham Kakade : A Natural Policy Gradient. *In Neural Information Processing Systems (NIPS)*, pages 1531–1538, 2001.

[64] Sham Kakade, Michael J. Kearns et John Langford : Exploration in Metric State Spaces. *In International Conference on Machine Learning (ICML 03)*, pages 306–312, 2003.

[65] Rudolph Emil Kalman : A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[66] Michael Kearns et Satinder Singh : Near-Optimal Reinforcement Learning in Polynomial Time. *Machine Learning*, 49(2-3):209–232, 2002.

[67] Philipp W. Keller, Shie Mannor et Doina Precup : Automatic basis function construction for approximate dynamic programming and reinforcement learning. *In Proceedings of the 23rd international conference on Machine learning (ICML 06)*, pages 449–456, New York, NY, USA, 2006. ACM Press.

[68] Chang-Jim Kim : Time-varying parameter models with endogenous regressor. *Economics letters*, 91:21–26, 2006.

[69] Hajime Kimura et Shigenobu Kobayashi : An Analysis of Actor-Critic Algorithms Using Eligibility Traces : Reinforcement Learning with Imperfect Value Function. *In Proceedings of the Fifteenth International Conference on Machine Learning (ICML 98)*, pages 278–286, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[70] J. Zico Kolter et Andrew Y. Ng : Near-Bayesian Exploration in Polynomial Time. *In Proceedings of the 26th international conference on Machine learning (ICML 09)*, New York, NY, USA, 2009. ACM.

[71] Vijay R. Konda et John N. Tsitsikli : Actor-Critic Algorithms. *In Advances in Neural Information Processing Systems (NIPS 12)*, 2000.

[72] Vijay R. Konda et John N. Tsitsiklis : On Actor-Critic Algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003.

[73] Michail G. Lagoudakis et Ronald Parr : Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[74] Bethany R. Leffler, Michael L. Littman, Alexander L. Strehl et Thomas J. Walsh : Efficient exploration with latent structure. *In Proceedings of Robotics : Science and Systems*, Cambridge, USA, Juin 2005.

[75] Lihong Li, Michael L. Littman et Christopher R. Mansley : Online exploration in least-squares policy iteration. *In Proceedings of the Conference for research in autonomous agents and multi-agent systems (AAMAS-09)*, Budapest, Hungary, Mai 2009.

[76] Michael L. Littman : The Witness Algorithm : Solving Partially Observable Markov Decision Processes. Rapport technique, Brown University, Providence, RI, USA, 1994.

[77] Michael L. Littman, Anthony R. Cassandra et Leslie Pack Kaelbling : Efficient dynamic-programming updates in partially observable markov decision processes. Rapport technique CS-95-19, décembre 1995.

[78] Sridhar Mahadevan et Mauro Maggioni : Proto-value Function : A Laplacian Framework for Learning Representation and Control in Markov Decisions. Rapport technique TR-2006-35, University of Massachusetts, Department of Computer Science, July 2006.

[79] Tetsuro Morimura, Eiji Uchibe et Kenji Doya : Utilizing the Natural Gradient in Temporal Difference Reinforcement Learning with Eligibility Traces. *In 2nd Internatinal Symposium on Information Geometry and its Applications*, pages 256–263, Tokyo, Japan, décembre 2005.

[80] Dirk Ormoneit et Saunak Sen : Kernel-Based Reinforcement Learning. *Machine Learning*, 49:161–178, 2002.

[81] Jan Peters et Stefan Schaal : Natural Actor-Critic. *Neurocomputing*, 71:1180–1190, 2008.

[82] Jan Peters, Sethu Vijayakumar et Stefan Schaal : Natural Actor-Critic. *In* J. Gama et al., éditeur : *Proceedings of the European Conference on Machine Learning (ECML 2005)*, Lecture Notes in Artificial Intelligence. Springer Verlag.

[83] Jan Peters, Sethu Vijayakumar et Stefan Schaal : Reinforcement Learning for Humanoid Robotics. *In third ieee-ras international conference on humanoid robots (Humanoids 2003)*, 2003.

[84] Chee Wee Phua et Robert Fitch : Tracking value function dynamics to improve reinforcement learning with piecewise linear function approximation. *In Proceedings of the International Conference on Machine Learning (ICML 07)*, 2007.

[85] Pascal Poupart, Nikos Vlassis, Jesse Hoey et Kevin Regan : An analytic solution to discrete bayesian reinforcement learning. *In Proceedings of the 23rd international conference on Machine learning (ICML 06)*, pages 697–704, New York, NY, USA, 2006. ACM Press.

[86] Doina Precup, Richard S. Sutton et Satinder P. Singh : Eligibility Traces for Off-Policy Policy Evaluation. *In Proceedings of the Seventeenth International Conference on Machine Learning (ICML 00)*, pages 759–766, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[87] Philippe Preux, Sertan Girgin et Manuel Loth : Feature Discovery in Approximate Dynamic Programming. *In Proceedings of IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2009)*, Nashville, TN, USA, 2009.

[88] G.V. Puskorius et L.A. Feldkamp : Neurocontrol of nonlinear dynamical systems with Kalman filtertrained recurrent networks. *IEEE Transactions on Neural Networks*, 5(2):279–297, 1994.

[89] G.V. Puskorius et L.A. Feldkamp : Roles of learning rates, artificial process noise and square rootfiltering for extended Kalman filter training. *In Proceedings of the International Joint Conference on Neural Networks (IJCNN 99)*, volume 3, pages 1809–1814, 1999.

[90] Martin L. Puterman : *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. Wiley-Interscience, Avril 1994.

[91] Carl Edward Rassmussen et Christopher K. I. Williams : *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[92] Isabelle Rivals et Léon Personnaz : A recursive algorithm based on the extended Kalman filter for the training of feedforward neural models. *Neurocomputing*, 20(1-3):279–294, 1998.

[93] Yutaka Sakaguchi et Mitsuo Takano : Reliability of internal prediction/estimation and its application : I. adaptive action selection reflecting reliability of value function. *Neural Network*, 17(7):935–952, 2004.

[94] Daniel Schneegass : On the bias of batch Bellman residual minimisation. *Neurocomputing*, 72(7-9):2005–2008, 2009.

[95] Ralph Schoknecht : Optimality of Reinforcement Learning Algorithms with Linear Function Approximation. *In Neural Information Processing Systems(NIPS 15)*, 2002.

[96] Bernhard Scholkopf et Alexander J. Smola : *Learning with Kernels : Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[97] Wolfram Schultz, Peter Dayan et P. Read Montague : A neural substrate of prediction and reward. *Science*, 275:1593–1599, 1997.

[98] Olivier Sigaud et Olivier Buffet : *Processus décisionnels de Markov en intelligence artificielle*. Hermes Science Publications / Lavoisier, 2008.

[99] Dan Simon : *Optimal State Estimation : Kalman, H Infinity, and Nonlinear Approaches*. Wiley & Sons, 1. auflage édition, August 2006.

[100] H. W. Sorenson : Approximate solutions of the nonlinear filtering problem. *In Conference on Decision and Control including the 16th Symposium on Adaptive Processes and A Special Symposium on Fuzzy Set Theory and Applications*, volume 16, pages 620–625, 1977.

[101] Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford et Michael L. Littman : PAC Model-Free Reinforcement Learning. *In 23rd International Conference on Machine Learning (ICML 2006)*, pages 881–888, Pittsburgh, PA, USA, 2006.

[102] Alexander L. Strehl et Michael L. Littman : An Empirical Evaluation of Interval Estimation for Markov Decision Processes. *In 16th IEEE International on Tools with Artificial Intelligence Conference (ICTAI 2004)*, pages 128–135, Boca Raton, FL, USA, 2004.

[103] Alexander L. Strehl et Michael L. Littman : An Analysis of Model-Based Interval Estimation for Markov Decision Processes. *Journal of Computer and System Sciences*, 2006.

[104] Malcolm Strens : A Bayesian Framework for Reinforcement Learning. *In Proceedings of the 17th International Conference on Machine Learning*, pages 943–950. Morgan Kaufmann, San Francisco, CA, 2000.

[105] Richard S. Sutton et Andrew G. Barto : *Reinforcement Learning : An Introduction (Adaptive Computation and Machine Learning).* The MIT Press, 3rd édition, March 1998.

[106] Richard S. Sutton, Anna Koop et David Silver : On the role of tracking in stationary environments. *In Proceedings of the 24th international conference on Machine learning (ICML 07)*, pages 871–878, New York, NY, USA, 2007. ACM.

[107] Richard S. Sutton, David A. McAllester, Satinder P. Singh et Yishay Mansour : Policy Gradient Methods for Reinforcement Learning with Function Approximation. *In Neural Information Processing Systems (NIPS)*, pages 1057–1063, 1999.

[108] Richard S. Sutton, Doina Precup et Satinder P. Singh : Between MDPs and semi-MDPs : A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[109] Peter Sykacek et Stephen Roberts : Adaptive Classification by Variational Kalman Filtering. *In Neural Information Processing Systems (NIPS 15)*, 2002.

[110] Torsten Söderström et Petre Stoica : Instrumental variable methods for system identification. *Circuits, Systems, and Signal Processing*, 21:1–9, 2002.

[111] Gerald Tesauro : Temporal Difference Learning and TD-Gammon. *Communications of the ACM*, 38(3), Mars 1995.

[112] William R. Thompson : On the likelihood that one unknown probability exceeds another in view of two samples. *Biometrika*, (25):285–294, 1933.

[113] Edward Thorndike : *Educational psychology : the psychology of learning.* Teachers College Press, New York, 1913.

[114] John N. Tsitsiklis et Benjamin Van Roy : An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42:674–690, 1997.

[115] Tsuyoshi Ueno, Motoaki Kawanabe, Takeshi Mori, Shin ichi Maeda et Shin Ishii : A Semiparametric Statistical Approach to Model-Free Policy Evaluation. *In International Conference on Machine Learning (ICML 08)*, Helsinki, Finland, 2008.

[116] Rudolph van der Merwe : *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models.* Thèse de doctorat, OGI School of Science & Engineering, Oregon Health & Science University, Portland, OR, USA, April 2004.

[117] Rudolph van der Merwe, Nando de Freitas, Arnaud Doucet et Eric Wan : The Unscented Particle Filter. Rapport technique CUED/F-INFENG/TR380, Cambridge University Engineering Department, août 2000.

[118] Vladimir Naumovich Vapnik : *Statisical Learning Theory.* John Wiley & Sons, Inc., 1998.

[119] Eric A. Wan et Rudolph Van Der Merwe : The unscented Kalman filter for nonlinear estimation. *In Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158, 2000.

[120] Tao Wang, Daniel Lizotte, Michael Bowling et Dale Schuurmans : Bayesian sparse sampling for on-line reward optimization. *In Proceedings of the 22nd international conference on Machine learning (ICML 05)*, pages 956–963, New York, NY, USA, 2005. ACM.

[121] C. Watkins : *Learning from Delayed Rewards*. Thèse de doctorat, Cambridge University, Cambridge, England, 1989.

[122] Marco Wiering et Jurgen Schmidhuber : Efficient model-based exploration. *In Proceedings of the fifth international conference on simulation of adaptive behavior on From animals to animats 5*, pages 223–228, Cambridge, MA, USA, 1998. MIT Press.

[123] Marco Wiering et Hado van Hasselt : The QV Family Compared to Other Reinforcement Learning Algorithms. *In IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2009)*, page 8 pages, Nashville, TN, USA, April 2009.

[124] Ronald J. Williams : Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, (8):229–256, 1992.

[125] Huizhen Yu et Dimitri P. Bertsekas : Basis Function Adaptation Methods for Cost Approximation in MDP. *In Proceedings of IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2009)*, Nashville, TN, USA, 2009.

[126] Karl J. Åström : Optimal control of markov processes with incomplete state information. *Journal of Mahematical Analysis and Application*, 10:174–205, 1965.