Optimisation de politiques séquentielles d'emploi et de maintenance prédictive de systèmes multi-composants

Tiffany Cherchi

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITE DE MONTPELLIER

En Biostatistiques

École doctorale : Information, Structures, Systèmes

Unité de recherche : Institut Montpelliérain Alexander Grothendieck

Optimisation de politiques séquentielles d'emploi et de maintenance prédictive de systèmes multi-composants

Présentée par Tiffany Cherchi

Le 16/12/2021

Sous la direction de Benoîte de Saporta et François Dufour

Devant le jury composé de

Anne Barros	Professeure, CentraleSupélec	Rapportrice
Camille Baysse	Ingénieure de recherche, Thales Optronique	Co-encadrante
Benoîte de Saporta	Professeure, Université de Montpellier	Directrice
François Dufour	Professeur, Institut Polytechnique de Bordeaux	Co-directeur
Mitra Fouladirad	Professeure, Ecole Centrale Marseille	Présidente du jury
Antoine Grall	Professeur, Université de Technologie de Troyes	Rapporteur
Nadine Hilgert	Directrice de Recherche, INRAE Montpellier	Examinatrice

UNIVERSITÉ DE MONTPELLIER

Remerciements

Mes premiers remerciements vont à ma direction de thèse : Benoîte, Camille et François. Bien qu'une supervision à trois encadrant.es dans trois villes différentes soit complexe, vous avez toujours fait en sorte de vous rendre disponibles pour nos réunions, tout en composant avec mes récurrentes variations d'optimisme, entre autres difficultés rencontrées. Merci !

Je tiens ensuite à exprimer ma gratitude à Anne Barros et à Antoine Grall d'avoir accepté de rapporter mon manuscrit. Vos commentaires très minutieux et si encourageants m'ont beaucoup aidé à enrichir ma présentation ! Merci aussi à Mitra Fouladirad et à Nadine Hilgert de m'avoir fait l'honneur d'accorder du temps à mes travaux et d'assister à ma soutenance. Je remercie l'ensemble de mon jury pour leurs éloges qui donnent une si belle fin à ma thèse.

Côté DM/IMAG : Merci Laurence de m'avoir aidée pour tout depuis mon master, puis pour mes vacations, et enfin juste pour m'encourager et garder de mes nouvelles. Merci à Bernadette, Brigitte, Carmella, Nathalie et Sophie pour votre support toujours enthousiaste.
Côté Thales : Merci à Camille de m'avoir intégrée à l'équipe qui m'a accueillie à chaque visite, et à Félicie d'avoir toujours veillé sur moi pour que mes missions se passent au mieux.

Mes co-bureaux : La bonne ambiance a contribué pour beaucoup à terminer cette thèse ! Merci May pour ton accueil si chaleureux dès mon stage, qui m'aura aidé toute ma thèse. Merci Paul-Marie pour tes précieux conseils, et notre collaboration sur les enseignements. Merci à Nicolas et Pelle pour l'invention du fameux *Leon Ball*, les défis et *espoirs* au tableau. Merci enfin à Hanine et Guillaume pour nos discussions, rires, et nos parties de Badminton.
Mes camarades (post-)doctorant.es : je vous dois énormément pour ces belles années ! Merci à Maud pour l'organisation de nos *Journées Doctorantes en Probabilités*, motrices de la création d'une *Commission Parité*, Gautier pour nos éssais de permettre aux thésard.es de manger ensemble au RA quel que soit leur statut, Abel&Paul pour la passation du *SemDoc*. Faustine, je te remercie pour l'incroyable exploit d'avoir fait du lundi mon jour préféré pendant 3 ans, et notre semaine inoubliable passée ensemble à se soutenir aux JDS Nancy. Thiziri, je crois qu'il a suffit d'une seule seconde pour qu'on devienne amies et veuille le meilleur l'une pour l'autre : tu me manques déjà, j'ai hâte de lever ma pancarte à ta thèse. Merci ensuite à Thibo et Morgane d'être passé.es outre mon accueil un peu maladroit pour me laisser de si bons souvenirs, dont l'invention grandiose (mais modeste) du dictionnaire ! Morgane, parce qu'avoir la classe en *pyjama* tout en adorant *des musiques que personne n'aime* n'est vraiment pas donné à tout le monde, tu mérites bien une deuxième citation. Merci à Alan pour les billards et notre amour partagé de B2o et des retouches de photos ! Merci à Julien d'être fervent défenseur des Stats, pour les randos et cinés avec Maelli et Flo ! Merci à Tanguy d'avoir vite compris mes plaintes sur les aspects numériques de la thèse, pour ton soutien si souvent renouvelé jusqu'à la fin, et notre amour commun de Grey's Anatomy.

Merci enfin à Radia pour ta force, Tom pour ton humour, Matthieu pour ta gentillesse, Zaineb pour notre semaine aux JdS et surtout notre soutien mutuel pour nos soutenances, Juliette pour la poussière de fée, les gâteaux et les encouragements choupinets, Aurélio pour tes conseils culinaires et l'Amaretto, Pablo pour le goût des punchlines et les jets végans !

L'équipe EPS : Merci à Ali, Lionnel, Joseph et Véronique pour le plaisir pris à enseigner avec vous, puis à Élodie de nous avoir permis d'encadrer plusieurs projets de M1 avec Florent ! Merci à Ghislain et François-David pour les conseils techniques sur l'utilisation de MESO. Merci à Gilles, Jean-Noël et Joseph pour votre oeil attentif sur le déroulement de ma thèse. Merci à Benjamin pour les conseils experts en vin rosé et le qualificatif de *chiante pertinante*. Merci à André, Ghislain, Gwladys, Paul et Xavier d'avoir toujours été si agréables à croiser. À Jean-Michel Marin, je saisis l'opportunité de déclarer mon admiration, si mal cachée depuis longtemps : Merci d'être un modèle inspirant de professeur, de chercheur et de responsabilités prises dans la communauté, tout en restant très attentif, patient et encourageant avec moi. Merci enfin à Joseph de m'avoir régulièrement encouragée à terminer ma thèse : ton soutien et le ruissellement de tout ce que tu apportes numériquement, scientifiquement, et humainement à la thèse de Florent auront été décisifs pour mes recherches, Merci infiniment !

La diffusion à l'IREM : Merci pour votre humanité et ces occasions de *nettoyer mon âme* ! Merci Anne de m'avoir fait confiance pour l'organisation de MATh.en.JEANS, et d'être restée à l'écoute toutes ces années, jusqu'à m'offrir le meilleur cadre possible pour ma rédaction ! Merci Monia pour ton accueil chaleureux et toutes nos discussions : tu es un modèle pour moi et je ne sais vraiment pas comment te remercier pour ton soutien qui m'a été très précieux. Merci Simon et Viviane d'avoir aussi souvent pris le temps de discuter, et en particulier d'avoir été aussi moteurs de la création de la commission parité, projet qui me tenait à cœur. Merci Nicolas d'avoir toujours valorisé mon dynamisme et atypisme : tu as été le repère stable et encourageant dont j'avais besoin en arrivant à la FdS et tu resteras ma personne préférée bien après ma thèse. Pour toutes les actions que tu organises pour les jeunes/maths et auxquelles tu m'as permis de participer, mon admiration et ma gratitude sont infinies.

Sur les épaules des géants : si j'ai pu me hisser jusqu'ici, c'est uniquement grâce à eux. Alain, je te remercie de ton mon cœur d'avoir un jour pris le temps d'écouter mes drames et de n'avoir jamais arrêté de me soutenir ensuite. Claire, merci d'avoir rejoint cette *drôle d'adoption* et m'avoir offert nos soirées au théâtre dont je garde de si précieux souvenirs. Merci enfin à M. El mannany pour cette main tendue qui me porte encore tant aujourd'hui.

Le TRIO(mphe). Je sais la chance incroyable que j'ai d'avoir des femmes aussi extraordinaires que vous dans ma vie. Merci pour cette décénie d'amitié qui survit à la distance, vos messages, plans et vocaux qui égaient mes journées, et surtout de vous être rendues dispo pour m'héberger et me rejoindre à chaque fois que j'étais en mission à l'autre boût de Paris !

Je te dédie le seul *nous* de ma thèse : je ne peux m'empêcher d'employer mon amour des mots pour exprimer celui que je te porte. Merci de m'avoir encouragée à me remettre au Python pour que Numba sauve ma thèse, d'avoir trépigné d'impatience avec moi pour chaque nouvelle expérience numérique, exulté avec moi pour chaque pourcentage d'accélération gagné et chaque étape validée, et enfin rappelé sans cesse ton intérêt pour l'application et les interprétations de mes recherches. Merci enfin pour ton investissement total dans la préparation de ma soutenance et du pot, juste après les nombreuses relectures du manuscrit que tu as subi. Sans toi, sans la force et l'abnégation incroyables que tu as montré chaque jour depuis que tu es dans ma vie et dans cette thèse, rien n'aurait été possible.

« J'aime l'idée que peut-être un jour quand vous serez [une vieille personne] comme moi, vous rencontrerez [une jeune personne] comme vous et vous lui direz comment un jour vous avez pris le citron le plus acide, le plus amer que la vie ait produit, pour en faire quelque chose qui ressemble à de la limonade.»

Dr. K. - This Is Us.

Titre. Optimisation de politiques séquentielles d'emploi et de maintenance prédictive de systèmes multi-composants

Résumé. On présente un problème d'optimisation pour la maintenance d'un système multi-composants conçu par Thales. Ce dernier est sujet à des détériorations et défaillances aléatoires de ses composants, au cours de missions pour lesquelles il est requis, entraînant l'évolution de son état et une pénalité d'indisponibilité en cas d'échec. L'enjeu est alors de définir une politique optimale d'emploi et de maintenance du système, afin de garantir le bon déroulement des missions, tout en minimisant ses coûts de gestion. Il s'agit de déterminer un compromis entre intervenir trop tôt, générant des coûts de maintenances inutiles, et intervenir trop tard, menant à la panne du système et à payer des pénalités et des opérations plus coûteuses.

Une des spécificités de ce travail est de considérer une prise de décision séquentielle sur le système. Ensuite, il s'agit de différencier les opérations de maintenance selon l'état de chacun de ses composants. L'idée principale de ce travail est alors de proposer un modèle mathématique pour l'évolution du système via le formalisme d'un Processus Markovien Décisionnel (MDP). Ainsi, l'objectif est de résoudre le problème d'optimisation associé, à savoir déterminer pour chaque date de décision et chaque état du système, l'action qui minimise la somme des coûts générés sur tout l'horizon. C'est ce qu'on appelle une politique. On propose ensuite plusieurs politiques de référence préventives et correctives et on compare leurs performances en termes de coûts et de statistiques de pannes, par simulations de Monte-Carlo. Ceci illustre l'intérêt de regrouper les maintenances lors des passages en atelier, et de considérer les temps de fonctionnement des composants pour les prises de décision, afin de réduire à la fois les coûts et le taux de pannes.

Le modèle spécifié pour cette problématique industrielle donne lieu à un problème d'optimisation non standard dans le cadre des MDP, car l'espace d'états du système est continu et le noyau de transition n'est pas explicite analytiquement, mais seulement simulable. Afin de pouvoir rechercher une politique optimale sur un ensemble fini de politiques admissibles, on discrétise la règle de décision du MDP. Ceci permet de prendre des décisions sur un nombre fini d'états, sans discrétiser la dynamique du MDP. Les coûts des politiques de référence sont utilisés pour calibrer cette discrétisation. Il s'agit de déterminer un compromis entre précision, conduisant à considérer un très grand nombre d'états, et une complexité numérique, conduisant à un nombre d'états le plus petit possible. Enfin, le noyau de transition n'étant toujours pas explicite, on implémente et on compare deux méthodes d'optimisation stochastiques basées sur les simulations afin d'approcher et d'expliciter une politique optimale. Une attention particulière est portée à l'identification de l'origine des gains, afin d'interpréter la politique optimale déterminée.

Mots-clés. Modélisation, Simulation, Processus Markovien décisionnel, Optimisation, Politique séquentielle, Maintenance prédictive, Système multi-composants.

Title. Optimization of sequential employment and predictive maintenance policies for multi-component systems

Abstract. We present an optimization problem for the maintenance of a multi-component system designed by Thales. This system is subject to random deteriorations and failures of its components, during the missions for which it is required, leading to the evolution of its state and an unavailability penalty in case of failure. The challenge is then to define an optimal policy for the use and maintenance of the system, in order to guarantee the good progress of the missions, while minimizing its management costs. It is a question of determining a compromise between intervening too early, generating unnecessary maintenance costs, and intervening too late, leading to the failure of the system and to paying penalties and more expensive operations.

One of the specificities of this work is to consider a sequential decision making on the system. Then, it is about differentiating the maintenance operations according to the state of each of its components. The main idea of this work is then to define a mathematical model for the evolution of the system via the formalism of a Markovian Decision Process (MDP). Thus, the objective is to solve the associated optimization problem, i.e. to determine for each decision date and each state of the system, the action that minimizes the sum of the costs generated over the whole horizon. This is called a policy. We define several preventive and corrective reference policies and compare their performances in terms of costs and failure statistics, by Monte-Carlo simulations. This illustrates the interest of grouping the maintenances during the workshop visits, and of considering the operating times of the components for decision making, in order to reduce both the costs and the failure rate.

The model specified for this industrial problem gives rise to a non-standard optimization problem in the MDP framework, because the system state space is continuous and the transition kernel is not analytically explicit, but only simulatable. In order to be able to search for an optimal policy over a finite set of admissible policies, we discretize the MDP decision rule. This allows decisions to be made over a finite number of states, without discretizing the MDP dynamics. The costs of the reference policies are used to calibrate this discretization. The goal is to determine a compromise between precision, leading to consider a very large number of states, and numerical complexity, leading to the smallest possible number of states. Finally, as the transition kernel is still not explicit, we implement and compare two stochastic optimization methods based on simulations in order to approach and make explicit an optimal policy. Particular attention is given to identifying the origin of the benefits, in order to interpret the determined optimal policy.

Keywords. Modeling, Simulation, Optimization, Markov decision process, Sequential policy, predictive maintenance Multi-component system.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Choisir parmi plusieurs possibilités dont les conséquences sont incertaines est un défi rencontré dans de nombreuses situations. Qu'il s'agisse de gérer un stock, guider un robot dans l'accomplissement d'une tâche, ou encore planifier une stratégie de maintenance, un des enjeux est de pouvoir mesurer les conséquences des décisions prises. Mais bien souvent, il ne suffit pas de prendre une décision isolée à un instant et une situation donnés, pour lesquels on observe les effets immédiats.

Si on prend l'exemple d'un système en exploitation, les décisions se prennent souvent de manière séquentielle : chaque jour, chaque semaine ou chaque année par exemple, et peuvent être des choix de maintenance ou d'emploi du système. Ces décisions ont des conséquences incertaines à court et à long termes. L'envoi en mission peut exposer à court terme à des dégradations et défaillances aléatoires du système, mais un entretien périodique trop fréquent peut aussi engendrer des coûts inutiles compte tenu de sa durée de vie et de son utilisation. Ainsi, ce que l'on cherche, c'est la suite d'actions qui minimise le coût cumulé sur tout un horizon, et non les conséquences d'une décision unique, c'est ce qu'on appelle une politique. Tout l'enjeu est alors de proposer un modèle représentant à la fois l'évolution du système et les conséquences des décisions, puis de déterminer la meilleure suite de décisions à prendre.

Les processus markoviens décisionels (MDP), introduits par [Bel57b], proposent un cadre qui pose formellement ce problème d'optimisation, en définissant les états et les actions possibles pour le système, ainsi que l'évolution et les coûts engendrés. Outre leur étude théorique [BH61, GSP19, WXL+17, ADPR16], les MDP ont de très nombreuses applications dans des domaines variés, tels que la robotique [CTKL13], le dialogue artificiel [LPE98, YGTW13] l'agroécologie [TPA+13] ou encore le médical [HPvB+10, KRL16, ARC18, SBSR04].

Dans ce travail, on propose d'utiliser le formalisme des MDP pour développer un modèle décrivant l'évolution d'un système industriel conçu et commercialisé par Thales. Le pod de désignation laser est un équipement optronique aéroporté peuvant être requis pour des missions. L'objectif est de déterminer une politique minimisant les coûts de sa maintenance.

Contexte industriel et objectifs. Pour une entreprise telle que Thales, il est primordial d'anticiper les pannes de ses équipements, pour garantir leur disponibilité auprès de ses clients tout en minimisant les coûts de maintenance. Thales est un groupe français spécialisé dans l'électronique pour les secteurs de la défense, l'aéronautique, l'espace, le transport et la sécurité. C'est aujourd'hui l'un des leaders mondiaux des équipements à destination des industries de l'aéronautique et de la défense. Aussi, ce travail se place dans la lignée de la politique d'investissement lancée par l'entreprise pour optimiser le rapport disponibilité – coût de ses équipements, en développant des outils de modélisation pour *prédire* les pannes. Ainsi, à l'issue des travaux de [Bay13, Gee17] sur l'optimisation d'instants puis de décisions de maintenance pour des caméras thermiques, Thales a souhaité changer d'équipement et d'échelle dans la problématique d'optimisation de la maintenance de ses équipements. En effet, les méthodes développées précédemment et basées sur des modèles de processus markovien déterministes par morceaux (PDMP) souffrent du fléau de la dimension et ne permettent pas de passer à l'échelle lorsque le nombre de décisions, le nombre de cycles de vie ou encore le nombre de composants augmentent. Développer d'autres modèles mathématiques pour Thales s'impose.

Le choix des MDP a été fait pour permettre un passage à l'échelle palliant ces limitations. Une des spécificités de ce travail est alors de considérer une prise de décision séquentielle sur le système considéré, plutôt qu'une prise de décision impulsionnelle, pour les PDMP. Ensuite, il s'agit de différencier les opérations pour chaque composant du système multi-composant. L'objectif de la thèse est de développer un modèle mathématique pour l'évolution du pod, et de construire une politique de maintenance optimisée pour assurer le bon déroulement des missions tout en minimisant les coûts de maintenance.

À ces dates de décisions, il faut décider si l'état du système lui permet effectuer sa mission ou s'il doit subir des entretiens ou des remplacements d'un ou plusieurs composants, mais également déterminer les actions de maintenance optimales à effectuer. Ces décisions peuvent avoir un impact sur les coûts de maintenance mais également sur la disponibilité du système. En effet, si une maintenance effectuée trop tôt peut réduire la disponibilité et être inutilement coûteuse, une maintenance trop tardive peut mener à la panne totale du système, générant une indisponibilité et des coûts de maintenance importants. L'enjeu est alors de trouver un compromis adapté à l'utilisation des équipements pour maximiser la disponibilité et minimiser les coûts de maintenance. En pratique, on souhaite donc construire une politique de maintenance qui indique à chaque date de décision et à chaque état du système, l'action optimale à effectuer.

Ce chapitre a plus particulièrement pour objectif de présenter les notions générales de maintenance et quelques uns de ses principaux aspects traités dans la littérature. Ceci permet de préciser le contexte de ce travail, les aspects de modélisation de la dynamique du système et l'optimisation de sa stratégie de maintenance.

Définition de la maintenance. Afin d'identifier les enjeux de la maintenance, on en présente la terminologie en reprenant la définition de la norme européenne AFNOR NF EN 13306. D'après cette norme, la maintenance est définie par *« l'ensemble de toutes les actions techniques, administratives et de management durant le cycle de vie d'un bien, destinées à le maintenir ou à le rétablir dans un état dans lequel il peut accomplir la fonction requise »*.

Dans cette définition, les termes *« maintenir »* et *« rétablir »* permettent de distinguer deux catégories de maintenance. Dans le premier cas, il est question d'effectuer des actions de prévention, sur un système encore en fonctionnement : on parle de maintenance préventive. Dans le second, il s'agit de corriger une défaillance sur un système qui n'est plus en mesure d'assurer sa fonction, on parle de maintenance corrective.

Maintenance corrective. La *maintenance corrective* est une *« maintenance exécutée après détection d'une panne et destinée à remettre un bien dans un état dans lequel il peut accomplir une fonction requise. »* (AFNOR, 2001). Il en existe deux catégories. La première, appelée maintenance palliative, permet d'effectuer une maintenance provisoire restaurant tout ou partie des fonctions requises. La deuxième, appelée maintenance curative, a pour objectif de restaurer toutes les fonctions requises du système. Ces types de maintenance engendrent généralement des surcoûts [LND+06, LBB+15] et une indisponibilité [Pal13].

Apport des HUMS. Quel que soit leur secteur d'activité, les progrès dans différents domaines du numérique permettent d'envisager l'automatisation de tâches de plus en plus complexes, et notamment les prises de décision de maintenance pour les systèmes équipés de HUMS (Health and Usage Monitoring Systems) [oD04, WPC+18, CC18]. En relevant et traitant des données sur un équipement, ils permettent d'évaluer son état de santé (diagnostic) et de déterminer son potentiel restant (pronostic), afin d'anticiper les pannes et de proposer une date et une décision de maintenance adaptée. Aussi, suite aux travaux de [Bay13] sur la détection d'état dégradé pour les systèmes équipés de HUMS de Thales, on suppose que les composants du pod ont 3 états (stable, dégradé, panne) parfaitement observables.

Maintenance préventive. Si la maintenance était auparavant centrée sur la correction des défauts, elle a maintenant pour objectif d'anticiper les dysfonctionnements des équipements, c'est-à-dire d'intervenir avant que la panne ne se produise. On parle alors de maintenance préventive [BRASK17, KSO+21], et on en distingue trois types.

— La *maintenance systématique* est la *« maintenance préventive exécutée à des intervalles de temps préétablis ou selon un nombre défini d'unités d'usage mais sans contrôle préalable de l'état du bien. »* (AFNOR, 2001). L'objectif consiste typiquement à déterminer une fréquence fixe entre les interventions [ZFB17]. Elle s'applique préférentiellement aux systèmes dont la surveillance est impossible, les inspections et défaillances sont trop coûteuses, ou qui sont soumis à une règlementation sécuritaire.

— La *maintenance conditionnelle* correspond à la *« maintenance préventive basée sur une surveillance du fonctionnement du bien et/ou des paramètres significatifs de ce fonctionnement intégrant les actions qui en découlent »* (AFNOR, 2001). Elle s'applique aux systèmes dont on peut mesurer et surveiller des caractéristiques physiques susceptibles de révéler une dégradation. L'objectif est de déterminer un seuil à partir duquel intervenir. Elle permet de mieux tenir compte des conditions d'utilisation en surveillant des signes de dégradation du système [AX17, GBD02].

— La *maintenance prévisionnelle* est une *« maintenance conditionnelle exécutée en suivant les prévisions extrapolées de l'analyse et de l'évaluation de paramètres significatifs de la dégradation du bien »* (AFNOR, 2001). C'est une sous catégorie de maintenance conditionnelle, appelée aussi maintenance prédictive, qui nécessite de modéliser l'évolution de la dégradation afin d'estimer le temps d'utilisation restant jusqu'à la défaillance [Gee17, ZYW19, Bay13, ZCdRR+20]. En adaptant les décisions à l'évolution réelle de l'état du système, ce type de maintenance permet de proposer les opérations les plus adaptées à son utilisation.

Combinaisons de types de maintenances. Aussitôt qu'on s'intéresse à un système constitué de plusieurs composants, il est naturel de chercher à différencier les actions selon l'état et l'importance de chaque composant. Des modèles spécifiques ont été développés pour des systèmes multi-composant [CP91, NFD07, Wan02]. Cela induit deux types de maintenances, qui sont des combinaisons des précédentes : les maintenances mixtes et opportunistes.

— Une stratégie de *maintenance mixte* permet des actions aussi bien préventives que correctives. Elle consiste à intervenir de façon préventive, par exemple pour des composants dits "critiques", et d'attendre la panne pour d'autres.

— Concernant la *maintenance opportuniste*, l'idée est de regrouper les maintenances de plusieurs composants, par exemple en entretenant les composants dégradés lors d'un retour en atelier pour une panne ou dégradation d'un autre composant. Ce type de maintenance vise à réduire le nombre de passages à l'atelier, afin de diminuer les coûts fixes et les périodes d'indisponibilité associées.

La Figure 1.1 résume les différents types de maintenances [GMZ07]. Dans ce travail, il s'agit de déterminer à chaque date et chaque état du système une action différentiée à la fois selon l'état des composants, en regroupant, avançant ou repoussant les maintenances, et selon la date de décision.

Types de dépendances dans les systèmes. Lorsqu'on étudie un système à plusieurs composants, ou plusieurs systèmes, l'optimisation de la maintenance doit permettre de tenir compte des dépendances qui peuvent les lier. En particulier, on distingue trois types de dépendances, à savoir les dépendances *structurelles, stochastiques* et *économiques*.

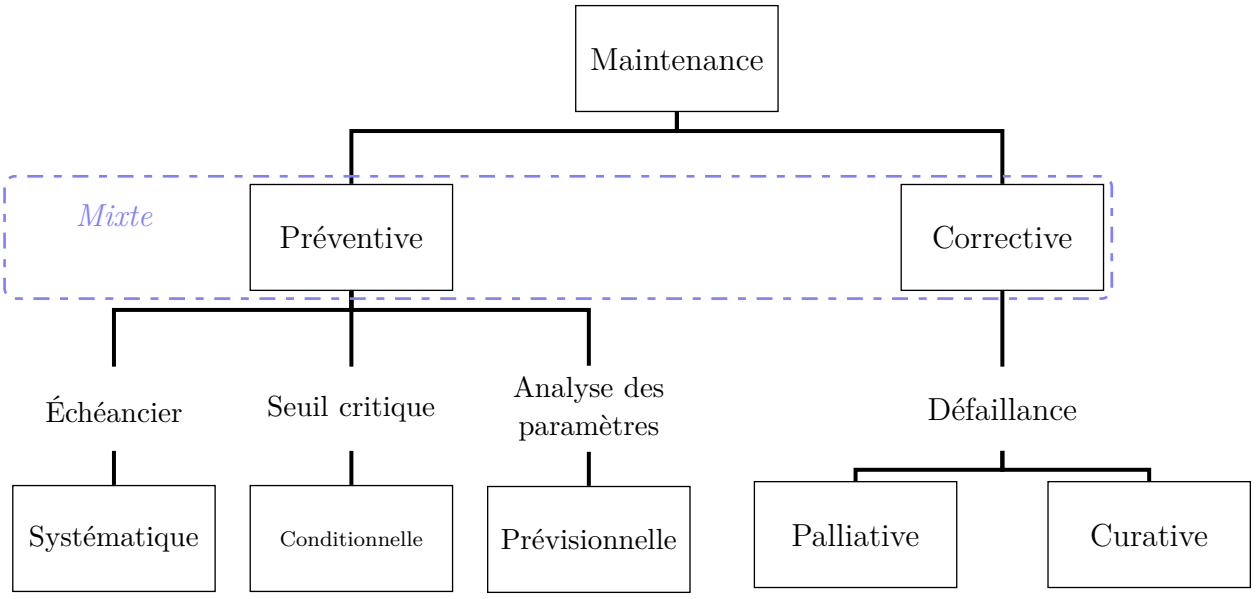


FIGURE 1.1 – Les différents types de maintenance.

— La *dépendance structurelle* se rapporte aux systèmes dont des sous-composants sont assemblés par bloc [Gee17, Bay13]. Dans le cas de défaillance d'un composant cette structure peut impliquer l'intervention sur tout le bloc.
— La *dépendance stochastique* concerne les systèmes dont l'état d'un ou de plusieurs composants peut affecter l'état d'autres composants. Par exemple, [BGB03, BBG06] ont étudié un système parallèle à deux composants et ont proposé une politique de maintenance préventive.
— La *dépendance économique* rend compte de l'influence des décisions de maintenance sur les coûts. Ce type de dépendance peut concerner les systèmes dont la structure se présente sous forme de blocs, qui sont stochastiquement liés, on encore qui partagent des frais d'immobilisation. Dans ce contexte, le regroupement de maintenance peut être très adapté pour réduire les durées et coûts de maintenance [BABC11].

Dans ce travail, on étudie des composants structurellement et stochastiquement indépendants, mais qui présentent une dépendance économique liée aux coûts d'immobilisation du système en maintenance et à des pénalités d'indisponibilité à payer lorsque les missions ne sont pas réalisées.

Efficacité des opérations de maintenance. La troisième notion introduite par la définition issue de la norme AFNOR, en évoquant *« un état dans lequel il peut accomplir la fonction requise »*, fait référence à l'état du système à la suite d'une opération de maintenance. Le type d'action effectué sur un système détermine son état en sortant de maintenance, en fonction du niveau de restauration de ses fonctions. On distingue trois types d'actions différentes, les maintenances *minimales, parfaites* et *imparfaites*.

— La *maintenance parfaite* correspond à une restauration du système qui lui permet de retrouver un état aussi bon que neuf (« As Good As New – AGAN »). Elle est équivalente à un remplacement du système par un nouveau, et implique un retour à l'état initial suite à l'intervention. Cette hypothèse est très courament utilisée dans la littérature [ND04, Wu11, Huy11, Lia16] car elle simplifie la modélisation.
— La *maintenance minimale* restaure le système dans l'état qui précède la défaillance, pour le rendre temporairement opérationnel. L'état de l'entité après intervention est supposé aussi mauvais que vieux (« As Bad As Old – ABAO ») [BP83]. Une synthèse des travaux sur la réparation minimale est disponible dans [AKOTY11].
— La *maintenance imparfaite* ne restaure pas le système à l'état neuf mais à un état intermédiaire entre la maintenance minimale et la maintenance parfaite. Cette maintenance vise à améliorer l'état de l'entité en réduisant par exemple son niveau de dégradation mais sans la remettre à l'état neuf [Kij89, NFD07, DG04, MC19]

Dans ce travail, on suppose que les opérations de maintenance faites sur tous les composants du pod sont des maintenances parfaites.

Les familles de politiques. Concernant la maintenance de systèmes industriels, il existe une littérature importante et basée sur des modèles aléatoires [CT97, CMBZ18, THDV+17]. Deux familles de problèmes sont principalement étudiées. La première famille concerne l'optimisation de familles spécifiques de politiques paramétrées. Par exemple, on peut vouloir optimiser la fréquence d'une maintenance systématique, voir par exemple [ZFB17], ou un seuil critique pour déclencher une maintenance conditionnelle, voir par exemple [MC19].
La deuxième famille, dans laquelle s'inscrit l'objectif de ce travail, concerne l'optimisation globale de toute une classe de règles de maintenance non paramétriques. Dans les travaux de [dSDZE12, Bay13], la classe de stratégies de maintenance qu'ils considèrent sont les stratégies dans lesquelles une seule intervention peut avoir lieu et la structure est ensuite remplacée par une nouvelle. Dans notre travail, on considère un problème plus général car on autorise plusieurs interventions sur le même équipement sans forcément le remplacer par un nouveau.

Ces décisions ne sont pas simples à optimiser, compte tenu de la complexité qui apparaît dès lors qu'on traite des systèmes multi-composants, pour lesquels on considère plusieurs états avant la panne, et plusieurs dates de décisions. Proposer des stratégies dans une approche d'évaluation et de comparaison ne permet pas d'explorer l'ensemble des politiques possibles. De plus, dans ce contexte, le nombre de stratégies possibles peut devenir rapidement trop grand pour être construit et analysé *manuellement*. C'est pourquoi il est nécessaire de proposer des outils mathématiques d'optimisation pour aider à répondre à ces questions. Tout l'enjeu est alors de proposer un modèle mathématique permettant de modéliser d'une part l'évolution d'un tel système multi-composant et d'autre part d'employer des méthodes d'optimisations afin de déterminer une politique minimisant les coûts de maintenance.

Organisation du manuscrit

Ce rapport de thèse est organisé comme suit. Dans le Chapitre 2, on définit les processus Markoviens décisionnels (MDP), ainsi que plusieurs techniques exactes ou approchées de résolution des MDP à espace d'états et d'actions finis. On décrit la modélisation d'un équipement monolithique dont l'état est caractérisé par une variable discrète et dont les transitions, selon un nombre fini d'actions possibles, sont également discrètes, par un MDP. On détaille ensuite sa résolution, ce qui permet d'illustrer l'ensemble des outils introduits. Ensuite, le Chapitre 3 est consacré à la description du système industriel étudié, et à la modélisation de sa dynamique par un processus markovien décisionnel à espace d'état continu. On introduit plusieurs politiques de maintenance correctives ou préventives et on compare leurs performances en termes de coûts ainsi que les statistiques de pannes des trajectoires. Enfin, dans le Chapitre 4, afin d'approcher la fonction valeur du MDP, via des méthodes d'optimisation stochastiques basées sur la simulation de trajectoires contrôlées du processus, on explore différents sous-ensembles finis de l'ensemble des politiques admissibles associé. Pour finir, le Chapitre 5 présente une conclusion et des perspectives à ce travail.

Chapitre 2

Processus Markoviens décisionnels

Les processus markoviens décisionnels (MDP) sont des processus stochastiques contrôlés qui modélisent des problèmes de prises de décision séquentielle dans l'incertain. Ils permettent de décrire l'évolution de l'état d'un système sujet à des prises de décision successives d'un·e agent. Ce choix d'action génère une transition du système vers un nouvel état, qui dépend à la fois de son état courant et de la décision prise. Ce couple engendre également un coût, appelé coût par transition, qui permet de modéliser la satisfaction vis-à-vis de l'état du système et de l'action choisie. Les décisions d'un·e agent doivent l'amener à réaliser un objectif sur tout un horizon d'étude, mais les conséquences des décisions ne sont pas nécessairement connues à l'avance. La première caractéristique de ce type de problèmes est donc liée à la nature stochastique de l'environnement, aux effets incertains des décisions possibles. Leur seconde caractéristique s'établit dans la durée, car ce n'est pas un, mais plusieurs problèmes de décision qu'un·e agent doit résoudre. Si les effets immédiats peuvent alors être facilement observables, les conséquences à long terme peuvent être plus difficiles à prendre en compte au moment de décider.

Les problèmes associés aux MDP consistent alors à trouver pour chaque date de décision la meilleure action à effectuer selon l'état du système, c'est-à-dire celle qui réalise le meilleur compromis entre les coûts immédiats et les pertes futures, pour minimiser le coût en moyenne. Une solution de ce problème d'optimisation est finalement une règle qui indique l'action à choisir pour chaque état possible du système et pour chaque instant de décision, c'est ce qu'on appelle une politique. Les MDP proposent un cadre qui pose formellement ce problème d'optimisation, pour un critère donné, en définissant les états et les actions possibles pour le système, ainsi que les transitions et les coûts engendrés. Plusieurs critères de coût sont étudiés dans la littérature, dont les deux plus courants se distinguent selon que l'horizon du MDP soit fini ou infini.

Dans la section 2.1, on décrit le formalisme d'un processus markovien décisionnel en se basant sur [BR11, HLL96, HLL99, Put94]. On définit les MDP dans la sous-section 2.1.1, puis on introduit la notion de politique et la propriété de Markov pour les MDP dans la sous-section 2.1.2. Ensuite, on décrit les critères de coûts usuels dans la sous-section 2.1.3, et on présente plusieurs méthodes de résolution de MDP dans la section 2.2. Enfin, on présente la quantification dans la section 2.3.

2.1 Formalisme d'un MDP

2.1.1 Définition d'un Processus markovien décisionnel

Définition 2.1.1. Un modèle markovien décisionnel est défini par un tuple :

$$(\mathbb{X}, \mathbb{A}, \{\mathbb{A}(x)|x \in \mathbb{X}\}, Q, c, C), \tag{2.1}$$

où :

- $\mathbb{X}$ est un sous-ensemble de $\mathbb{R}^d$ muni de la tribu des boréliens, appelé *espace d'états* ;
- $\mathbb{A}$ est un ensemble fini muni de la tribu de ses parties, appelé *espace d'actions*, qui permettent d'agir sur le système ;
- $\{\mathbb{A}(x)|x \in \mathbb{X}\}$ est un ensemble de sous-ensembles non vides de $\mathbb{A}$, où $\mathbb{A}(x)$ est l'*ensemble des actions possibles* lorsque le système est dans l'état $x \in \mathbb{X}$, tel que $\mathbb{K} = \{(x, a)|x \in \mathbb{X}, a \in \mathbb{A}(x)\}$ soit un sous-ensemble mesurable de $\mathbb{X} \times \mathbb{A}$;
- $Q(dy|x, a)$ est un noyau stochastique qui définit les transitions aléatoires du système lorsqu'il se trouve dans l'état $x \in \mathbb{X}$ et que l'action $a \in \mathbb{A}$ est choisie ;
- $c : \mathbb{K} \to \mathbb{R}$ est une fonction appelée *fonction de coût par transition*, associée au fait d'avoir choisi l'action $a \in \mathbb{A}$ lorsque le système est dans l'état $x \in \mathbb{X}$;
- $C : \mathbb{X} \to \mathbb{R}$ est la fonction de *coût terminal*.

Remarque 2.1.1. Selon le problème étudié, il peut être plus adapté de considérer une fonction de récompense au lieu d'une fonction de coût. Dans ce cas, l'objectif sera de maximiser une récompense au lieu de minimiser un coût.

Par la suite, on note $\mathcal{T}$ l'espace des dates de décisions. L'espace $\mathcal{T}$ est un ensemble discret, qui peut être fini ($\mathcal{T} = \{0, 1, \dots, H\},\ H \in \mathbb{N}^*$) ou infini ($\mathcal{T} = \mathbb{N}$). On parle alors d'horizon fini ou d'horizon infini du processus. Dans le cas général, les espaces $\mathbb{X}$ et $\mathbb{A}$ et les fonctions de transition Q et de coûts c et C peuvent être fonction de la date de décision $t \in \mathcal{T}$. On se limite ici au cas classique où le modèle est stationnaire, c'est-à-dire que ces derniers sont constants tout au long du processus et ne dépendent donc pas de la date de décision.

Pour la suite de ce chapitre, on se donne un modèle markovien décisionnel (Équation 2.1). On note $X_t \in \mathbb{X}$ la variable aléatoire correspondant à l'état du système observé à la date t, et x_t sa réalisation. De même, on note $A_t \in \mathbb{A}$ la variable aléatoire correspondant à l'action qui est choisie à la date t, et a_t sa réalisation. Pour contrôler un MDP, on doit spécifier comment choisir les actions à chaque date de décision, c'est ce qu'on appelle une politique. Pour s'assurer que l'ensemble des politiques n'est pas vide, on suppose ce qui suit.

Hypothèse 2.1. Il existe une fonction mesurable $f : \mathbb{X} \to \mathbb{A}$ telle que $f(x) \in \mathbb{A}(x)$ pour tout $x \in \mathbb{X}$.

On peut alors définir pour tout $t \in \mathbb{N}$, l'ensemble H_t des *histoires admissibles* jusqu'à la date t par $H_0 = \mathbb{X}$, et

$$H_t = \mathbb{K} \times H_{t-1} = \mathbb{K}^t \times \mathbb{X}. \tag{2.2}$$

Un élément h_t de H_t, appelé *t-histoire admissible*, est un vecteur de la forme :

$$h_t = (x_0, a_0, \dots, x_{t-1}, a_{t-1}, x_t), \tag{2.3}$$

où pour tout $0 \leq i \leq t-1, (x_i, a_i) \in \mathbb{K}$ et $x_t \in \mathbb{X}$.

On peut ensuite donner la définition générale d'une politique.

Définition 2.1.2. Une politique est une suite $\pi = (\pi_t)_{t\in\mathbb{N}}$ de noyaux stochastiques sur $\mathbb{A}$ sachant H_t, tels que $\pi_t(\mathbb{A}(x_t)|h_t) = 1,\ \forall\ h_t = (x_0, a_0, \dots, x_t) \in H_t,\ \forall\ t \in \mathbb{N}$. L'ensemble de toutes les politiques, dit *ensemble des politiques admissibles*, est noté Π.

L'objectif associé à un MDP étant de trouver une politique qui minimise un certain critère de coût sur tout l'horizon, il faut constuire l'histoire du processus pour calculer ce coût.

Construction d'un MDP. On note $\Omega = (\mathbb{X}, \mathbb{A})^{\mathbb{N}}$ l'ensemble dont les éléments sont les suites $\omega = (x_0, a_0, x_1, a_1, \dots)$, avec $x_t \in \mathbb{X}$, $a_t \in \mathbb{A}, \forall t \in \mathbb{N}$. Notons que l'ensemble $\mathbb{K}^{\mathbb{N}}$ est inclus dans Ω. On note F la tribu produit associée. On définit deux suites de variables aléatoires (X_t) et (A_t) sur Ω telles que pour tout $\omega \in \Omega$ et pour tout $t \in \mathbb{N}$, on pose :

$$X_t(\omega) = x_t \text{ et } A_t(\omega) = a_t. \tag{2.4}$$

Soit $\pi = \{\pi_t\}_{t\in\mathbb{N}}$ une politique de contrôle arbitaire et ν une mesure de probabilité arbitraire sur $\mathbb{X}$, appelée *distribution initiale*. D'après le théorème de Ionescu-Tulcea (voir [HLL96], annexe C, théorème C.10) il existe une unique probabilité P_ν^π sur (Ω, F) telle que, pour tous $B \in \mathcal{B}(\mathbb{X})$ et $C \in \mathcal{B}(\mathbb{A})$,

$$\mathbb{P}_\nu^\pi(X_0 \in B) = \nu(B), \tag{2.5}$$
$$\mathbb{P}_\nu^\pi(A_t \in C \mid h_t) = \pi_t(C \mid h_t), \tag{2.6}$$
$$\mathbb{P}_\nu^\pi(X_{t+1} \in B \mid h_t, a_t) = Q(B \mid X_t, A_t). \tag{2.7}$$

L'Équation 2.5 définit comment déterminer l'état initial, noté X_0, étant donnée une *distribution initiale* ν. Ensuite, l'Équation 2.6 spécifie le choix d'une action $A_t \in C$ à la t-ième date de décision comme la probabilité selon le t-ième noyau stochastique π_t de choisir C conditionnellement à toute la t-histoire observée h_t. Enfin, l'Équation 2.7 définit la transition vers un prochain état X_{t+1}, conditionnellement à l'état courant X_t et à l'action choisie A_t. Les probabilités de transition caractérisent la dynamique de l'état du système, et on peut noter que, contrairement à l'Équation 2.6 qui dépend de toute la t-histoire observée h_t, le noyau de transition ne se base finalement que sur l'état courant X_t et l'action choisie A_t.

Remarque 2.1.2. Bien que l'Équation 2.7 ressemble à une condition de Markov, elle n'implique pas que le processus stochastique induit $(X_t)_{t\geq 0}$ soit lui-même markovien. Ceci s'explique par le fait que le processus dépend de la politique de choix des actions. Le processus $(X_t)_{t\geq 0}$ n'est généralement pas un processus markovien, mais on étudiera dans la section suivante une condition sur la politique π pour que le processus soit un processus markovien.

Par la suite, on note $\mathbb{E}_\nu^\pi$ l'espérance associée à $\mathbb{P}_\nu^\pi$. On donne maintenant la définition d'un processus markovien décisionnel.

Définition 2.1.3. Un processus markovien décisionnel à temps discret est le processus stochastique $(X_k)_{k\geq 0}$ défini sur l'espace probabilisé $(\Omega,\ F,\ \mathbb{P}_\nu^\pi)$.

Remarque 2.1.3. La loi du processus $(X_t)_{t\geq 0}$ dépend clairement de la politique π et de la distribution initiale ν choisies.

Construction d'un MDP en pratique. Soit $\pi = \{\pi_t\}_{t\in\mathbb{N}}$ une politique de contrôle arbitaire et ν une distribution initiale arbitaire. L'évolution de l'état du système, contrôlé par cette politique π et partant de cette distribution initiale ν, est déterminée de la façon suivante :

(0) on tire un état initial x_0 selon ν ;

(1) on observe l'état du système $x_0 \in \mathbb{X}$ et on choisit une action de contrôle $A_0 \in \mathbb{A}(x_0)$ avec la loi $\pi_0(\cdot\,|h_0)$;

(2) cela génère un coût $c(x_0, a_0)$;

(3) une transition aléatoire est alors induite pour le système vers un nouvel état $X_1 \in \mathbb{X}$ selon l'action choisie a_0 et la distribution de probabilité donnée par le noyau stochastique $Q(\cdot\,|x_0, a_0)$.

Cette procédure est ensuite répétée de manière itérative en repartant de (1), pour générer une trajectoire du processus en termes d'une suite d'états et d'actions (x_t, a_t).

La Figure 2.1 donne une représentation graphique de la dynamique d'un MDP à horizon fini sous forme d'un diagramme d'influence [Sha86]. À chaque date de décision $t \in \mathcal{T}$, l'action a_t est appliquée dans l'état courant x_t, ce qui influence le processus dans sa transition vers le prochain état x_{t+1} et produit un coût $c(x_t, a_t)$.

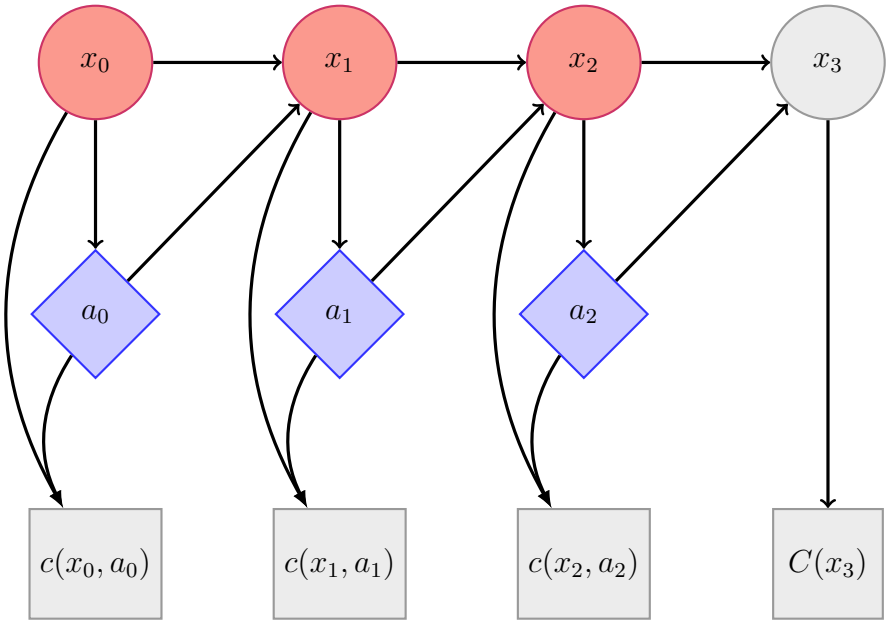


FIGURE 2.1 – Représentation de la dynamique d'un MDP sur un horizon fini $H = 3$. Les cercles correspondent à des états, les losanges à des décisions, et les rectangles à des coûts.

Exemple 1: Modèle d'un système monolithique à plusieurs états

Soit un système requis pour des missions quotidiennes, sur un horizon de 10 jours.

Espace d'états. Le système est sujet à des détériorations ou défaillances aléatoires, et peut être dans trois états possibles. On note : $\mathbb{X} = \{\text{stable}, \text{dégradé}, \text{panne}\}$.

Espace d'actions. En plus d'aller en mission, deux opérations sont possibles pour prolonger la durée de vie du système. Ces maintenances, supposées parfaites, sont soit un entretien, soit un remplacement. On note $\mathbb{A} = \{\text{mission}, \text{entretenir}, \text{remplacer}\}$.

Dans les états stable et dégradé, l'action d'entretien peut être faite , mais pas en cas de panne. Les actions mission et remplacer sont possibles dans tous les états.

Noyau de transition. Toutes les actions durent 1 jour, et l'état au jour $t+1$ dépend uniquement de l'état et de l'action du jour t. En mission, le système passe de l'état stable à dégradé selon une loi de Bernouilli $\mathcal{B}(0.1)$ puis de l'état dégradé à panne selon une loi de Bernouilli $\mathcal{B}(0.7)$. Les maintenances, supposées parfaites, ramènent le système en état stable. Les matrices de transitions sont données Table 2.1.

TABLE 2.1 – Détails des différentes matrices de transitions selon l'action choisie.

x \ x'	S	D	P
S	0.9	0.1	0
D	0	0.3	0.7
P	0	0	1

(a) Cas : a=mission.

x \ x'	S	D	P
S	1	0	0
D	1	0	0
P	0	0	1

(b) Cas : a=entretenir.

x \ x'	S	D	P
S	1	0	0
D	1	0	0
P	1	0	0

(c) Cas : a=remplacer.

Fonction de coût journalier. Le coût dépend de l'état et de l'action choisie. En plus du coût d'entretien et de remplacement, on définit une pénalité d'indisponibilité à payer si le système part en mission en panne. Les valeurs des coûts sont données Table 2.2a, et Table 2.2b résume les coûts associés aux couples (x, a). Le coût terminal est nul.

TABLE 2.2 – Valeurs des coûts des actions, et détails selon l'état courant.

Coût entretien	$c_e = 50$
Coût remplacement	$c_r = 200$
Pénalité indisponibilité	$c_u = 500$

(a) Valeurs des différents coûts.

x \ a	M	E	R
S	0	50	200
D	0	50	200
P	500	·	200

(b) Coût des couples (état, action).

2.1.2 Politiques et propriété de Markov

Selon la Définition 2.1.2, une politique est une suite de noyaux stochastiques telle que le choix d'une action a à l'instant t dépend de toute la t-histoire h_t observée du processus. Cependant, selon l'application étudiée, il peut être plus adapté de considérer des politiques déterminant des actions choisies comme des fonctions *déterministes* de la t-histoire h_t et/ou qui dépendent seulement de l'état courant X_t. D'après [HLL96], cette classe restreinte de politiques est importante, aussi bien du point de vue théorique que numérique, car plus simple à traiter, et la recherche de conditions pour l'existence de politiques *optimales* dans cette classe est un sujet central associé aux MDP. De ce fait, on présente maintenant plusieurs sous-familles de politiques. Tout d'abord, on introduit la définition suivante.

Définition 2.1.4. On définit d'abord l'ensemble de tous les noyaux stochastiques $\Phi = \{\varphi \in \mathcal{P}(\mathbb{A} \mid \mathbb{X}) \mid \varphi(\mathbb{A}(x) \mid x) = 1, \forall x \in \mathbb{X}\}$. On définit ensuite l'ensemble de toutes les fonctions mesurables $\mathbb{F} = \{f : \mathbb{X} \to \mathbb{A} \text{ mesurable} \mid f(x) \in \mathbb{A}(x), \forall x \in \mathbb{X}\}$. Les fonctions de $\mathbb{F}$ sont appelées sélecteurs de la fonction $x \mapsto \mathbb{A}(x)$.

Une fonction $f \in \mathbb{F}$ peut s'identifier à un noyau $\varphi \in \Phi$ avec l'égalité $\varphi(C \mid x) = \mathbb{1}_C[f(x)], \quad \forall\, x \in \mathbb{X}, \quad \forall\, C \in \mathcal{B}(\mathbb{A})$. On peut alors voir $\mathbb{F}$ comme un sous-ensemble de Φ.

On peut ensuite distinguer plusieurs sous-ensembles de politiques. On dit qu'une politique est *markovienne* si la sélection de l'action à exécuter ne dépend que de l'état courant du système et non de l'historique. À l'inverse, on appelle politique *dépendante de l'historique* toute politique dont la sélection de l'action dépend de tout l'historique, et elle est non markovienne dans ce cas. Aussi, une politique est *déterministe* si elle choisit toujours la même action pour le même état ou historique, quelque soit la date de décision. Enfin, une politique est dite *aléatoire* si la sélection de l'action dépend non seulement de l'état ou de l'historique, mais aussi d'une distribution de probabilités sur le choix de l'action à exécuter. Plus formellement, on définit les ensembles de politiques suivants.

Définition 2.1.5. Une politique $\pi = (\pi_t)_{t \geq 0}$ est dite :

— *markovienne aléatoire* s'il existe une suite (φ_t) d'éléments de Φ telle que

$$\pi_t(\cdot \mid h_t) = \varphi_t(\cdot \mid x_t) \quad \forall\, h_t \in H_t. \tag{2.8}$$

— *stationnaire (markovienne) aléatoire* s'il existe $\varphi \in \Phi$ tel que

$$\pi_t(\cdot \mid h_t) = \varphi(\cdot \mid x_t) \quad \forall\, h_t \in H_t. \tag{2.9}$$

Les ensembles de politiques aléatoires associés sont notés Π_{AM} et Π_{AMS}.

— *déterministe* s'il existe une suite (g_t) de fonctions mesurables $g_t : H_t \to \mathbb{A}$ telle que, $\forall\ h_t$ dans H_t, $g_t(h_t) \in \mathbb{A}(x_t)$ et

$$\pi_t(C \mid h_t) = \mathbb{1}_C(g_t(h_t)) \quad \forall\ C \in \mathcal{B}(\mathbb{A}). \tag{2.10}$$

— *déterministe markovienne* s'il existe une suite (f_t) de sélecteurs telle que

$$\pi_t(C \mid h_t) = \mathbb{1}_C(f_t(x_t)) \quad \forall\ C \in \mathcal{B}(\mathbb{A}). \tag{2.11}$$

— *déterministe (markovienne) stationnaire* s'il existe un sélecteur f tel que

$$\pi_t(C \mid h_t) = \mathbb{1}_C(f(x_t)) \quad \forall\ C \in \mathcal{B}(\mathbb{A}). \tag{2.12}$$

Les ensembles de politiques déterministes associés sont notés Π_D, Π_{DM} et Π_{DMS}.

Remarque 2.1.4. On a les inclusions $\Pi_{AMS} \subset \Pi_{AM} \subset \Pi$ et $\Pi_{DMS} \subset \Pi_{DM} \subset \Pi_D \subset \Pi$. Le fait que $\mathbb{F} \subset \Phi$ implique que $\Pi_{DMS} \subset \Pi_{AMS}$. Ainsi, l'Hypothèse 2.1 est suffisante pour que tous les ensembles de politiques définis (Π y compris) soient non vides.

Notations. Soient Φ et $\mathbb{F}$ les ensembles de la Définition 2.1.4 et c et Q le coût par transition et la loi de transition de la Définition 2.1.1. On définit pour tout $x \in \mathbb{X}$, $c(x, \varphi) = \int_{\mathbb{A}} c(x, a)\ \varphi(\mathrm{d}a \mid x)$ et $Q(\cdot \mid x, \varphi) = \int_{\mathbb{A}} Q(\cdot \mid x, a)\ \varphi(\mathrm{d}a \mid x)$. En particulier, pour $f \in \mathbb{F}$, on note $c(x, f) = c(x, f(x))$ et $Q(B \mid x, f) = Q(B \mid x, f(x))$. Par abus de notation, on confondra une politique déterministe avec le(s) sélecteur(s) associé(s), et une politique aléatoire avec le(s) noyau(x) de Φ associés.

La propriété de Markov. On rappelle la définition d'un processus de Markov à temps discret.

Définition 2.1.6. Soit X un espace de Borel et $\mathcal{P}(X \mid X)$ l'ensemble des noyaux stochastiques sur X sachant X. Soit $(R_t)_{t \geq 0}$ une suite de noyaux stochastiques de $\mathcal{P}(X \mid X)$ et soit $(Z_t)_{t \geq 0}$ un processus à valeurs dans X. On dit que (Z_t) est un *processus de Markov inhomogène de noyaux de transition* (R_t) si pour tout $B \in \mathcal{B}(X)$ et pour tout t,

$$\mathbb{P}\{Z_{t+1} \in B \mid Z_0, \ldots, Z_t\} = \mathbb{P}\{Z_{t+1} \in B \mid Z_t\} = R_t(B \mid Z_t). \tag{2.13}$$

Si tous les noyaux R_t sont égaux à un noyau stochastique R (c'est-à-dire que les R_t ne dépendent pas de t), alors on dit que (Z_t) est un processus de Markov homogène.

On énonce maintenant la proposition suivante qui donne une condition pour qu'un MDP soit un processus de Markov.

Proposition 2.1.1. Soit ν une distribution initiale arbitraire. Si $\pi = (\varphi_t)_{t\geq 0}$ est une politique markovienne aléatoire (*i.e.* $\pi \in \Pi_{MA}$), alors $(X_t)_{t\geq 0}$ est un processus de Markov inhomogène de noyaux de transition $(Q(. \mid ., \varphi_t))_{t\geq 0}$, c'est-à-dire que pour tout $B \in \mathcal{B}(\mathbb{X})$ et pour tout t,

$$\mathbb{P}_\nu^\pi \{X_{t+1} \in B \mid (X_0, \dots, X_t) = (x_0, \dots, x_t)\} = \mathbb{P}_\nu^\pi \{X_{t+1} \in B \mid X_t = x_t\} \tag{2.14}$$

$$= Q(B \mid x_t, \varphi_t). \tag{2.15}$$

Si $\pi = (f_t)$ est une politique déterministe markovienne (*i.e.* $\pi \in \Pi_{DM}$), les égalités précédentes sont valables pour les noyaux $Q(\cdot \mid \cdot, f_t)$. De plus, si π est une politique stationnaire (*i.e.* $\pi \in \Pi_{DMS}$), alors (X_t) est un processus de Markov homogène.

Exemple 2: Définition de politiques de références

On introduit trois politiques pour contrôler le système défini dans l'Exemple 1, et on considère que les maintenances sont faites le lendemain d'un changement d'état. Ces trois politiques, qui sont déterministes, markoviennes et stationnaires, peuvent être simulées de façon exacte. On les détaille sous la forme de vecteurs qui indiquent l'action à choisir (la même pour toutes les dates de décision) selon l'état du système, dans la Table 2.3.

Politique sans intervention π_0. Le système est quotidiennement envoyé en mission, quel que soit son état, et ne reçoit pas de maintenance sur l'horizon d'étude.

Politique de maintenance corrective π_1. Le système est quotidiennement envoyé en mission tant qu'il est stable ou dégradé. Lorsque le système est en panne, les interventions consistent à le remplacer. On ne fait jamais d'entretien.

Politique de maintenance préventive π_2. Le système est quotidiennement envoyé en mission tant qu'il est stable. Sinon, les actions consistent à le remplacer s'il est en état de panne ou l'entretenir s'il est en état dégradé.

TABLE 2.3 – Détails de l'action choisie par état, pour les politiques π_0 à π_2.

État	π_0	π_1	π_2
stable	mission	mission	mission
dégradé	mission	mission	entretenir
panne	mission	remplacer	remplacer

Les coûts de ces politiques sont ensuite évalués et comparés dans l'Exemple 3.

2.1.3 Critères d'évaluation de politiques

Considérons un modèle markovien décisionnel (2.1) introduit dans la Définition 2.1.1. L'objectif associé à un MDP est alors de trouver pour chaque état du système et pour chaque date de décision, la meilleure action à effectuer en vue de minimiser les coûts générés sur tout l'horizon d'étude. Si le coût par transition c et le coût terminal C permettent de modéliser respectivement la satisfaction instantanée d'une action dans un état, et celle concernant de l'état final du système, il reste à définir des critères qui tiennent compte de toute la dynamique du processus, afin d'évaluer et comparer des politiques. En particulier, ces critères doivent permettre de considérer les effets immédiats mais aussi les conséquences à long terme d'une décision, qui peuvent être plus difficiles à prendre en compte au moment de décider. Ainsi, l'objectif ne vise pas à optimiser le coût par transition, mais plutôt à optimiser des critères de coût, qui sont fonctions de la loi initiale du processus $\nu \in \mathcal{P}(\mathbb{X})$ ainsi que de la politique choisie $\pi \in \Pi$. Le problème d'optimisation associé à un critère de coût consiste alors à le minimiser (ou à le maximiser en cas de récompenses), sur tout l'ensemble des politiques admissibles Π. Plusieurs critères de coût sont étudiés dans la littérature (voir [HLL96]), dont les deux plus courants se distinguent selon que l'horizon du MDP soit fini ou infini.

Le critère à horizon fini. Soit un entier $H \in \mathbb{N}$. On suppose ici qu'un·e agent doit contrôler le système en H étapes, avec H fini. Le critère à horizon fini, partant de l'état initial $x \in \mathbb{X}$ et suivant la politique $\pi \in \Pi$, représente la somme des coûts générés en appliquant π partant de x jusqu'à l'horizon, auquel on ajoute le coût associé à l'état terminal. Ce critère est défini par :

$$V_H(\pi, x) = \mathbb{E}_x^\pi \left[\sum_{t=0}^{H-1} c(X_t, A_t) + C(X_H) \right]. \tag{2.16}$$

Le critère pondéré. Soit un réel $\alpha \in]0, 1[$. On suppose maintenant qu'un·e agent doit contrôler le système en une infinité de dates de décisions. Dans ce cas, l'horizon sur lequel on somme les coûts étant infini, chaque coût par transition est pondéré en fonction du paramètre α. Le critère α-pondéré, partant de l'état initial $x \in \mathbb{X}$ et suivant la politique $\pi \in \Pi$, est défini par :

$$V_\alpha(\pi, x) = \mathbb{E}_x^\pi \left[\sum_{t=0}^{\infty} \alpha^t c(X_t, A_t) \right] \quad . \tag{2.17}$$

Le paramètre $\alpha \in]0, 1[$ de la série permet alors, sous certaines hypothèses (par exemple que la fonction de coût c soit bornée), d'assurer la convergence de la série.

Exemple 3: Comparaison des politiques de références

Grâce à des simulations exactes, on peut maintenant comparer les performances des politiques de références décrites dans l'Exemple 2, pour contrôler le système introduit dans l'Exemple 1. L'horizon de son étude étant fini et fixé à $H = 10$ jours, on évalue et compare le coût de ces politiques via le critère à horizon fini, partant de chacun des états initiaux possibles du système.

Les coûts de ces politiques de référence sont évalués au moyen de 10^5 simulations de Monte-Carlo. Les résultats numériques sont donnés dans la Table 2.4. Pour ces trois politiques, l'intervalle de confiance des coûts ont une longueur inférieure à 1 et ne sont donc pas affichés.

TABLE 2.4 – Coûts des politiques de référence.

État	π_0	π_1	π_2
$x_0 =$ stable	1320	127	**41**
$x_0 =$ dégradé	4281	289	86
$x_0 =$ panne	5000	310	236

Comme attendu, la politique sans maintenance π_0 est bien la plus coûteuse, quel que soit l'état initial du système, car elle génère des pénalités d'échec dès que ce dernier passe dans l'état de panne. On peut également noter que, le coût d'échec étant fixé à 500 et l'horizon à $H = 10$ jours, cette politique partant d'un état initial de panne génère bien un coût de 5000. Ce résultat, qui est attendu, permet de valider le simulateur et confirmer la cohérence des résultats de simulations obtenus.

Ensuite, la politique de maintenance corrective π_1 réduit les coûts de gestion du système en intervenant dès sa panne, ce qui évite de cumuler des pénalités d'échec. Cela génère un gain relatif de 90% et 93% par rapport à la politique sans maintenance π_0, partant par exemple des états initiaux stable et dégradé. Néanmoins, cette politique mène à des effectuer des opérations correctives sur le système, qui sont plus coûteuses que les opérations préventives.

Enfin, une politique de maintenance préventive π_2 réduit efficacement les coûts de maintenance quel que soit son état initial, en intervenant sur le système avant sa défaillance, et plus précisément dès son passage dans l'état dégradé. Cela génère un gain relatif par rapport à la politique sans maintenance π_0 de 96% et un gain relatif de 67% par rapport à la politique de maintenance corrective π_1, partant par exemple de l'état initial stable.

2.2 Méthodes de résolution du critère fini

Le problème d'optimisation associé à un MDP consiste alors à minimiser la fonction qui associe à une politique l'évaluation du critère donné, sur l'ensemble des politiques Π. L'optimum est appelé *fonction valeur* et une politique est dite *optimale* si elle réalise cet optimum. Les méthodes permettant de les calculer diffèrent ensuite selon le critère à optimiser.

Brièvement, pour optimiser le critère pondéré, les équations de *programmation dynamique* [Bel57a] donnent lieu à deux méthodes de résolution. Ces méthodes, appelées *itération de la valeur* (voir e.g. [Ber87, HLL96]) et *itération de la stratégie* (voir e.g. [How60, WW89, Den03]) fournissent des algorithmes très étudiés et qui sont utilisés dans de nombreuses applications. Enfin, ce critère s'optimise aussi par *programmation linéaire* (voir e.g. [Den70, HLL96]).

Aussi, concernant l'application industrielle étudiée au Chapitre 3, on s'intéresse à la dynamique d'un système étudié sur un horizon fini. Ainsi, on se restreint pour la suite de ce chapitre à présenter uniquement des méthodes de résolution du critère à horizon fini.

Dans cette partie, notre objectif est de présenter plusieurs méthodes de résolution d'un MDP à espace d'état et d'action finis. On présente sous-section 2.2.1 la programmation dynamique, qui fournit à la fois l'optimum et une politique optimale associée, ainsi que son application à la résolution du MDP introduit dans l'Exemple 1. Ensuite, on décrit les méthodes approchées MRAS (sous-section 2.2.2) et ASA (sous-section 2.2.3), qui seront ensuite appliquées à la résolution de notre problématique industrielle dans le Chapitre 4.

2.2.1 Programmation dynamique à horizon fini

Dans le cas du critère à horizon fini, la fonction valeur est donnée par :

$$V(x) = \inf_{\pi \in \Pi} V_H(\pi; x). \tag{2.18}$$

Une politique $\pi^* \in \Pi$ est dite *optimale* si elle vérifie :

$$V_H(\pi^*, x) = V(x). \tag{2.19}$$

Dans cette section, on présente la programmation dynamique à horizon fini en se basant sur [HLL96]. Elle consiste à résoudre un problème en le décomposant en sous-problèmes, puis à résoudre ces derniers des plus petits aux plus grands en stockant les résultats intermédiaires. Ce principe permet de déterminer une solution optimale d'un problème à partir des solutions de tous les sous-problèmes. Cette méthode d'optimisation fournit à la fois l'optimum $V(x)$ du problème et une politique optimale π^* qui le réalise.

L'optimisation du critère fini repose sur le théorème de programmation dynamique (voir [HLL96] section 3.2) qui établit une récurrence initialisée avec le coût terminal C.

Théorème. Soient $v_0, v_1, \ldots, v_H$ de $\mathbb{X}$ dans $\mathbb{R}$ définies par récurrence rétrograde par :

$$v_H(x) = C(x) \text{ pour tout } x \in \mathbb{X}$$

et pour t allant de $H-1$ à 0

$$v_t(x) = \min_{a \in \mathbb{A}(x)} \left[c(x,a) + \int_{\mathbb{X}} v_{t+1}(y) Q(\,\mathrm{d}y \mid x, a) \right]. \tag{2.20}$$

Si ces fonctions sont mesurables et que, $\forall\, t \in [\![0; H-1]\!]$, il existe un sélecteur $f_k \in \mathbb{F}$ tel que $f_t(x) \in \mathbb{A}(x)$ atteigne le minimum dans 2.20, alors la politique **déterministe markovienne** $\pi^* = \{f_0, \ldots, f_{H-1}\}$ est optimale et la fonction valeur V^* est égale à v_0.

De ce théorème découle l'algorithme de programmation dynamique décrit Algorithme 1.

Algorithme 1 : Programmation dynamique à horizon fini

Entrées : $\mathbb{X}$, $\mathbb{A}$, Q , fonction de coûts c, coût terminal C, horizon H

début
  pour tout $x \in \mathbb{X}$ **faire**
    $v_H(x) = C(x)$
  pour t de $H-1$ **à** 0 **faire**
    pour tout $x \in \mathbb{X}$ **faire**
      $v_t(x) = \min_{a \in \mathbb{A}(x)} \left[c(x,a) + \sum_{y \in \mathbb{X}} v_{t+1}(y) Q(y \mid x, a) \right]$
      $\pi_t^*(x) = \operatorname{argmin}_{a \in \mathbb{A}(x)} \left[c(x,a) + \sum_{y \in \mathbb{X}} v_{t+1}(y) Q(y \mid x, a) \right]$
  retourner v_0, π^*

Remarque 2.2.1. Bien que la programmation dynamique soit définie pour des espaces très généraux, son implémentation pour calculer la fonction valeur requiert que les espaces d'états et d'actions soient finis pour être parcourus, et que le noyau de transition soit explicite analytiquement. Par ailleurs, elle souffre du *fléau de la dimension* [Bel57a], phénomène selon lequel la complexité de résolution d'un problème d'optimisation croît avec le nombre de paramètres et la taille de leurs domaines respectifs.

En pratique, si l'espace d'états du MDP n'est pas fini, ce qui sera le cas pour le modèle physique défini au Chapitre 3, on peut utiliser une méthode de discrétisation pour se ramener au cas fini. On présente notamment la méthode de quantification pour discrétiser une variable aléatoire dans la section 2.3. De plus, si le noyau n'est pas explicite analytiquement, ce qui sera le cas pour le modèle physique défini au Chapitre 3, on peut utiliser des méthodes de résolution approchées pour résoudre le MDP. On présente notamment les méthodes MRAS et ASA pour résoudre un MDP dans ces cas, dans les sections 2.2.2 et 2.2.3.

Exemple 4: Calcul de la fonction valeur et d'une politique optimale

On met en oeuvre la **programmation dynamique** pour calculer la **fonction valeur**, ainsi qu'une la **politique optimale associée**, pour le MDP introduit dans l'Exemple 1. Leur calcul montre qu'il existe une politique qui réduit encore les coûts de maintenances par rapport aux politiques de références introduites dans l'Exemple 2 et évaluées dans l'Exemple 3. Son coût, calculé par méthode de Monte-Carlo via 10^5 simulations, est bien celui de la fonction valeur. Ces résultats sont résumés Table 2.5, où l'intervalle de confiance du coût n'est pas donné car d'une longueur proche de zero.

TABLE 2.5 – Fonction valeur et coût de la politique optimale obtenus par application de la programmation dynamique.

État	Fonction valeur	coût de π^*
$x_0 =$ stable	**36**	36
$x_0 =$ dégradé	**82**	82
$x_0 =$ panne	**232**	232

Elle génère un gain relatif de 97% par rapport à la politique sans maintenance π_0 et un gain relatif de 71% par rapport à la politique de maintenance corrective π_1, partant par exemple de l'état initial stable. On note que cette politique réduit aussi les coûts de maintenance par rapport à la politique de maintenance préventive π_2, quel que soit son état initial, et réalise un gain relatif de 12%, partant par exemple de l'état initial stable.

La **politique optimale** π^* donnée par la programmation dynamique, qui est markovienne déterministe mais non stationnaire, est donnée Table 2.6. On note que cette politique est proche de la politique préventive π_2, en envoyant le système en atelier dès la première dégradation ou panne, et en mission sinon. Aussi, elle s'en distingue par le choix d'une action différente à la dernière date de décision, $t = H-1$. Cette non stationnarité s'explique par le fait qu'aucune pénalité ne sera à payer si le système tombe en panne au dernier jour de l'étude. Aussi, il n'est pas non plus utile de prolonger sa durée de vie via des opérations de maintenance à cette date : l'envoi en mission est l'action la moins coûteuse à ce terme.

TABLE 2.6 – Détails des actions par état pour la politique optimale π^*.

État	$0 \leq t < H-1$	$t = H-1$
stable	mission	mission
dégradé	entretenir	mission
panne	remplacer	remplacer

2.2.2 Model Reference Adaptive Search

Dans cette section, on présente la méthode *Model Reference Adaptive Search* (MRAS) en se basant sur [CFHM07, HHC12, CFHM13]. C'est une méthode d'optimisation stochastique qui propose une approche **basée sur des modèles** et utilise des simulations et la méthode de Monte-Carlo. Dans cette démarche, la recherche de solution est **itérative** et alterne deux phases :

1. Générer des solutions candidates selon un modèle probabiliste spécifié. Dans le cas d'un MDP, la solution recherchée est une politique qui réalise le plus faible coût possible, et le modèle probabiliste est une distribution de probabilités sur l'ensemble des politiques admissibles
2. Mettre à jour les paramètres associés au modèle probabiliste, grâce à l'évaluation des solutions candidates générées lors de la première phase, afin de diriger la recherche vers de meilleures solutions.

Dans un premier temps on introduit sous-sous-section 2.2.2.1 les principes de cette méthode d'optimisation, dans le cas général d'une densité à optimiser. Ensuite on s'intéresse au cas où la fonction de coût ne peut pas être évaluée de manière analytique, mais estimée typiquement par simulations via la méthode de Monte-Carlo. En particulier, on s'intéresse à cette procédure appliquée à l'optimisation d'un MDP à espace d'états, d'actions et horizon fini. Dans ce cas, la procédure MRAS est une méthode d'apprentissage sur des politiques, dont le but est de trouver la *meilleure politique* dans une classe de politiques paramétrée.

2.2.2.1 Idée générale de la méthode MRAS

Soit le problème de minimisation suivant, où $f : E \to \mathbb{R}$ est une fonction minorée sur un ensemble non vide $E \subset \mathbb{R}^n$. On cherche un minimum $x^* \in E$ tel que :

$$x^* \in \underset{x \in E}{\operatorname{argmin}}\ f(x). \tag{2.21}$$

La méthode MRAS s'appuie sur une famille de lois de probabilités $\varphi(\cdot, \theta)$ sur E dépendant d'un paramètre θ pour générer des solutions candidates au problème. Ainsi, la recherche d'un minimum de f se traduit par la recherche d'un paramètre optimal θ^* pour lequel les réalisations de la distributions $\varphi(\cdot, \theta^*)$ sont des solutions candidates qui minimisent la fonction f. Aussi, la mise à jour du paramètre θ au fil des itérations s'appuie sur une suite de distributions de référence $(g_k)_{k \geq 1}$, où à chaque itération k, la nouvelle estimation du paramètre θ^* est celle qui minimise la divergence de Kullback-Leibler :

$$D(g_k, \varphi(\cdot, \theta)) = \int_E \ln\left(\frac{g_k(x)}{\varphi(x, \theta)}\right) g_k(x) \mathrm{d}x. \tag{2.22}$$

Enfin, la suite de lois de références $(g_k)_{k\geq 1}$ est déterminée selon une distribution initiale $g_1(x) > 0,\ \forall x \in E$ puis construite par récurrence, selon l'équation :

$$g_k(x) = \frac{f(x)g_{k-1}(x)}{\int_E f(x)g_{k-1}(x)\mathrm{d}x} \text{ pour tout } x \in E, \tag{2.23}$$

La recherche de θ^* dépend alors du choix de la famille $\varphi(\cdot, \theta)$. La famille exponentielle, qui fournit une expression analytique pour le calcul du paramètre θ qui minimise la divergence de Kullback-Leibler, est alors couramment utilisée [ZFM10]. C'est cette famille qui est utilisée dans l'algorithme qu'on décrit ensuite dans la sous-sous-section 2.2.2.2, consacrée au cas de l'optimisation de la stratégie de contrôle d'un MDP à horizon fini.

2.2.2.2 Application à la résolution de MDP à horizon fini

On détaille dans cette section l'application de la méthode MRAS dans le cas particulier d'optimisation de la fonction valeur et d'une politique associée, pour un MDP à horizon fini.

Dans ce cas, la fonction à minimiser est celle qui associe à une politique son coût moyen, calculé par méthode de Monte-Carlo, afin d'estimer le critère à horizon fini. Ainsi, le paramètre θ à optimiser est une distribution de probabilités sur l'ensemble des politiques admissibles du MDP. Dans le détail, il s'agit d'une matrice de probabilités P à trois dimensions où chaque coefficient $P(i, j, t)$ représente la probabilité de choisir l'action a_j lorsque le système se trouve dans l'état x_i à la date de décision t.

À chaque itération de l'algorithme, cette matrice de probabilités est mise à jour en fonction des résultats de simulations du MDP que l'on cherche à optimiser, c'est à dire en fonction du coût des politiques candidates générées selon l'estimation courante de la distribution de probabilités. On obtient ainsi une suite $(P_k)_{k>0}$ de matrices qui représente la succession des mises à jours. Le principe de l'algorithme est de mettre une probabilité plus élevée sur les actions menant à des politiques moins coûteuses, de sorte que pour chaque date de décision et chaque état du MDP, la distribution de probabilités se concentre sur l'action ayant conduit à construire la politique la moins coûteuse. Cette méthode permet donc de construire des politiques déterministes markoviennes qui permettent de simuler des trajectoires du MDP contrôlées par ces politiques.

Dynamique de l'algorithme. On détaille maintenant les 4 différentes étapes qui sont effectuées à chaque itération de l'algorithme.

Étape 1 : simulation de politiques candidates. On génère un nombre (initialisé en entrée de l'algoritme) de politiques candidates, qui sont construites en tirant les actions soit selon la matrice de probabilité courante P, soit selon la matrice de probabilité initiale P_0.

Étape 2 : évaluation des performances. Pour chacune des politiques construites à l'étape précédente, on évalue leur coût via un certain nombre de simulations (donné en entrée), par méthode de Monte-Carlo. Ce nombre de simulations, qui doit être croissant au cours des itérations, augmente selon un paramètre donné en entrée de l'algorithme.

Étape 3 : détermination des meilleures politiques candidates. Cette étape consiste à sélectionner les politiques candidates gardées pour le calcul de la mise à jour de P. Pour ce faire, on calcule un seuil de performance et on ne garde que la proportion de politiques dont le coût est inférieur à ce seuil. Si ce seuil est meilleur que celui de l'itération précédente, alors la mise à jour se fait encore avec cette proportion de politiques. Sinon, on cherche s'il existe une politique qui engendre un coût moins élevé que celui de l'itération précédente, et ce seuil plus faible devient le nouveau. Ce seuil, qui décroît donc à chaque itération où il est nécessaire de sélectionner moins de politiques pour améliorer les performances, conduit alors la distribution de probabilités à générer des politiques les moins coûteuses possible.

Étape 4 : mise à jour de P. Selon l'issue de l'étape 3, c'est ici que la distribution de probabilités P peut être mise à jour. En effet, si l'étape 3 n'a pas mené à un meilleur seuil qu'à l'itération précédente, alors P ne change pas. Sinon, une nouvelle estimation $\hat{P}_{k+1}$ est calculée en ne faisant intervenir que les politiques sélectionnées, à une tolérance ε près, donnée en paramètre. Enfin, P_{k+1} est mise-à-jour via une moyenne pondérée de $\hat{P}_{k+1}$ et P_k.

Hyperparamètres de l'algorithme MRAS. On note $\#\mathbb{X}$ le nombre d'états, $\#\mathbb{A}$ le nombre d'actions, et H l'horizon du MDP. L'algorithme dépend des hyperparamètres :

- $P_0 \in \mathcal{M}_{\#\mathbb{X};\#\mathbb{A};H}([0;1])$: matrice de probabilités initiale, qui doit être proche d'une loi uniforme sur les politiques admissibles, mais peut aussi être choisie selon un prior.
- $\rho_0 \in]0;1]$: quantile initial qui sert à calculer le seuil de performance, et détermine la proportion de politiques retenues pour l'étape de mise-à-jour de P,
- $\varepsilon > 0$: tolérance sur le coût d'autres politiques retenues pour la mise à jour de P,
- $N_0 \geq 2$: nombre initial de politiques candidates à générer,
- $M_0 \geq 1$: nombre initial de simulations pour la méthode de Monte-Carlo,
- $\alpha > 1$: facteur par lequel on multiplie le nombre de politiques candidates à générer à la prochaine itération, lorsqu'on ne trouve pas de meilleur seuil,
- $\beta > 1$: facteur par lequel on multiplie le nombre de simulations de la méthode de Monte-Carlo pour la prochaine itération,
- $\lambda \in]0;1[$: coefficient de mélange qui détermine la proportion de politiques candidates qui seront simulées selon la matrice initiale P_0,
- $\nu \in]0;1[$: coefficient utilisé au moment de la mise à jour de P afin d'améliorer les performances numériques de l'algorithme,
- $K \in \mathbb{N}$: nombre limite d'itérations,

— $\mathcal{H} : \mathbb{R} \to \mathbb{R}^+$ fonction strictement décroissante, qui garantit que les coefficients de P sont positifs.

L'algorithme fait ensuite appel aux deux fonctions suivantes.

Fonction de seuillage. La fonction $\mathcal{I}$ permet de quantifier la contribution d'une politique en fonction de son coût x, du seuil de performance sélectionné χ, et d'une tolérance ε. Elle est définie par l'équation 2.24 et la Figure 2.2 en donne une représentation graphique.

$$\mathcal{I}(x, \chi) = \begin{cases} 0 \text{ si } x \geq \chi + \varepsilon \\ \dfrac{\chi + \varepsilon - x}{\varepsilon} \text{ si } \chi < x < \chi + \varepsilon \\ 1 \text{ si } x \leq \chi \end{cases} \tag{2.24}$$

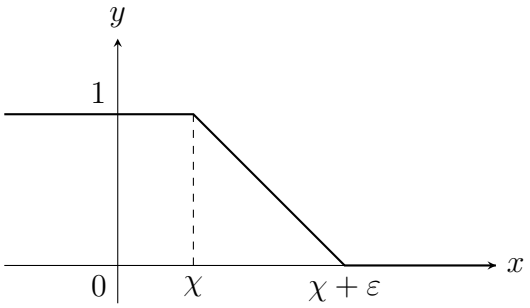


FIGURE 2.2 – Tracé de la fonction de seuillage $x \to \mathcal{I}(x, \chi)$.

Distribution sur les politiques candidates. La fonction **f** apparait dans le calcul de la mise à jour de la distribution de probabilités P. Pour tous états $1 \leq i \leq \#\mathbb{X}$, actions $1 \leq j \leq \#\mathbb{A}$, et dates de décision $0 \leq t \leq H - 1$, on pose :

$$\Pi_{i,j}(t) = \left\{ \text{politiques } \pi = (\pi_k)_{0 \leq k \leq H-1} \mid \pi_t(x_i) = a_j \right\}, \tag{2.25}$$

qui indique si l'action a_j a été choisie dans l'état x_i à la date de décision t de la politique π, et on définit ensuite **f** par :

$$\mathbf{f}(\pi, P) = (1 - \lambda) f(\pi, P) + \lambda f(\pi, P_0), \tag{2.26}$$

avec

$$f(\pi, P) = \prod_{t=0}^{H-1} \prod_{i=1}^{\#\mathbb{X}} \prod_{j=1}^{\#\mathbb{A}} P(i, j, t)^{\mathbb{1}_{\Pi_{i,j}(t)}(\pi)}. \tag{2.27}$$

La description détaillée de l'algorithme est finalement décrite ci-dessous.

Instructions. Pour k de 0 à $K-1$, répéter les instructions suivantes :

1. Simulation des politiques candidates :
   Construire N_k politiques notées $\pi_n\,(n \in [\![1; N_k]\!])$ de sorte que pour chaque n :
   — avec probabilité β_k, tirer les actions de π_n selon la matrice P_0 ;
   — avec probabilité $1-\beta_k$, tirer les actions de π_n selon la matrice P.
2. Évaluation des performances des politiques candidates :
   Pour chaque $n \in [\![1; N_k]\!]$, calculer par méthode de Monte-Carlo $\bar{V}_k^n$, le coût de la π_n-ième politique avec M_k simulations de trajectoire partant de l'état initial x_0.
3. Détermination des meilleures politiques candidates :
   (a) Trier les coûts des politiques $\bar{V}_k^n$ dans l'ordre décroissant.
   (b) Calculer le seuil $\gamma_k\,(\rho_k, N_k) = \bar{V}_k^{(\lceil (1-\rho_k)N_k \rceil)}$.
   (c) Mettre à jour le seuil de performance $\bar{\gamma}_k$, le quantile ρ_{k+1} et le nombre de stratégies à simuler N_{k+1}, de la façon suivante :
      i. SI $k=0$ OU $\gamma_k\,(\rho_k, N_k) \leq \bar{\gamma}_{k-1} - \varepsilon$,
         ≻ $\bar{\gamma}_k,\ \rho_{k+1},\ N_{k+1} \leftarrow \gamma_k\,(\rho_k, N_k)\,,\ \rho_k,\ N_k$;
         ≻ $\boldsymbol{\pi}_k^* \leftarrow$ une stratégie π^n telle que $\bar{V}_k^{(n)} = \gamma_k\,(\rho_k, N_k)$.
      ii. SINON on cherche le plus petit entier $\nu > \lceil (1-\rho_k)\,N_k \rceil$ tel que $\bar{V}_k^{(\nu)} \leq \bar{\gamma}_{k-1} - \varepsilon$;
         ▷ Si ν existe :
            ≻ $\bar{\gamma}_k,\ \rho_{k+1},\ N_{k+1} \leftarrow \bar{V}_k^{(\nu)},\ 1 - \frac{\nu}{N_k},\ N_k$;
            ≻ $\boldsymbol{\pi}_k^* \leftarrow$ une stratégie π^n telle que $\bar{V}_k^{(n)} = \bar{V}_k^{(\nu)}$.
         ▷ Sinon :
            ≻ $\bar{\gamma}_k \leftarrow \bar{V}_k^n$ où $\pi^n = \boldsymbol{\pi}_{k-1}^*$;
            ≻ $\rho_{k+1},\ N_{k+1},\ \boldsymbol{\pi}_k^* \leftarrow \rho_k,\ \lceil \alpha N_k \rceil,\ \boldsymbol{\pi}_{k-1}^*$.
4. Calcul de la matrice P_{k+1}
   (a) Si pour tout $n \in [\![1; N_k]\!]$, $\dfrac{\left[\mathcal{H}\left(\bar{V}_k^n\right)\right]^k}{\mathbf{f}\,(\pi_n, P_k)} \mathcal{I}\left(\bar{V}_k^n, \bar{\gamma}_k\right) = 0$, alors $P_{k+1} \leftarrow P_k$,
   (b) Sinon on calcule une matrice auxiliaire $\hat{P}_{k+1}$ de la façon suivante : pour tous entiers $1 \leq i \leq \#\mathbb{X},\ 1 \leq j \leq \#\mathbb{A}$ et $0 \leq t \leq H-1$,

$$\hat{P}_{k+1}(i,j,t) \leftarrow \frac{\sum_{n=1}^{N_k} \dfrac{\left[\mathcal{H}\left(\bar{V}_k^n\right)\right]^k}{\mathbf{f}\,(\pi_n, P_k)} \mathcal{I}\left(\bar{V}_k^n, \bar{\gamma}_k\right) \mathbb{1}_{\Pi_{i,j}(t)}\,(\pi_n)}{\sum_{n=1}^{N_k} \dfrac{\left[\mathcal{H}\left(\bar{V}_k^n\right)\right]^k}{\mathbf{f}\,(\pi_n, P_k)} \mathcal{I}\left(\bar{V}_k^n, \bar{\gamma}_k\right)}. \tag{2.28}$$

   (c) $P_{k+1} \leftarrow \alpha_k \hat{P}_{k+1} + (1-\alpha_k)\,P_k$.
   (d) $M_{k+1} \leftarrow \lceil \beta M_k \rceil$.

Réécriture des coefficients des matrices $\hat{P}$. Le calcul des matrices $\hat{P}_{k+1}$ (2.28) qui permet la mise à jour des distributions de probabilités fait intervenir de nombreux produits de probabilités dans le calcul de $\mathbf{f}(\pi, P)$ (2.26). Aussi, on observe rapidement des problèmes numériques liés à l'accumulation de valeurs proches de zéro, et a fortiori lorsque l'espace d'état $\mathbb{X}$, l'espace d'ation $\mathbb{A}$ et l'horizon H sont grands. On pose alors pour toute itération $k \in [\![0; K-1]\!]$ et pour toute politique $n \in [\![1; N_k]\!]$,

$$c_n^{(k)} = \frac{A_n^{(k)}}{Fk_n^{(k)} + F0_n^{(k)}}, \tag{2.29}$$

avec

$$A_n^{(k)} = \left[\mathcal{H}\left(\bar{V}_k^n\right)\right]^k \mathcal{I}\left(\bar{V}_k^n, \bar{\gamma}_k\right), Fk_n^{(k)} = (1-\lambda) f(\pi_n, P_k) \text{ et } F0_n^{(k)} = \lambda f(\pi_n, P_0), \tag{2.30}$$

de sorte que

$$\hat{P}_{k+1}(i,j,t) = \frac{\sum_{n=1}^{N_k} c_n^{(k)} \mathbb{1}_{\Pi_{i,j}(t)}(\pi_n)}{\sum_{n=1}^{N_k} c_n^{(k)}}. \tag{2.31}$$

L'instabilité observée provient en particulier des calculs du dénominateur des coefficients $c_n^{(k)}$. Une première étape est alors de calculer le logarithme des termes $F0_n^{(k)}$ et $Fk_n^{(k)}$, puisque les sommes sont numériquement plus stables que les produits. Ensuite, diviser le numérateur et le dénominateur des coefficients $c_n^{(k)}$ par le plus grand des deux termes du dénominateur conduit à évaluer une exponentielle la plus faible possible, afin de stabiliser les calculs. On réécrit alors les coefficients $c_n^{(k)}$ comme un produit de deux termes par,

$$c_n^{(k)} = a_n^{(k)} \exp(b_n^{(k)}) = \begin{cases} \frac{\exp\left(\log A_n^{(k)} - \log Fk_n^{(k)}\right)}{1+\exp\left(\log F0_n^{(k)} - \log Fk_n^{(k)}\right)} & \text{si } Fk_n^{(k)} \geq F0_n^{(k)} \\ \frac{\exp\left(\log A_n^{(k)} - \log F0_n^{(k)}\right)}{1+\exp\left(\log Fk_n^{(k)} - \log F0_n^{(k)}\right)} & \text{si } F0_n^{(k)} \geq Fk_n^{(k)}. \end{cases}$$

Si le dénominateur est stabilisé via cette démarche, et toujours calculable puisqu'il est compris entre 1 et 2, le numérateur peut encore se montrer très instable, du fait du terme apparu au numérateur. Une seconde étape est alors de normaliser chacun des coefficients $c_n^{(k)}$. Aussi, on calcule $\hat{N} = \underset{n \in [\![1;N_k]\!]}{\operatorname{argmax}} b_n^{(k)}$, et on factorise les coefficients $c_n^{(k)}$ par $c_{\hat{N}}^{(k)}$, de sorte que,

$$\hat{P}_{k+1}(i,j,t) = \frac{\sum_{n=1}^{N_k} \frac{c_n^{(k)}}{c_{\hat{N}}^{(k)}} \mathbb{1}_{\Pi_{i,j}(t)}(\pi^n)}{\sum_{n=1}^{N_k} \frac{c_n^{(k)}}{c_{\hat{N}}^{(k)}}}.$$

Exemple 5: Recherche d'une politique stationnaire optimale avec MRAS.

On va maintenant appliquer la méthode MRAS pour optimiser la stratégie de maintenance du système introduit dans l'Exemple 1. Une première possibilité, qu'on détaille dans cet exemple, est de l'utiliser pour rechercher une politique stationnaire optimale. Pour ce faire, l'hyperparamètre de l'algorithme MRAS qui permet à la fois d'encoder les actions possibles par état et la stationnarité recherchée est la matrice $P_0 \in \mathcal{M}_{\#X;\#A;H}([0;1])$.

D'une part, sa construction se base sur $\mathbb{A}(x)$ l'ensemble des actions possibles par état, détaillé dans la Table 2.7a, où on désigne par 1 (resp. 0) les actions possibles (resp. interdites) dans un état donné. D'autre part, si la politique optimale recherchée est stationnaire, alors la probabilité de choisir une action dans un état ne dépend plus de la date de décision, et dans ce cas on peut restreindre la matrice $P_0 \in \mathcal{M}_{\#\mathbb{X};\#\mathbb{A}}([0;1])$. Elle est ensuite initialisée à une loi uniforme (les coefficients associés à un état x sont $\frac{1}{\#\mathbb{A}(x)}$) et est détaillée dans la Table 2.7b. Les autres paramètres sont fixés arbitrairement à $N_0 = 10,\ \alpha = 1.05,\ M_0 = 10^3,\ \beta = 1.01,\ \varepsilon = 1,\ \nu = 0.1,\ \lambda = 0.05,\ \rho = 0.25,\ K = 20$.

TABLE 2.7 – Distribution de probabilités sur l'ensemble des actions possibles par état.

x \ a	M	E	R
stable	1	1	1
dégradé	1	1	1
panne	1	0	1

(a) Ensemble d'actions possibles par état $\mathbb{A}(x)$.

x \ a	M	E	R
stable	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
dégradé	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
panne	$\frac{1}{2}$	0	$\frac{1}{2}$

(b) Initialisation de la matrice P_0.

La **politique stationnaire optimale** π^*_{st} donnée par l'algorithme MRAS est décrite Table 2.8b. On note que cette politique correspond à la politique de référence π_2 introduite dans l'Exemple 2. Ce résultat est cohérent car la politique π^* donnée par la programmation dynamique ne se différencie de π_2 que par une action à la date $H-1$. Son coût, calculé par méthode de Monte-Carlo dans l'Exemple 3, correspond bien à celui de la fonction valeur calculée par MRAS, dont les résultats sont résumés Table 2.8a.

TABLE 2.8 – Fonction valeur et politique optimale déterminée par l'algorithme MRAS.

État initial	Fonction valeur
x_0 = stable	**41**
x_0 = dégradé	**86**
x_0 = panne	**236**

(a) Fonction valeur.

État	Action
stable	mission
dégradé	entretenir
panne	remplacer

(b) Action par état pour la politique π^*_{st}.

Exemple 6: Discussions autour de l'Exemple 5.

Pour chaque état initial (x_0), on représente l'évolution du seuil de performance (coût) de la méthode MRAS, dans la Figure 2.3, en fonction du nombre d'itérations (K). On note que la méthode converge en seulement quelques itérations. En effet, il existe $\prod_{x\in\mathbb{X}} \#\mathbb{A}(x) = 18$ politiques possibles sur ce problème, ce qui explique que la méthode s'oriente rapidement vers une solution, partant de $N_0 = 10$ politiques candidates.

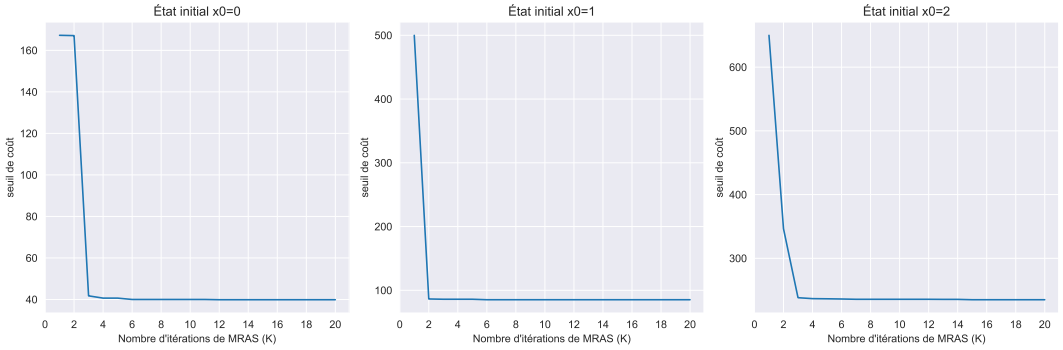


FIGURE 2.3 – Évolution du seuil d'acceptation de politiques pour la mise à jour de la matrice P_k, en fonction du nombre d'itérations et de l'état initial du système x_0.

Aussi, on s'intéresse aux matrices des probabilités (P_k), détaillées dans la Table 2.9. On observe que ces distributions semblent converger vers des masses de Dirac, partant des états dégradé et panne. Par contre, en partant de l'état stable, on note que deux actions gardent la même probabilité d'être choisie dans l'état de panne.

TABLE 2.9 – Détails des matrices de probabilités (P_k) selon l'état initial (x_0).

x \a	M	E	R
S	0.9	0.05	0.05
D	0.05	0.9	0.05
P	0.5	0	0.5

(a) Cas : x_0=stable.

x \a	M	E	R
S	0.97	0.02	0.01
D	0.02	0.96	0.02
P	0.06	0	0.94

(b) Cas : x_0=dégradé.

x \a	M	E	R
S	0.98	0.01	0.01
D	0.02	0.96	0.02
P	0.02	0	0.98

(c) Cas : x_0=panne.

On étudie la politique stationnaire qui associe à l'état stable (resp. dégradé et panne) l'action mission (resp. entretien et mission). Le calcul de son coût montre qu'elle vérifie également le coût optimal (41) partant de l'état stable. Ceci s'explique par le fait que le système observe en moyenne 0 panne lorsqu'il part de l'état stable et qu'il est contrôlé par cette politique, sur ces simulations. La méthode MRAS étant basée sur les simulations de trajectoires, elle ne peut estimer la fonction valeur qu'à partir d'observations, contrairement à la programmation dynamique qui bénéficie des probabilités théoriques.

2.2.3 Approximate Stochastic Annealing

Dans cette section, on décrit la méthode *Approximate Stochastic Annealing* (ASA) (voir [CFHM13]), qui est une méthode d'optimisation stochastique similaire à la méthode MRAS, également basée sur le concept de recuit simulé (voir [KGV83, Cer85]).

Son fonctionnement repose lui aussi sur l'optimisation d'un paramètre θ qui minimise la divergence de Kullback-Leibler (2.22), et dont les réalisations de la distributions $\varphi(\cdot, \theta^*)$ sont des solutions candidates qui minimisent la fonction f (2.21). Dans le cadre des MDP à horizon fini, ce paramètre est encore une matrice de probabilités (notée P) à trois dimensions où chaque coefficient $P(i, j, t)$ représente la probabilité de choisir l'action a_j lorsque le système se trouve dans l'état x_i à la date de décision t.

Aussi, on note que cette méthode présente deux principales différences avec la méthode MRAS. La première différence vient de la suite de lois de référence $(g_k)_{k\geq 1}$ choisie, qui est ici une distribution de Boltzmann :

$$g_k(x) = \frac{\exp\left(-\frac{f(x)}{T_k}\right)}{\int_E \exp\left(-\frac{f(x)}{T_k}\right) \mathrm{d}x}, \tag{2.32}$$

où T_k est un paramètre de température décroissant lorsque k tend vers l'infini.

Enfin, la seconde différence entre les deux méthodes est la fréquence de la mise à jour de la distribution de probabilités P_k. En effet, la méthode MRAS se base sur l'évolution d'un quantile ρ pour ne sélectionner qu'une proportion des politiques candidates les plus performantes, afin de mettre à jour la distribution P_k. Cette mise à jour n'est alors effectuée que si au moins une des politiques candidates a donné une performance meilleure, à ε près, que le seuil associé à ρ. À l'inverse, pour la méthode ASA, la distribution de probabilités P_k est mise à jour à toutes les itérations de l'algorithme et à partir de toutes les politiques candidates générées via les distributions.

Dynamique de l'algorithme ASA. On détaille maintenant les 4 différentes étapes, très similaires à celle de la méthode MRAS, qui sont effectuées à chaque itération de l'algorithme.

Étape 1 : simulation des politiques candidates. L'algorithme génère un nombre (initialisé en entrée de l'algorithme) de politiques candidates. Comme pour MRAS, certaines sont construites en tirant les actions selon la matrice de probabilité courante P_k, et les autres selon la matrice de probabilité initiale P_0. La proportion de politiques simulées selon P_0 doit décroitre de manière au moins polynômiale au cours des itérations, et le nombre de politiques candidates doit croître de façon polynômiale.

Étape 2 : évaluation des performances. Pour chaque politique candidate construite à l'étape précédente, on évalue son coût via un certain nombre de simulations (donné en entrée), par méthode de Monte-Carlo. Ce nombre de simulations, qui doit être croissant au cours des itérations, augmente selon une suite donnée en entrée de l'algorithme.

Étape 3 : **mise à jour de la distribution de probabilités** P**.** Enfin, la matrice P_k est mise à jour via la prise en compte de toutes les politiques générées, sans en sélectionner. Aussi, son calcul se fait en donnant une pondération aux politique qui est proportionnelle à leur performance, où intervient un paramètre de température qui décroît au fur et à mesure des itérations. Enfin, P_{k+1} est mise-à-jour via une moyenne pondérée de $\hat{P}_{k+1}$ et P_k.

Hyperparamètres de la méthode ASA. L'algorithme dépend de plusieurs hyperparamètres d'entrée.

- $P_0 \in \mathcal{M}_{\#\mathbb{X},\#\mathbb{A},H}([0;1])$: matrice de probabilités initiale, qui doit être initialisée à une loi uniforme (i.e. tous ses coefficients doivent valoir $\frac{1}{\#\mathbb{A}(x)}\big)$
- $\alpha_0 = \frac{1}{100^{0,501}}$: coefficient de lissage initial qui contrôle l'évolution de la suite (P_k) ;
- $\beta_0 = 1$: coefficient de mélange initial qui détermine la proportion de politiques simulées selon la matrice P_0 ;
- $T_0 \in \mathbb{R}_+^*$: température initiale, qui dirige la convergence de l'algorithme ;
- $N_0 \in \mathbb{N}^*$: nombre initial de politiques candidates à simuler ;
- $M_0 \in \mathbb{N}^*$: nombre initial de simulations pour la méthode de Monte-Carlo ;
- $K \in \mathbb{N}^*$: nombre limite d'itérations.

Remarque 2.2.2. Les valeurs prises pour α_0 et β_0 sont déterminées par le choix des suites (α_k) et (β_k) dont les formules sont données dans la description des instructions.

Une distribution sur l'ensemble des politiques candidates L'algorithme fait encore appel à une distribution de probabilité $\mathbf{f}$ sur l'ensemble des politiques, comme évoquée pour l'algorithme MRAS, et apparait dans le calcul de la mise à jour de la matrice P. Elle est définie par :

$$\mathbf{f}(\pi, P) = (1 - \beta_k)\, f(\pi, P) + \beta_k f(\pi, P_0), \tag{2.33}$$

avec

$$f(\pi, P) = \prod_{t=0}^{H-1} \prod_{i=1}^{\#\mathbb{X}} \prod_{j=1}^{\#\mathbb{A}} P(i,j,t)^{\mathbb{1}_{\Pi_{i,j(t)}(\pi)}}. \tag{2.34}$$

La description détaillée de l'algorithme en fonction de ces différents hyperparamètres et fonctions est finalement décrite ci-dessous.

Instructions. Pour k de 0 à $K-1$, répéter les instructions suivantes :

1. Simulation des politiques candidates
   Construire N_k politiques notées $\pi_n\,(n \in [\![1; N_k]\!])$ de sorte que pour chaque n :
   — avec probabilité β_k, tirer les actions de π_n selon la matrice P_0 ;
   — avec probabilité $1-\beta_k$, tirer les actions de π_n selon la matrice P.
2. Évaluation des performances
   Pour chaque $n \in [\![1; N_k]\!]$, calculer par méthode de Monte-Carlo $\bar{V}_k^n$ le coût de la π_n-ième politique avec M_k simulations de trajectoire partant de l'état initial x_0.
3. Calcul de la matrice P_{k+1}
   (a) on calcule d'abord une matrice auxiliaire $\hat{P}_{k+1}$ de la façon suivante : pour tous entiers $1 \leq i \leq \#\mathbb{X}, 1 \leq j \leq \#\mathbb{A}$ et $1 \leq t \leq H$

$$\hat{P}_{k+1}(i,j,t) = \frac{\sum_{n=1}^{N_k} \dfrac{\exp\left(-\bar{V}_k^n T_k^{-1}\right)}{\mathbf{f}\left(\pi_n, P_k\right)} \mathbb{1}_{\Pi_{i,j}(t)}\left(\pi_n\right)}{\sum_{n=1}^{N_k} \dfrac{\exp\left(-\bar{V}_k^n T_k^{-1}\right)}{\mathbf{f}\left(\pi_n, P_k\right)}}. \tag{2.35}$$

   (b) $P_{k+1} \leftarrow \alpha_k \hat{P}_{k+1} + (1-\alpha_k) P_k$.
4. Mise à jour des paramètres de l'algorithme
   $M_{k+1},\ N_{k+1} \leftarrow \max\left(M_0,\ \left\lfloor 1.01 \log^3(k) \right\rfloor\right),\ \max\left(N_0, \lfloor k^{0.501} \rfloor\right)$;
   $\alpha_{k+1},\ \beta_{k+1},\ T_{k+1} \leftarrow \frac{1}{(k+100)^{0.501}},\ \frac{1}{\sqrt{k+1}},\ \frac{T_0}{\log(k+\mathrm{e})}$.

Réécriture des coefficients des matrices $\hat{P}$. Comme pour l'algorithme MRAS, le calcul des matrices $\hat{P}_{k+1}$ occasionne des instabilités numériques via le produit de nombreuses probabilités proches de 0. Aussi, on stabilise ces calculs via une réécriture similaire de ses coefficients, et on pose pour toute itération $k \in [\![0; K-1]\!]$ et pour toute politique $n \in [\![1; N_k]\!]$,

$$c_n^{(k)} = \frac{A_n^{(k)}}{Fk_n^{(k)} + F0_n^{(k)}},$$

avec

$$A_n^{(k)} = \exp\left(-\bar{V}_k^n T_k^{-1}\right), Fk_n^{(k)} = (1-\beta_k)\, f\left(\pi^n, P_k\right) \text{ et } F0_n^{(k)} = \beta_k f\left(\pi^n, P_0\right),$$

de sorte que

$$\hat{P}_{k+1}(i,j,t) = \frac{\sum_{n=1}^{N_k} c_n^{(k)} \mathbb{1}_{\Pi_{i,j}(t)}\left(\pi^n\right)}{\sum_{n=1}^{N_k} c_n^{(k)}}.$$

Comme précédemment pour les coefficients calculés pour la méthode MRAS, la réécriture ainsi que la normalisation des coefficients $c_n^{(k)}$ conduit finalement à des calculs stables.

Exemple 7: Recherche d'une politique optimale avec la méthode ASA.

On va maintenant appliquer la méthode ASA pour optimiser la stratégie de maintenance du système introduit dans l'Exemple 1, et comparer les résultats obtenus avec ceux donnés par l'application de la programmation dynamique, détaillés dans l'Exemple 4. Contrairement à l'Exemple 5 où on s'est restreint à approcher la fonction valeur sur des politiques stationnaires, on rappelle que le cadre général des méthodes MRAS et ASA est de l'approcher sur l'ensemble des politiques non stationnaires admissibles associées au MDP.

L'ensemble des actions possibles par état de ce système est détaillé dans la Table 2.7a de l'Exemple 5. Aussi, la distribution de probabilité $P_0 \in \mathcal{M}_{\#\mathbb{X},\#\mathbb{A},H}([0;1])$ est telle que pour chacune des H dates de décision, les probabilités de choisir une action a dans un état x sont détaillées dans la Table 2.7b de l'Exemple 5. Il existe alors $\left(\prod_{x\in\mathbb{X}} \#\mathbb{A}(x)\right)^H = 18^H$ politiques possibles sur ce problème. On fixe ensuite arbitrairement le nombre initial de politiques candidates, le nombre de trajectoires utilisées pour évaluer le coût d'une politique par méthode de Monte-Carlo, le nombre d'itérations de l'algorithme, et la température initiale à $N_0 = 10,\ M_0 = 10^3,\ K = 20,\ T_0 = 5$. Enfin, d'après [CFHM13], on peut fixer les hyperparamètres $\alpha_0 = \frac{1}{100^{0,501}} \approx 0.1$ et $\beta_0 = 1$.

On rappelle que la méthode ASA tient compte de toutes les politiques candidates simulées à chaque itération pour mettre à jour la distribution $\hat{P}$, contrairement à la méthode MRAS qui se base sur l'évolution d'un quantile de coût pour ne sélectionner qu'une proportion des meilleures d'entre elles. La distribution $\hat{P}$ est alors mise à jour à chaque itération de l'algorithme ASA. Ainsi, on donne pour résultat de l'algorithme ASA le coût de la meilleure politique trouvée via $\hat{P}$: on dit que c'est la politique optimale renvoyée par l'algorithme.

Fonction valeur. La fonction valeur obtenue via l'algorithme ASA est décrite dans la Table 2.11. On note que ces résultats sont les mêmes que ceux déterminés via l'application de la programmation dynamique, décrits dans la Table 2.5 de l'Exemple 4.

TABLE 2.11 – Fonction valeur déterminée via l'algorithme ASA.

État initial	Fonction valeur
$x_0 =$ stable	**36**
$x_0 =$ dégradé	**82**
$x_0 =$ panne	**232**

Exemple 8: Discussions autour de l'Exemple 7.

Politique associée. On explicite ensuite la politique non stationnaire déterminée via la méthode ASA, partant de l'état stable, selon l'état et la date de décision, dans la Table 2.12. Si la fonction valeur calculée via les deux méthodes est la même, la politique optimale trouvée avec la méthode ASA n'est pas celle déterminée via la programmation dynamique. Si les décisions prises sont les mêmes à la dernière date de décision, le comportement observé (en explicitant la politique stationnaire déterminée via la méthode MRAS) dans l'Exemple 6, se retrouve ici, c'est-à-dire que pour plusieurs dates de décisions, l'action à effectuer dans l'état de panne est d'envoyer le système en mission, alors que la programmation dynamique conduit à effectuer un remplacement dans cet état. Ceci s'explique par le fait que le système observe en moyenne 0 panne lorsque la politique indique d'entretenir le système dès l'état dégradé. La méthode ASA étant basée sur la simulation de trajectoires, elle ne peut estimer la fonction valeur qu'à partir d'observations et non de probabilités théoriques, comme l'algorithme MRAS. Les distributions de probabilités sont également similaires à celles observées dans l'Exemple 6 via la méthode MRAS, c'est à dire que les probabilités se concentrent sur une seule action possible pour les états suffisamment observés (stable et dégradé), mais restent uniformément réparties entre plusieurs actions pour les états rarement observés (panne), pour toutes les dates de décisions.

TABLE 2.12 – Détails de l'action choisie par état et par date de décision, pour la politique non stationnaire déterminée via la méthode ASA, partant de l'état stable.

x	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$
S	mission	mission	mission	mission	mission	mission	mission	mission	mission	mission
D	entretenir	entretenir	entretenir	entretenir	entretenir	entretenir	entretenir	entretenir	entretenir	mission
P	mission	mission	remplacer	mission	remplacer	mission	mission	remplacer	mission	remplacer

Comparaison avec la méthode MRAS L'application de la méthode MRAS à ce problème d'optimisation conduit au même coût, avec une politique et un temps de calcul similaires. L'avantage de la méthode MRAS tient à l'évolution d'un seuil de performance, dont l'arrêt des mises à jour peut être utilisé comme critère d'arrêt de l'algorithme. Néanmoins, la méthode MRAS dépend de nombreux hyperparamètres à ajuster, tandis que pour l'algorithme ASA, seule la température initale T_0 reste à calibrer. Pour ces raisons, les deux méthodes présentent aussi bien des avantages que des limites et seront utilisées et à départager pour l'optimisation de la stratégie de maintenance du système industriel étudié dans cette thèse, dont la modélisation est détaillée au Chapitre 3 et la recherche d'une politique optimale est mise en oeuvre au Chapitre 4.

2.3 Quantification d'une variable aléatoire

Dans cette section, on introduit la notion de quantification, en se basant sur [Pag98, Sel05]. L'algorithme de quantification d'une variable aléatoire, qui sera utilisé dans le Chapitre 4 pour permettre la mise en oeuvre de l'approximation numérique de la fonction valeur, dans le cadre de l'optimisation de la stratégie de maintenance du pod, est aussi détaillé.

La quantification est une méthode issue du traitement du signal, développée pour approcher un signal continu par un signal discret de manière optimale. Si à l'origine son étude provient du domaine de la théorie de l'information, du traitement du signal et de la compression (voir [GG92]), ses applications se sont ensuite élargies à différents domaines et notamment aux probabilités numériques.

1.3.1 Principe de la quantification

Soit X une variable aléatoire à valeurs dans $\mathbb{R}^d$, définie sur un espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$, telle que $||X||_p = E[|X|^p]^{1/p} < \infty$. On note $\mathbb{P}_X$ la loi de X supposée simulable. Soit un entier $K > 0$. On appelle K-quantificateur une application borélienne $\pi_\Gamma : \mathbb{R}^d \to \mathbb{R}^d$ dont l'image de $\mathbb{R}^d$ est un ensemble fini $\Gamma = \pi_\Gamma(\mathbb{R}^d) = \{x_1, \ldots, x_K\} \subset \mathbb{R}^d$ appelée *grille de quantification* ou encore ensemble des centres, des points de quantification ou des centroïdes.

Pour définir de manière unique l'application π_Γ, on spécifie une partition borélienne $(C_i)_{1\leq i\leq K}$ de l'espace $\mathbb{R}^d$ qui associe à chaque ensemble C_i un point $x_i \in C_i$ qui est son centroïde associé à π_Γ. On peut ainsi définir π_Γ par :

$$\pi_\Gamma(X) := \sum_{i=1}^{K} x_i \mathbb{1}_{C_i}(X). \tag{2.36}$$

Quand $X \in L^p$, on définit un K-quantifieur L^p-optimal de X par une application $\pi_\Gamma^*(X)$ solution du problème d'optimisation paramétré par la taille de la grille de quantification K :

$$\inf \Big\{ ||X - \pi_\Gamma(X)||_p^p, \pi_\Gamma : \mathbb{R}^d \to \mathbb{R}^d, \text{ application borélienne t.q. } \#(\pi_\Gamma(\mathbb{R}^d)) \leq N \Big\}. \tag{2.37}$$

D'après les résultats de [GL00], ce problème admet toujours une solution π_Γ^* définie d'une part par le choix d'une grille de quantification optimale $\Gamma^* = \{x_1, \ldots, x_K\}$, qui vérifie :

$$E\Big[|X - \pi_\Gamma^*(X)|^p\Big] = E\Big[\min_{x\in\pi_\Gamma^*(\mathbb{R}^d)} |X - x|^p\Big],$$

et d'autre part par une partition optimale $(C_i(\Gamma^*))_{1\leq i\leq K}$, dite de Voronoï, associée à cette grille de quantification, qui définit π_Γ^* comme une projection aux plus proches voisins sur l'ensemble des centroïdes $(x_i)_{1\leq i\leq K}$.

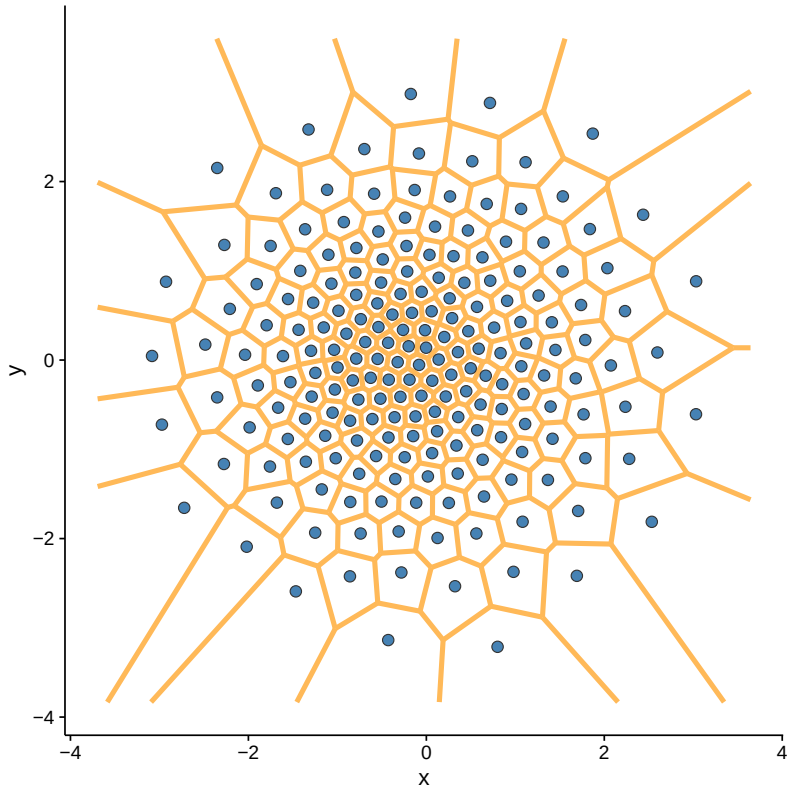


FIGURE 2.4 – Mosaique de Voronoï à 200 points d'une variable aléatoire gaussienne centrée réduite en dimension deux. Les points sont les points de la grille de quantification, et les cellules de Voronoï engendrées par la grille délimitent les plus proches voisins.

Une partition de $\mathbb{R}^d$ vérifiant :

$$(C_i(\Gamma^*)) \subset \{\xi \in \mathbb{R}^d \text{ t.q. } |\xi - x_i| = \min_{1\leq k\leq K} |\xi - x_k|\},$$

est appelée mosaique (ou diagramme) de Voronoi de Γ^*. Les $C_i(\Gamma^*)$, $i = 1, \ldots, K$, sont les cellules de Voronoï engendrées par Γ^*. Ainsi, la grille de quantification $C_i(\Gamma^*)$ consiste en tous les points ξ de R^d tels que x_i est le plus proche voisin de ξ parmi les points x_j de Γ^*. Une telle grille aura plus de points dans les zones de forte densité, et fournit donc une meilleure approximation qu'une discrétisation régulière.

La Figure 2.4 représente un exemple de grille de quantification pour une variable aléatoire gaussienne centrée réduite en dimension 2, où on observe que l'algorithme de quantification place bien plus de points dans la zone centrale qui est de plus forte densité.

La distorsion, qui correspond à l'erreur de projection commise, s'écrit alors :

$$D_N^X = ||X - \pi_\Gamma^*(X)||^p = || \min_{1\leq i\leq K} |X - x_i| \, ||_p^p.$$

La distorsion converge alors vers zéro lorsque le nombre de points K de la grille de quantification tend vers $+\infty$, et sa vitesse de convergence est donnée par le théorème de Zador (voir [PP03, GL00]).

1.3.2 Construction d'une grille de quantification

En pratique, il n'existe pas de forme générale d'une grille optimale, sauf dans quelques cas particuliers (voir par exemple en dimension 1 la méthode de Newton pour la loi normale ou dans le cas d'une loi à densité log-concave méthode de gradient ou point fixe). En dehors de ces cas, on peut utiliser l'algorithme CLVQ (Competitive Learning Vector Quantization), décrit par exemple dans ([BP03, PPP04]). Aussi appelé algorithme de Kohonen, c'est un algorithme d'apprentissage basé sur les simulations et une méthode de gradient stochastique. Il est décrit en détails dans l'Algorithme 2 et ses paramètres sont donnés dans la Table 2.13. Son application requiert une grille initiale, notée Γ^0, et une suite $(\gamma_n)_{n\geq 0}$. Concernant Γ^0, une possibilité est de tirer aléatoirement K points selon la distribution cible ν. Une alternative possible, appelée *splitting-initializing* consiste à initialiser une grille de taille K en rajoutant un point à une grille optimale de taille $K-1$ [PP03]. Un exemple de suite $(\gamma_n)_{n\geq 0}$ les vérifiant pour le cas d'une distribution ν à valeur dans $\mathbb{R}^d$, décrite dans ([PP03]) est donnée dans la Table 2.13.

Pour plus de détails sur la méthode de quantification, voir par exemple [BP03, Pag98].

Algorithme 2 : Méthode CLVQ pour calculer une grille de quantification optimale.

Entrées : Nombre de points K, Nombre de répétitions NR, Séquence (γ_n), Grille initiale (Γ^0) avec K points, Simulateur de la loi cible ν.

```
début
    pour m ← 0 à NR-1 faire
        simuler un point x selon la loi ν
        sélectionner y le plus proche voisin de x dans Γ^m
        poser y' = y − γ_{n+1}(y − x)
        Γ^{m+1} ← Γ^m ∪ {y'}\{y}
    retourner (Γ^{NR})
```

TABLE 2.13 – Paramètres de l'algorithme de quantification CLVQ pour K points dans la grille.

Paramètre	Valeur
Nombre de points dans la grille	K
Nombre d'itérations	NR = K × 100000
Suite $(\gamma_n)_{1\leq n\leq NR}$	$\gamma_n = a(a+b\times n)^{-1}$, $a = 4K^{\frac{1}{d}}$, $b = \pi^2 K^{-\frac{2}{d}}$

2.4 Conclusions

Dans ce premier chapitre, on a défini les processus markoviens décisionnels et le problème d'optimisation associé, c'est-à-dire le calcul de la fonction valeur et d'une politique qui réalise ce coût minimal, pour un critère donné. On a ensuite détaillé plusieurs méthodes numériques d'approximation de la fonction valeur via le critère fini, notamment la programmation dynamique et les méthodes MRAS et ASA, qui peuvent être implémentées dans les cas où l'espace d'états, l'espace d'action et l'horizon sont finis.

On a explicité la modélisation de la dynamique d'un système monolythique (à plusieurs états et sujet à des dégradations et défaillances), par un MDP. L'introduction de cet exemple, dont le noyau de transition est explicite via une matrice de transitions, a permis d'illustrer la mise en oeuvre de ces différentes méthodes et d'interpréter leurs diverses sorties numériques. On a notamment pu comparer plusieurs politiques de références avec les politiques optimales déterminées via ces méthodes. Sur cet exemple minimaliste, l'application des méthodes MRAS et ASA a permis de mettre en évidence que l'estimation d'une distribution de probabilités optimale sur les actions à effectuer par états est limitée pour les états n'étant pas ou que peu observés lors des simulations. Outre cette observation, on peut noter que malgré les garanties théoriques de convergence, l'implémentation de ces méthodes est très sensible aux dimensions du problème, c'est à dire à l'augmentation de la taille de l'espace d'états, d'actions et/ou de l'horizon.

Enfin, lorsque l'espace d'états n'est pas fini, une étape de discrétisation est nécessaire pour appliquer notamment les méthodes MRAS et ASA. Pour ce faire, on a présenté le principe de quantification d'une variable aléatoire, qui sera utilisé en vue d'appliquer ces méthodes à notre problématique industrielle. Aussi, on s'intéresse maintenant à la modélisation de la dynamique d'un équipement optronique produit par Thales, par un MDP à espace d'états continu, dans le Chapitre 3. L'objectif sera alors d'optimiser la stratégie de maintenance du système étudié : le noyau de transition n'étant pas explicite, on cherchera à résoudre ce MDP via l'application de ces méthodes basées sur les simulations, dans le Chapitre 4.

Chapitre 3

Modélisation de la dynamique du pod par un MDP à espace d'états continu

Ce travail de thèse porte sur l'étude de la maintenance d'un pod développé par Thales. Le pod de désignation laser est un équipement optronique aéroporté développé et commercialisé par Thales, qui en assure la maintenance auprès de ses clients. Chaque pod est constitué de plusieurs composants sujets à des détériorations ou défaillances aléatoires pouvant survenir

au cours de missions pour lesquelles il est requis. Aussi, pour enregistrer les temps de fonctionnement de ses composants et évaluer leur état, le pod est équipé de HUMS (Health and Usage Monitoring System). L'objectif industriel est alors de déterminer une politique qui garantit le bon déroulement des missions, tout en minimisant les coûts de maintenance.

Dans ce chapitre, on s'intéresse au fonctionnement et à la maintenance du pod, afin de modéliser la problématique de l'entreprise. Pour ce faire, la première étape est d'expliciter le contexte d'exploitation du pod et définir les coûts générés par les actions de maintenances possibles. Ensuite, l'idée principale de ce travail est de proposer un modèle mathématique pour décrire la dynamique du système, en utilisant le formalisme d'un Processus Markovien Décisionnel. Ainsi, l'objectif est de résoudre le problème d'optimisation associé, c'est-à-dire de déterminer pour chaque date de décision et chaque état, la meilleure action à effectuer pour minimiser le coût de gestion du pod.

Ce chapitre est organisé comme suit. La section 3.1 est consacrée à la description physique du pod, de ses modes de fonctionnement et de panne, ainsi que ses changements de modes. On y présente ensuite les transitions et coûts engendrés par les décisions qui peuvent lui être appliquées. Dans la section 3.2, on définit le modèle MDP correspondant à ce problème d'optimisation de la stratégie de maintenance du pod. Ensuite, on propose plusieurs politiques de référence dans la section 3.3. La simulation de trajectoires du processus, contrôlé par ces politiques, y est détaillée. Enfin, on compare leurs performances et on commente les résultats numériques obtenus.

3.1 Description du pod de désignation laser

Le pod de désignation laser, représenté dans la Figure 3.1, est un système constitué de 8 composants dont la nature est confidentielle. Aussi, cette étude se plaçant avant le déploiement du pod, les lois et les paramètres de fiabilité de ses différents composants sont inconnus. Les valeurs choisies dans ce document sont donc des valeurs arbitraires prises à titre d'exemple pour illustrer les résultats des méthodes mises en œuvre.

3.1.1 États de fonctionnement et de panne

Dans cette section, on s'intéresse à la caractérisation de l'état du pod. On suppose que le système est sujet à des détériorations ou pannes aléatoires de ses composants, uniquement au cours de son fonctionnement. Ainsi, ses composants peuvent chacun être décrits par une *variable discrète* qu'on appellera le *mode*.

Dans notre étude, *trois* modes sont possibles. On dit qu'un composant est dans un mode :

— stable s'il fonctionne normalement,

— dégradé s'il fonctionne mais présente une faiblesse (non perceptible par l'utilisateur),

— panne s'il ne fonctionne plus.

FIGURE 3.1 – Pod de désignation laser

On considère que les différents composants du pod ont des processus de détérioration et de défaillance indépendants (les composants sont stochastiquement indépendants), et tous sont nécessaires au bon fonctionnement du pod, de sorte que le système s'arrête dès la première panne d'un composant. Les modes de chaque composant déterminent alors le mode global du pod.

On dit que le système est globalement dans un mode :

— stable si tous ses composants sont dans le mode stable,
— dégradé si au moins un composant est dégradé et si ceux qui ne le sont pas sont dans le mode stable.
— en panne si au moins un des composants est en panne.

Dans toute la suite, on va distinguer le mode du système selon le mode de chacun de ses composants. On dénombre ainsi 3^n modes possibles, avec n le nombre de composants. Pour un pod à 8 composants, on aurait à considérer $3^8 = 6561$ modes possibles. On choisit pour toute la suite de simplifier le modèle en se restreignant à $n = 3$ composants, et donc à 27 modes. De plus, on suppose que les pannes ne peuvent pas survenir simultanément, et comme le système est arrêté dès la première panne, les modes de pannes multiples ne sont jamais atteints. Pour toute la suite, on choisit de numéroter ces 27 modes dans l'ordre lexicographique décrit Table 3.1, où les 7 lignes en grisé représentent les modes de pannes multiples, non atteignables et par conséquent exclus de l'ensemble des modes du système. Les 20 modes restants constituent l'ensemble des modes du système.

Chaque composant du système est également décrit par une *variable continue*, qui correspond à une *durée de fonctionnement* passée depuis le dernier changement de mode, et est exprimée en *heures*.

TABLE 3.1 – Numérotation et description des combinaisons possibles de modes du système à trois composants, où les lignes grisées représentent les modes non atteignables, exclus de l'ensemble des modes du système.

Mode	Composant 1	Composant 2	Composant 3
0	stable	stable	stable
1	stable	stable	dégradé
2	stable	stable	panne
3	stable	dégradé	stable
4	stable	dégradé	dégradé
5	stable	dégradé	panne
6	stable	panne	stable
7	stable	panne	dégradé
8	stable	panne	panne
9	dégradé	stable	stable
10	dégradé	stable	dégradé
11	dégradé	stable	panne
12	dégradé	dégradé	stable
13	dégradé	dégradé	dégradé
14	dégradé	dégradé	panne
15	dégradé	panne	stable
16	dégradé	panne	dégradé
17	dégradé	panne	panne
18	panne	stable	stable
19	panne	stable	dégradé
20	panne	stable	panne
21	panne	dégradé	stable
22	panne	dégradé	dégradé
23	panne	dégradé	panne
24	panne	panne	stable
25	panne	panne	dégradé
26	panne	panne	panne

Remarque 3.1.1. Bien que les modes non atteingnables ne soient pas visités par le système, il est utile de les considérer pour des raisons numériques. En effet, on encode par la suite les modes stable, dégradé, panne par les nombres 0, 1, 2. Ainsi, on a une correspondance entre le numéro d'un mode (de 0 à 26) et sa décomposition en base 3 qui est le triplet du mode de ses composants (composé de 0, 1, 2). Cette numérotation sera utilisée notamment dans la sous-section 3.3.2 pour décrire comment simuler la dynamique du système contrôlé par différentes politiques de références définies dans la sous-section 3.3.1.

3.1.2 Dynamique de dégradation et de défaillance

On a vu sous-section 3.1.1 que chaque composant peut se trouver dans un mode stable, dégradé, ou en panne. Dans un premier temps, on décrit la dynamique du système au cours de son utilisation, sans considérer de maintenance. Afin de modéliser le passage d'un mode à un autre pour les composants du pod, on utilisera des lois de Weibull, qui modélisent la durée de vie de systèmes notamment qui s'usent au cours du temps.

Définition 3.1.1. Une variable aléatoire continue X suit une loi de Weibull de paramètres $\alpha, \beta > 0$, notée $\mathcal{W}(\alpha, \beta)$, si elle admet pour densité de probabilité la fonction

$$f_X(t) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} \exp\left\{-\left(\frac{t}{\alpha}\right)^{\beta}\right\}\mathbb{1}_{t\in[0;+\infty[}.$$

Soit X une v.a de loi $\mathcal{W}(\alpha, \beta)$, son taux de saut μ est défini, pour tout $t \in \mathbb{R}^+$, par

$$\mu(t) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1}.$$

La loi de Weibull est caractérisée par deux paramètres (α, β) strictement positifs. Le paramètre α est le paramètre d'échelle, qui indique *quand* l'évènement (e.g la panne d'un élément) a le plus de probabilité de se produire, et β est le paramètre de forme, qui dépend de l'évolution du taux de panne, supposée monotone. On distingue trois cas selon que le taux de panne augmente, soit constant, ou diminue au cours du temps.

Si le taux de panne est croissant alors, $\beta > 1$, et on parle d'*usure* : le risque de tomber en panne augmente au cours du temps. C'est typiquement le cas des éléments mécaniques. Lorsque le taux de panne est constant alors, $\beta = 1$, et l'élément est dit *sans mémoire, sans usure*. C'est typiquement le cas des éléments électroniques. Dans ce cas, la probabilité de défaillance suit une loi exponentielle de paramètre $1/\alpha$. Si le taux de panne est décroissant, alors $\beta < 1$, et on parle de *défaillance de jeunesse*.

Dans ce travail, on suppose qu'un composant $1 \leq i \leq 3$ passe d'un mode stable à un mode dégradé selon une distribution de Weibull de paramètres $(\alpha_{s_i}, \beta_{s_i})$, puis d'un mode dégradé à un mode panne selon une distribution de Weibull de paramètres $(\alpha_{d_i}, \beta_{d_i})$. Les valeurs des paramètres de fiabilité sont données Table 3.2, où m_{s_i} et m_{d_i} représentent l'espérance du temps de vie dans les modes stables et dégradés respectivement.

Aussi, un composant dégradé ne peut pas repasser au mode stable sans opération de maintenance, et de la même façon, un composant en panne ne peut pas repasser en mode stable ou dégradé sans intervention. Ainsi, en l'absence de maintenance, le système passe en mode panne et y reste.

La structure du système et les lois de transitions des trois composants sont résumées Figure 3.2. La combinatoire des changements de modes possibles est explicitée Figure 3.3.

TABLE 3.2 – Paramètres de fiabilité des trois composants du système.

α_{s_1}	700	α_{s_2}	1100	α_{s_3}	1500
β_{s_1}	1.1	β_{s_2}	1.4	β_{s_3}	1.25
m_{s_1}	675	m_{s_2}	1002	m_{s_3}	1397
α_{d_1}	508	α_{d_2}	486	α_{d_3}	677
β_{d_1}	2	β_{d_2}	2	β_{d_3}	2
m_{d_1}	450	m_{d_2}	430	m_{d_3}	600

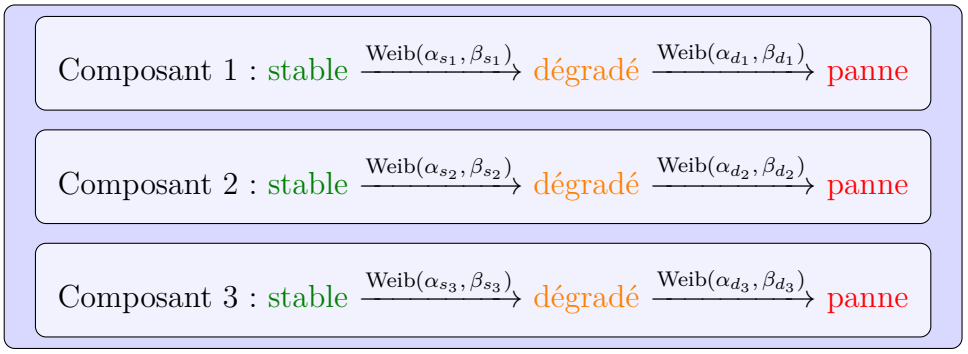


FIGURE 3.2 – Structure du système et lois de changements de modes de ses composants.

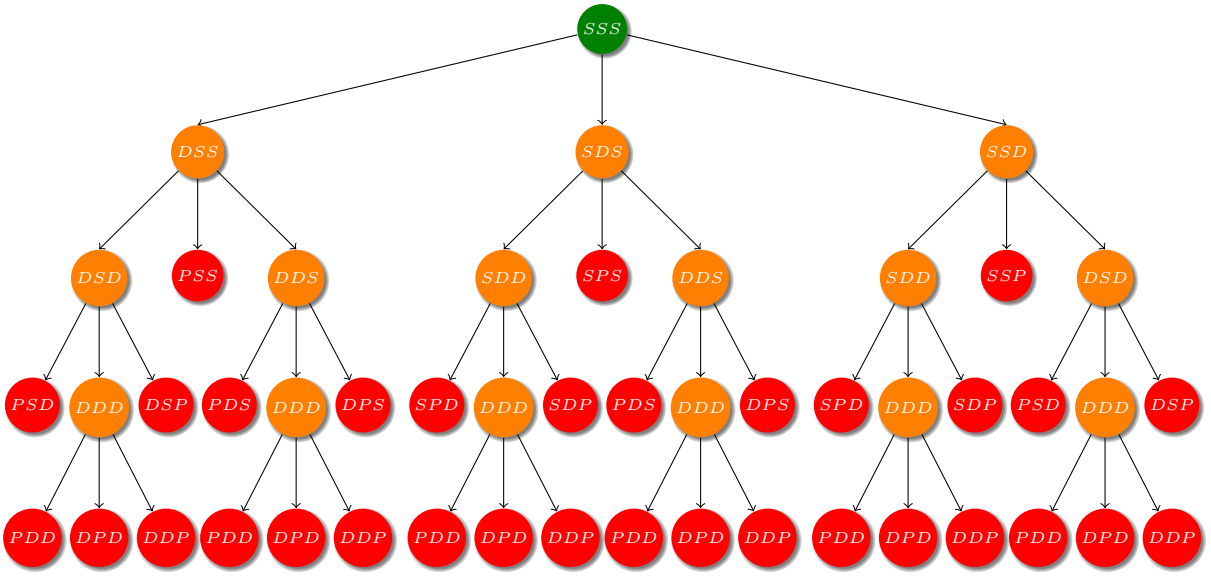


FIGURE 3.3 – Combinatoire des changements de modes possibles du système à trois composants, où les lettres S, D, P désignent respectivement les modes stable, dégradé et panne, et où la couleur du noeud indique le mode global du pod.

Remarque 3.1.2. Les modes de pannes multiples, qui sont en grisé dans la Table 3.1, n'apparaissent pas dans la Figure 3.3 car ils ne sont jamais atteints par le système.

Le système décrit précedemment peut être sollicité pour des périodes de fonctionnement qu'on appellera des *missions*. En pratique, le pod est prévu pour être requis régulièrement pour des missions au cours d'une certaine période d'exploitation, qu'on appelera l'horizon. Dans ce travail, afin de simplifier la modélisation, on suppose que le pod est requis pour des missions dont la fréquence et la durée sont fixées. Par la suite, la fréquence des missions est arbitrairement fixée hebdomadaire, l'horizon est exprimé en semaines et la durée d'une mission est arbitrairement fixée à 40 heures.

Ainsi, l'objectif industriel est de décider chaque semaine si le pod peut effectuer une mission, ou s'il doit être envoyé en atelier pour maintenance, selon son état. Dans le cas d'une visite à l'atelier, il s'agira de choisir des opérations de maintenance différenciées selon l'état de chacun des composants du système. Le choix de ces opérations a pour but de prolonger la durée de vie du système, sans toutefois faire d'opérations trop précoces, afin qu'il puisse assurer ses missions au coût minimal. Enfin, on suppose que les opérations peuvent être faites en parallèle. Ainsi, le temps passé dans l'atelier est fixé à une durée d_a, quelles que soient les maintenances choisies. Cette durée est fixée arbitrairement à $d_a = 4$ semaines, et aucune nouvelle décision n'est prise jusqu'à la sortie de l'atelier.

3.1.3 Actions de contrôles

Pour ce système, la première action possible est de l'envoyer effectuer une mission pour laquelle il est requis chaque semaine.

Ensuite, deux opérations de maintenance parfaites sont possibles dans l'atelier, afin de ramener un composant à neuf, c'est-à-dire dans un mode stable et avec un temps de fonctionnement remis à 0. Ces opérations sont soit un entretien, soit un remplacement. Elles se différencient par le fait que l'entretien est une maintenance préventive, tandis que le remplacement est une maintenance corrective. L'entretien d'un composant en panne sera alors considéré comme physiquement impossible. On considère qu'une visite à l'atelier mobilise tout le système, mais que les opérations de maintenance sont différenciées pour chaque composant. Aussi, lors d'un passage dans l'atelier, on suppose qu'il y a au moins une opération de maintenance, mais que les composants ne subissent pas forcément tous une intervention, et dans ce cas on dira que l'opération effectuée sur un composant non traité est *rien*. Étant donné qu'on ne fait qu'une seule opération de maintenance par composant, c'est à dire qu'on ne fait pas à la fois un entretien et un remplacement sur un même composant, on compte 26 combinaisons possibles de maintenance. Par la suite, une telle combinaison est appelée une *action de maintenance*.

L'ensemble de ces actions est décrit dans la Table 3.3.

Remarque 3.1.3. Ces actions ne sont pas toutes possibles dans tous les états du pod, et certaines (en grisé) sont interdites pour tous les modes atteignables du système. On explicite l'ensemble des actions possibles par état dans la sous-section 3.1.4.

TABLE 3.3 – Numérotation et description des combinaisons possibles d'actions définies pour le système à trois composants, où les lignes grisées représentent les actions qui sont interdites pour l'ensemble des modes atteignables du système.

Action	Composant 1	Composant 2	Composant 3
0	mission	mission	mission
1	rien	rien	entretenir
2	rien	rien	remplacer
3	rien	entretenir	rien
4	rien	entretenir	entretenir
5	rien	entretenir	remplacer
6	rien	remplacer	rien
7	rien	remplacer	entretenir
8	rien	remplacer	remplacer
9	entretenir	rien	rien
10	entretenir	rien	entretenir
11	entretenir	rien	remplacer
12	entretenir	entretenir	rien
13	entretenir	entretenir	entretenir
14	entretenir	entretenir	remplacer
15	entretenir	remplacer	rien
16	entretenir	remplacer	entretenir
17	entretenir	remplacer	remplacer
18	remplacer	rien	rien
19	remplacer	rien	entretenir
20	remplacer	rien	remplacer
21	remplacer	entretenir	rien
22	remplacer	entretenir	entretenir
23	remplacer	entretenir	remplacer
24	remplacer	remplacer	rien
25	remplacer	remplacer	entretenir
26	remplacer	remplacer	remplacer

Remarque 3.1.4. Bien que ces actions ne soient pas toutes autorisées sur le système, il est utile de les considérer pour des raisons numériques. En effet, on va encoder les actions *rien*, entretenir, et remplacer par les nombres 0, 1, et 2. Ainsi, on a une correspondance entre le numéro d'une action (de 0 à 26) et sa décomposition en base 3 qui est le triplet (composé de 0, 1, 2) d'actions faites sur ses composants. Cette numérotation sera utilisée notamment dans la sous-section 3.3.2 pour décrire comment simuler la dynamique du système contrôlé par des politiques de références définies dans la sous-section 3.3.1.

3.1.4 Actions possibles par état

On se propose ici d'expliciter quelles sont les actions possibles selon le mode du système. Les deux familles d'actions définies pour le pod étant d'aller en mission ou d'aller en atelier, on va préciser pour ces deux catégories d'actions, dans quels modes elles sont possibles.

Tout d'abord, on exprime l'ensemble des contraintes liées aux actions de maintenance.

- Une première restriction est liée à l'impossibilité physique d'entretenir un composant en panne. Aussi, l'opération d'entretien est interdite pour un composant en panne.
- Ensuite, afin que chaque passage à l'atelier prolonge effectivement la durée de vie du système, on choisit d'imposer que chaque visite à l'atelier ramène le système en état de fonctionnement. Néanmoins, on n'impose pas que ces visites ramènent nécéssairement le pod dans le mode global stable. Pour un pod ayant par exemple deux composants dégradés et un autre en panne, si on remplace l'élément en panne mais qu'on ne fait pas d'entretien sur les éléments dégradés, le pod sort d'atelier en état dégradé.
- En plus des contraintes physiques, on impose des restrictions liées au coût des interventions. Aussi, on interdit le remplacement d'un composant dégradé, étant donné que l'entretien, supposé moins coûteux sous-section 3.1.6, suffit à le remettre à neuf.
- Enfin, on choisit d'interdire l'envoi en atelier lorsque le pod est stable, et de ne pas autoriser l'entretien ni le remplacement d'un composant stable. Ce choix sera conforté par les performances des politiques de références présentées dans la sous-section 3.3.1, et permet de réduire le nombre d'actions possibles par mode à explorer lors de la recherche d'une politique optimale dans le Chapitre 4.

Si on résume ces contraintes à l'échelle du pod, les actions de maintenance qui opèrent un remplacement sur des composants stable ou dégradé, ou un entretien sur un composant stable ou en panne sont donc interdites. Par ailleurs, les actions de maintenances qui n'opèrent pas de remplacement sur un composant en panne sont interdites car elles ne permettent pas de ramener le pod en état de fonctionnement. Enfin, les pannes multiples étant inatteignables, les actions de maintenance à plusieurs remplacements (grisées dans la Table 3.3) sont interdites.

Ensuite, on explicite les spécificités relatives aux missions.

- L'action d'aller en mission est possible dans les modes stable et dégradés.
- Pour les modes de pannes, le pod ne pouvant pas fonctionner, plusieurs choix de modélisation sont ici possibles. Une possibilité serait d'ajouter une action "ne rien faire", mais ce choix est contraire à l'objectif industriel de réussite de mission, et nécessiterait en plus de lui associer un coût d'indisponibilité afin de pénaliser cette situation. Une alternative, qu'on choisit par la suite, est d'autoriser l'envoi du pod en mission en panne et de modéliser l'échec de mission par un coût défini sous-section 3.1.6.

Remarque 3.1.5. Ces contraintes sur les actions possibles par état sont, pour un mode donné, les mêmes quelque soit la durée de fonctionnement passée dans ce mode.

3.1.5 Dynamique du système contrôlé

On va maintenant décrire les transitions possibles du système, qui dépendent à la fois de l'état du pod et de l'action choisie parmi la Table 3.3. Compte tenu des contraintes sur les actions explicitées dans la sous-section 3.1.4, on distingue à nouveau deux cas.

- Si la décision est une action de maintenance, le système entre en atelier pour une durée d_a quelle que soit l'action de maintenance choisie, et aucune nouvelle décision ni transition n'a lieu jusqu'à la sortie de l'atelier. Aussi, à la suite d'un entretien ou d'un remplacement, les composants sont considérés comme neufs. Ils ressortent donc de l'atelier en mode stable après une durée d_a, et leurs durées de fonctionnement sont remises à 0. Les temps de fonctionnement et les modes des composants non traités sont inchangés.
- Si l'action choisie est d'aller en mission, son issue dépend du mode du pod.

  1. Si le système est en panne avant d'aller en mission, il ne peut pas fonctionner, donc les modes et temps de fonctionnements des composants n'évoluent pas.
  2. Sinon, le système étant dans un mode stable ou dégradé, il peut être sujet à des détériorations et défaillances aléatoires de ses composants, selon la dynamique décrite dans la sous-section 3.1.2. Ainsi, tant que la mission (initialisée à 40h) n'est pas terminée, plusieurs transitions sont encore possibles :

     (a) Si les évènements de défaillances et de détériorations sont ultérieurs à la fin de la mission, les modes des composants ne changent pas, mais leurs temps de fonctionnement évoluent. Ils sont incrémentés de la durée écoulée depuis l'entrée en mission ou la dernière transition, et la mission se termine.

     (b) Sinon, si l'évènement qui survient est une panne, ceci met un terme à la mission. Les temps de fonctionnements sont incrémentés de la durée passée depuis l'entrée en mission ou la dernière transition, puis les modes et temps des composants ne peuvent plus évoluer car le système s'arrête.

     (c) Sinon, l'évènement qui survient est la dégradation d'un composant. Il passe en mode dégradé, avec un temps de fonctionnement dans ce mode égal à 0. Les modes des autres composants sont inchangés et leurs temps de fonctionnement sont incrémentés de la durée écoulée depuis l'entrée en mission ou la dernière transition. Enfin, le système pouvant fonctionner dans un mode dégradé, on retourne au point 2 si la mission n'est pas terminée, puisque les transitions (a), (b) et (c) sont encore possibles.

  On ne sait donc pas combien d'événements vont se produire au cours d'une mission, mais on compte au plus 4 évènements. On dira que le noyau de transition n'est pas explicite analytiquement et c'est une des spécificités de ce système physique.

3.1.6 Fonction de coût hebdomadaire

On s'intéresse maintenant à la caractérisation des coûts générés en fonction des actions choisies sur le système, et des transitions qui peuvent en résulter. Comme on considère deux familles d'actions distinctes, on définit deux types de coûts.

D'une part, le coût d'une visite à l'atelier se décompose en une part fixe, qui est un coût d'immobilisation, et une part variable, qui dépend de la nature et du nombre d'opérations de maintenances effectuées sur les composants du pod. Aussi, on suppose que le coût d'une opération de remplacement est supérieur à celui d'un entretien.

D'autre part, on définit une pénalité d'échec qui doit être payée en cas de mission non réussie, c'est-à-dire si le système part en étant dans un mode de panne ou passe dans un mode de panne au cours d'une mission. Ce coût correspond à une pénalité d'indisponibilité du pod, choisie pour représenter la contrainte d'assurer les missions, et dépend donc uniquement de l'issue de la mission. Par ailleurs, on considère qu'il n'est pas à payer lorsque le système se trouve dans l'atelier. Enfin, on considère qu'une mission réussie entraîne un coût nul.

Les valeurs des coûts, définis dans une unité arbitraire, sont donnés dans la Table 3.4. Ils ont été calibrés afin d'illustrer un compromis entre intervenir trop tôt sur le système, ce qui engendre des coûts de maintenance prématurés trop fréquents, et intervenir trop tard, exposant à des maintenances plus coûteuses et à des pénalités d'échec. Plus précisément, ils ont été determinés en comparant les coûts de plusieurs politiques de références, qui seront présentées dans la section 3.3.

Immobilisation	$c_i = 50$
Remplacement	$c_r = 50$
Entretien	$c_e = 10$
Échec	$c_f = 20$

TABLE 3.4 – Coûts des opérations de maintenance et de la pénalité d'échec de mission.

Dans cette section, on a défini un modèle physique contrôlé, pour lequel on peut proposer des stratégies de gestion du pod. Ces stratégies peuvent, par exemple, associer à chaque mode du pod une action à effectuer pour chaque décision. En simulant une trajectoire du système contrôlée par une stratégie, on peut calculer la somme des coûts générés jusqu'à l'horizon d'étude. Ainsi, en calculant le coût moyen associé à chacune d'elle, par méthode de Monte-Carlo, on peut comparer des stratégies entre elles. L'objectif est alors de trouver un coût moyen minimum, et une stratégie réalisant ce minimum. Pour ce faire, on choisit de modéliser la dynamique du pod via le formalisme d'un processus markovien décisionnel.

3.2 Modèle mathématique

Cette section se concentre sur la définition d'un processus markovien décisionnel modélisant la dynamique du système physique décrit section 3.1. Cette famille de processus permet de formaliser le problème d'optimisation de la maintenance du pod, ainsi que l'ensemble des politiques sur lequel on cherche un optimum, mais fournit également des méthodes numériques pour rechercher un optimum et une politique associée. En particulier, on définit ce processus en vue d'appliquer la méthode MRAS, détaillée dans la sous-section 2.2.2.

Dans la section 3.1, on a supposé une fréquence des missions hebdomadaire, et une durée d'atelier supérieure à cette fréquence. Pour un *horizon* d'étude exprimé en semaine, la dynamique décrite sous-section 3.1.5 dépend d'au plus *horizon* décisions, puisqu'aucune décision n'est prise dans l'atelier. L'application de MRAS pour répondre à notre problématique requiert, entre autres hypothèses, de définir une fréquence des décisions fixe. L'ajout d'une variable d'état et d'une action sont alors nécessaires pour caractériser la dynamique du pod, mais ceci permet de construire et comparer des politiques ayant le même nombre de décisions, dans le but d'appliquer la méthode MRAS. On choisit de garder la fréquence des missions comme fréquence des décisions. Ainsi, avec le MDP, on a exactement *horizon* décisions.

3.2.1 Espace d'états du système

On a vu au paragraphe 3.1.1 que chaque composant du pod peut être décrit par un mode et un temps de fonctionnement passé dans ce mode. Pour chacun de ses modes, le système peut se trouver soit en atelier pour maintenance, soit en dehors de l'atelier.

La fréquence des décisions étant inférieure à la durée passée dans l'atelier, on ajoute aux caractéristiques du système une **variable discrète** qui compte le temps passé dans l'atelier, notée t_a et exprimée en **semaines**. On fixe $t_a = 0$ si le système est en dehors de l'atelier, et $t_a \in \{1, \dots, d_a - 1\}$ sinon. L'ajout de ce compteur permet également de conditionner les actions possibles selon le nombre de semaines passées dans l'atelier, notamment d'y rester tant qu'il est inférieur à la durée fixée d'une visite à l'atelier.

Pour définir $\mathbb{X}$ l'espace des états du système, on note :

— $\mathcal{M}$: le sous-ensemble des 20 modes atteignables parmi les 27 modes énumérés dans la Table 3.1.

— $\mathcal{T} = \{(t_1, t_2, t_3), t_i \in \mathbb{R}_+, 1 \leq i \leq 3\}$, l'ensemble des temps de fonctionnement des trois composants, exprimés en heures.

— t_a le temps passé dans l'atelier exprimé en semaine, où $t_a \in [\![0, d_a - 1]\!]$.

Finalement, on a $\mathbb{X} = \Big(\mathcal{M} \times \mathcal{T} \times [\![0, d_a - 1]\!]\Big)$ et un état du système $x \in \mathbb{X}$ est de la forme $x = (m, t_1, t_2, t_3, t_a)$.

Parmi les 20 modes possibles du pod, on distingue 3 sous-ensembles, et on note :

— $\mathcal{S}$: l'ensemble du mode stable. Cet ensemble n'est constitué que d'un seul mode, où tous les composants du système sont dans le mode stable.

— $\mathcal{D}$: l'ensemble des modes dégradé, qui concerne les modes pour lesquels au moins un composant est dégradé mais aucun composant n'est en panne.

— $\mathcal{P}$: l'ensemble des modes de panne. Le troisième ensemble correspond aux modes du système pour lesquels un des composants est en panne.

3.2.2 Espace d'actions

On détermine maintenant l'ensemble des actions définies pour contrôler le système. Lorsque le système se trouve dans l'atelier, on spécifie une action qui est de '*rester en maintenance dans atelier*' pour une semaine supplémentaire, et qu'on note en maintenance. À l'inverse, si le pod se trouve hors de l'atelier, il peut-être à nouveau controlé selon les actions introduites dans la sous-section 3.1.3.

Ainsi, pour définir $\mathbb{A}$ l'espace des actions de contrôle du système, on note :

— mission l'action correspondant à un envoi en mission.

— $\mathcal{C} \subset \{(a_1,\ a_2,\ a_3),\ a_i \in \{\text{rien, entretenir, remplacer}\},\ 1 \leq i \leq 3\}$ l'ensemble des 19 actions de maintenances possibles en atelier, parmi les 27 combinaisons de maintenances énumérées dans la Table 3.3.

— en maintenance l'action consistant à rester en atelier une semaine supplémentaire.

Finalement, $\mathbb{A} = \{$ mission $\} \cup \mathcal{C} \cup \{$ en maintenance $\}$.

Ensemble $\mathbb{A}(x)$. L'ensemble $\mathbb{A}(x)$ des actions possibles dans l'état x est tel que :

— Lorsque le système se trouve dans l'atelier, c'est-à-dire tant que $1 \leq t_a \leq d_a - 1$, la seule action possible sur le pod est de rester en maintenance une semaine supplémentaire, et cette action est impossible en dehors de l'atelier.

— Lorsque le système est en dehors de l'atelier, c'est-à-dire tant que $t_a = 0$, l'action de rester en maintenance est interdite, et l'ensemble des actions possibles par mode hors atelier est explicité en détails dans la sous-section 3.1.4.

3.2.3 Noyau de transition

On s'intéresse ici à la description des transitions possibles du système, qui dépendent de l'état du pod et de l'action a choisie. Au vue des contraintes sur les actions définies précedemment, on va distinguer trois cas :

▷ Si $a =$ mission, l'ensemble des transitions possibles selon le mode du système sont celles précedemment décrites dans la sous-section 3.1.5 et qui sont ensuite reprises dans l'Algorithme 3.

- ▷ Si $a \in \mathcal{C}$, plusieurs choix de modélisation sont possibles puisque la durée d'atelier est fixée à d_a semaines. En effet, même si les opérations sont physiquement faites pendant toute la durée de la visite à l'atelier, on choisit pour simplifier la modélisation qu'elles soient faites sur une seule semaine. On choisit arbitrairement que les maintenances soient faites la première semaine passée dans l'atelier, ce qui incrémente le compteur de $t_a = 0$ à $t_a = 1$. Ainsi, dès la fin de la première semaine, un composant entretenu ou remplacé est considéré comme neuf, avec une durée de fonctionnement en mode stable remise à 0. Les temps de fonctionnement et les modes des composants non traités sont inchangés. Le déroulement des différentes opérations selon l'action de maintenance choisie est décrit dans l'Algorithme 4.
- ▷ Si $a =$ en maintenance, les modes et les temps de fonctionnement des composants sont inchangés, et seul le temps passé dans l'atelier évolue, avec $t_a = t_a + 1$. Lorsque le temps passé dans l'atelier atteint la durée fixée d'une visite à l'atelier, c'est-à-dire que $t_a = d_a - 1$, on considère que le système sera sorti d'atelier pour la prochaine décision, et réinitialise le compteur t_a à 0. La transition qui resulte du choix de cette action est décrite dans l'Algorithme 5.

3.2.4 Fonction de coût

La fonction de coût hebdomadaire $c : \mathbb{X} \times \mathbb{A} \times \mathbb{X} \to \mathbb{R}_+$ dépend de l'état courant du système $x = (m, t_1, t_2, t_3, t_a)$, ainsi que de l'action choisie a et de l'état du système à l'issue de la décision $x' = (m', t'_1, t'_2, t'_3, t'_a)$. On note n_e le nombre d'entretien et n_r le nombre de remplacement effectués pour une action de maintenance donnée. En reprenant les notations de la Table 3.4, on définit le coût hebdomadaire en distinguant trois cas :

- ▷ si $a =$ mission, $c(x, a, x') = c_f * \mathbb{1}_{\{m' \in \mathcal{P}\}}$.
- ▷ si $a \in \mathcal{C}$, $c(x, a, x') = c_i + c_r * n_r + c_e * n_e$.
- ▷ si $a =$ en maintenance, $c(x, a, x') = 0$.

Aussi, on ajoute un coût terminal $C : \mathbb{X} \to \mathbb{R}_+$ tel que $\forall\, x \in \mathbb{X},\ C(x) = 0$.

Dans cette section, on a défini un MDP modélisant la problématique d'optimisation de maintenance du pod. Le calcul de la fonction valeur et d'une politique optimale ne sont pas simples dans notre étude car l'espace d'état $\mathbb{X}$ est hybride et le noyau de transition n'est pas explicite. L'application de la programmation dynamique (voir 2.2.1) n'est donc pas possible. Des méthodes possibles sont alors MRAS et ASA (voir 2.2.2 et 2.2.3), qui ne nécessitent pas d'avoir un noyau explicite, mais requiert que l'espace d'états $\mathbb{X}$ soit discret. La première étape pour résoudre notre problème d'optimisation consiste à discrétiser l'espace des états, en vue d'utiliser cette méthode. Pour ce faire, le calcul des coûts de politiques de références permettra de calibrer les paramètres de l'étape de discrétisation.

3.3 Politiques de référence et simulations

Dans cette section, on s'intéresse à la simulation de trajectoires du processus markovien décisionnel défini dans la section 3.2, et contrôlé par plusieurs politiques de référence données. En effet, pour construire une trajectoire de ce processus, il est nécessaire de définir une politique qui associe à chaque état du système et à chaque date de décision, une action à effectuer sur le pod.

On définit plusieurs politiques de référence dans la sous-section 3.3.1, pour contrôler le système. Dans un premier temps, on introduit quatre politiques dont les décisions prises se basent uniquement sur les modes des composants du pod et le temps d'atelier. Ensuite, on s'intéresse aussi à une politique paramétrique qui tient également compte du temps de fonctionnement des composants. La simulation de trajectoires du processus, contrôlé par ces politiques, est détaillée dans la sous-section 3.3.2. Enfin, on compare les performances de ces politiques en terme de coût dans la sous-section 3.3.3 et on commente les résultats numériques obtenus.

3.3.1 Définition des politiques de référence

Politique de référence π_0. Cette politique de référence est une politique sans intervention de maintenance (ni remplacement, ni entretien) pendant toute la période étudiée.

Elle est définie telle que, pour toutes les dates de décision de l'étude, et pour tous les états hors atelier du système, l'action choisie est un envoi en mission du pod. Le système n'étant jamais envoyé dans l'atelier, seule l'action mission est choisie.

La dynamique du processus contrôlé par la politique de référence π_0 est décrite dans l'Algorithme 6.

Ainsi, le système va évoluer du mode stable aux modes dégradé puis au mode panne. En l'absence de maintenance, le système reste en panne jusqu'à la fin de l'étude et ne peut plus réussir de mission, ce qui génère une pénalité d'échec pour chaque fois où la mission n'est pas remplie.

L'évolution des modes du système, lorsqu'il est contrôlé par la politique π_0, est illustrée dans la Figure 3.4.

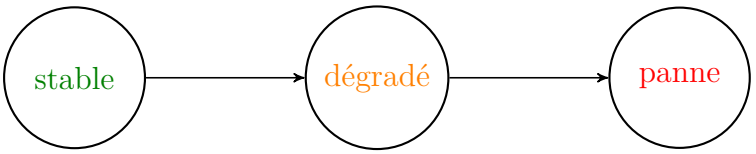


FIGURE 3.4 – Dynamique du processus contrôlé par la politique de référence π_0.

Politique de référence π_1. Cette politique de référence est une politique dite de maintenance corrective *ans action de maintenance préventive*. Elle autorise les interventions de maintenance, mais uniquement lorsque le système se trouve dans un mode de panne.

Ainsi, pour toutes les dates de décisions où le système se trouve dans un état stable ou dégradé, avec $t_a = 0$, l'action choisie est un envoi en mission.

Ensuite, à la première date de décision après l'entrée dans le mode panne, l'action choisie est l'action de maintenance telle que le composant en panne est remplacé et les composants en état stable ou dégradé sont inchangés, au premier temps d'atelier.

Enfin, lorsque le système se trouve dans l'atelier, la seule action possible est de rester en maintenance dans l'atelier jusqu'à écoulement de la durée d_a.

La dynamique du processus contrôlé par la politique π_1 est décrite dans l'Algorithme 7.

Politique de référence π_2 : est une politique de maintenance corrective *avec actions opportunistes de maintenances préventives*. À la différence de la politique π_1, elle permet les actions de maintenance sur des composants en mode dégradé, mais uniquement lorsque le système est dans un mode de panne, après la défaillance d'un des composants.

Ainsi, pour toutes les dates de décisions où le système se trouve dans un mode stable ou dégradé, avec $t_a = 0$, l'action choisie est un envoi en mission.

Ensuite, à la première date de décision après l'entrée dans le mode panne, l'action choisie est l'action de maintenance telle que le composant en mode panne est remplacé, et chaque composant dégradé est entretenu.

Enfin, lorsque le système se trouve dans l'atelier, la seule action possible est de rester en maintenance dans l'atelier jusqu'à écoulement de la durée d_a.

La dynamique du processus contrôlé par la politique π_2 est décrite dans l'Algorithme 8. L'évolution des modes du pod sous les politiques π_1 et π_2 est illustrée dans la Figure 3.5.

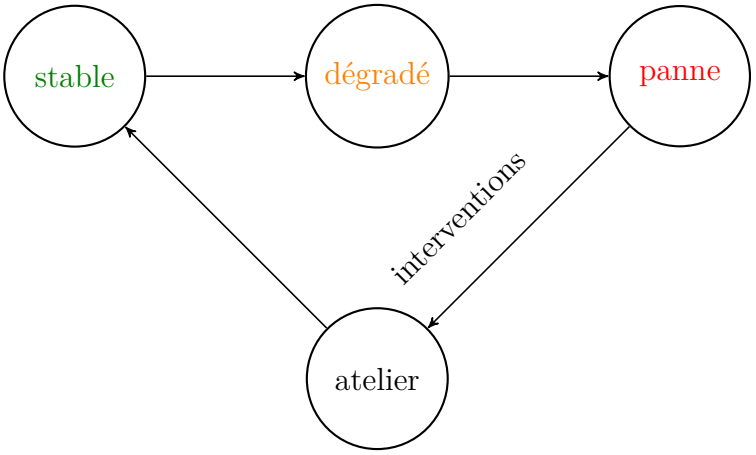


FIGURE 3.5 – Dynamique du processus contrôlé par les politiques π_1 et π_2.

Politique de référence π_3 : est une politique de maintenance préventive. Contrairement aux politiques π_0 à π_2, elle autorise l'envoi en atelier du pod dès qu'il se trouve dans un mode dégradé, sans attendre la panne. Néanmoins, il peut arriver qu'un composant passe d'un mode stable à un mode dégradé puis de panne au cours d'une même mission.

Ainsi, pour toutes les dates de décisions où le système se trouve dans un état stable, avec $t_a = 0$, la seule action possible est un envoi en mission.

Ensuite, à la première date de décision après l'entrée dans le mode panne ou dégradé, l'action choisie est un envoi à l'atelier. Chaque composant en mode panne y est remplacé, et chaque composant dégradé y est entretenu, au premier temps d'atelier.

Enfin, lorsque le pod est à l'atelier, l'action est de rester en maintenance à l'atelier jusqu'à écoulement de la durée d_a.

La dynamique du processus contrôlé par la politique π_3 est décrite dans l'Algorithme 9. Les politiques de références π_0 et π_3 sont détaillées en Annexe A.1, Table A.1.

Politique dépendant des temps de fonctionnement π_4 : est une politique préventive qui permet l'envoi en atelier du pod dans un mode panne, ou dans un mode dégradé avec un temps de fonctionnement supérieur à un certain seuil, qu'on notera s_p.

Ainsi, pour toutes les dates de décisions où le système se trouve dans un état stable, ou dans un état dégradé avec un temps de fonctionnement inférieur à s_p, avec $t_a = 0$, la seule action possible est un envoi en mission.

Ensuite, à la première date de décision après l'entrée en panne, ou dans le mode dégradé avec au moins un composant dégradé dont le temps de fonctionnement est supérieur ou égal au seuil s_p, l'action choisie est un envoi à l'atelier. Chaque composant en mode panne y est remplacé, et chaque composant en mode dégradé est entretenu, quel que soit son temps de fonctionnement, au premier jour d'atelier.

Enfin, lorsque le pod est à l'atelier, l'action est de rester en maintenance à l'atelier jusqu'à écoulement de la durée d_a.

La dynamique du processus contrôlé par la politique π_4 décrite dans l'Algorithme 10. L'évolution des modes du pod sous les politiques π_3 et π_4 est illustrée dans la Figure 3.6.

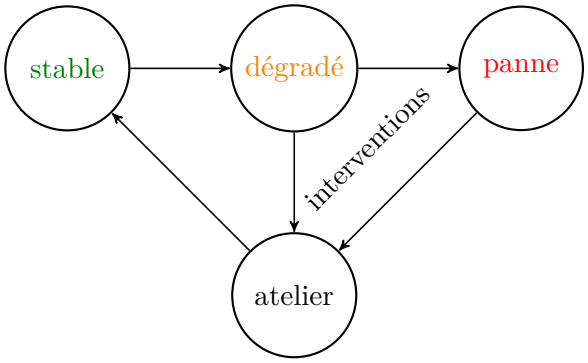


FIGURE 3.6 – Dynamique du processus contrôlé par les politique π_3 et π_4.

Choix des seuils pour la politique π_4 : Bien que d'autres moyens soient possibles pour définir un seuil, on choisit pour toute la suite de le déterminer comme un quantile du temps de fonctionnement passé dans le mode dégradé. Cette politique dépend donc d'un paramètre $p \in [0, 1]$, qui permet de calculer le quantile correspondant au temps de fonctionnement à partir duquel le composant a fonctionné pendant au moins p % de sa distribution.

Aussi, on choisit pour toute la suite que le paramètre p soit commun à l'ensemble des composants, mais on décide de differencier le seuil pour chaque composant en fonction de ses paramètres de fiabilités. On rappelle que pour chacun des composants du pod, le temps passé dans le mode dégradé est distribué selon une loi de Weibull de paramètres $(\alpha_{s_i}, \beta_{s_i})$, donnés Table 3.2.

Pour la loi de Weibull de paramètres (α, β), on a une forme explicite de la fonction quantile :

$$f_q(p) = -\alpha * \log(1-p)^{\beta}, \quad p \in [0, 1]. \tag{3.1}$$

Le seuil s_p est ensuite donné par la relation $s_p = f_q(p)$.

Remarque 3.3.1. On peut noter que, du point de vue des MDP, chacune des politiques de références π_0 à π_4 est *déterministe*, puisqu'une seule action est possible pour chaque état du système. Elles sont également *stationnaires*, car les décisions possibles dépendent de l'état du système au moment de la décision, mais sont les mêmes quelque soit la date de décision, et *markoviennes*, car elles ne dépendent que de l'état courant du système au moment de la décision, et non des états précédents.

Remarque 3.3.2. Du point de vue de l'application, les politiques π_0 à π_3 associent la même action à tous les composants dans un même mode. En particulier, la politique π_3 ne tient pas compte de la durée de fonctionnement passée dans le mode dégradé, alors que la distribution du temps de vie restant dans ce mode avant la panne varie pourtant selon ses paramètres de fiabilités, différents pour chaque composant. En revanche, la politique π_4, en tenant compte du temps de fonctionnement passé dans le mode dégradé et en conditionnant les actions de maintenance au dépassement d'un seuil propre à chaque composant, permet en partie de pallier cette limite. Néanmoins, rien ne garantit que le quantile donnant un seuil optimal soit le même pour tous les composants.

La recherche d'une politique paramétrique stationnaire optimale, paramétrée par un seuil différencié pour chaque composant ne sera pas traitée dans ce travail. Par contre, l'application de la méthode MRAS permet de rechercher une politique optimale non stationnaire, qui tient compte du temps de fonctionnement en mode dégradé différencié pour chaque composant, ce qui permet de chercher sur un espace de politiques plus grand et donc a priori d'obtenir de meilleures performances.

3.3.2 Simulations de trajectoires contrôlées

Maintenant, on s'intéresse à la simulation de trajectoires contrôlées par ces différentes politiques. On rappelle les valeurs fixées arbitrairement pour toutes les simulations à :

- `da` $= 4$: la durée fixée d'une visite à l'atelier, exprimée en semaine ;
- `dm` $= 40$: la durée initiale d'une mission, exprimée en heures ;
- `ci` $= 50$: le coût d'immobilisation lors d'un passage à l'atelier ;
- `cr` $= 50$: le coût d'un remplacement opéré lors du passage à l'atelier ;
- `ce` $= 10$: le coût d'un entretien opéré lors du passage à l'atelier ;
- `cf` $= 20$: le coût de la pénalité d'echec en cas de panne.

Ensuite, on explicite les variables qui sont nécessaires pour caractériser l'état du système et sa dynamique, selon les politiques de références. On note :

- `m` $\in \mathcal{M}$: le mode courant, parmi les 20 modes possibles décrits Table 3.1
- `t = (t1, t2, t3)` $\in \mathcal{T}$: le temps de fonctionnement passé dans le mode courant de chaque composant ;
- `ta` $\in [\![0,$ `da-1`$]\!]$ le temps passé dans l'atelier, exprimé en semaine ;
- `a` $\in \mathbb{A}$: l'action courante choisie sur le système.

Remarque 3.3.3. Par la suite, on ne tire pas pour chaque itération du noyau un temps de vie restant *conditionnellement* au temps de vie passé, mais on tire uniquement un temps de vie par mode par composant selon les paramètres des lois de Weibull Table 3.2.

En plus des variables précédemment définies, on ajoute des variables supplémentaires utiles à la simulation. On note :

- `tr = (tr1, tr2, tr3)` $\in \mathbb{R}_+^3$: le temps de vie restant dans le mode courant de chaque composant ;
- `dcm` $\in \mathcal{M}$: la décomposition du numéro de mode du pod `m` en base 3 (nombre de modes), tel qu'explicité dans la Remarque 3.1.1 ;
- `dca` $\in \mathcal{C}$: la décomposition du numéro de l'action de maintenance à effectuer sur le pod `m` en base 3 (nombre d'actions), tel qu'explicité dans la Remarque 3.1.4 ;
- `cout` $\in \mathbb{R}_+$: le coût cumulé sur une trajectoire depuis le début de vie du système ;
- `tm` $\in [\![0,$ `dm`$]\!]$: le temps restant à accomplir en mission, exprimé en heures ;
- `seuils` $\in \mathbb{R}_+^3$: seuils des temps de fonctionnement en état dégradé conditionnant les actions de maintenance pour la politique à seuils π_4.

Remarque 3.3.4. Le vecteur `dcm` permet la mise à jour du mode du pod `m` simplement via celle du mode d'un composant, c'est-à-dire le remettre à 0 en cas d'opération et l'incrémenter de 1 en cas de dégradation ou panne, puis calculer `m` en base 10 via `dcm`. Aussi, ce vecteur permet de savoir si le mode global `m` $\in \mathcal{S}, \mathcal{D}$ ou $\mathcal{P}$, simplement en calculant le maximum du vecteur `dcm`, qui vaut 0, 1 ou 2, comme expliqué dans la Remarque 3.1.1.

3.3.2.1 Scripts de l'évolution de l'état du système selon l'action choisie

Maintenant, on présente le pseudo-code de l'évolution de l'état du pod lorsque l'action choisie est une mission, dans l'Algorithme 3. Étant donné que la nature des composants du pod est confidentielle, ils sont par la suite désignés par leur numéro, de 1 à 3. Aussi, la description des changements d'états du système se fait en cherchant l'indice du composant concerné.

En particulier, au cours d'une mission on cherche le temps du premier évènement (ligne 9), et l'indice qui réalise cette première transition (ligne 8). Cet évènement est soit la fin de la mission, soit un changement de mode du système. Ensuite, on rappelle que l'ensemble des transitions possibles est décrit en détails dans la sous-section 3.1.2.

Algorithme 3 : Transition lorsque l'action choisie est une mission

Entrées : x=(m, t1, t2, t3, ta), cout ; tr

```
1  début
2      si m ∈ 𝒫 alors
3          cout ← cout + cf
4      sinon
5          tm ← dm ;            // Durée initiale d'une mission : dm = 40 heures
6          dcm ← base_3(m) ;                  // Décomposition du mode m en base 3
7          tant que tm > 0 faire
8              idx ← argmin(tr1, tr2, tr3, tm) ;      // Sélectionner indice évènement
9              evt ← min(tr1, tr2, tr3, tm) ;          // Sélectionner temps évènement
10             t ← t + evt
11             tr ← tr - evt
12             si evt = tm alors
13                 tm ← 0 ;                // Fin de mission sans changement de mode
14             sinon
15                 dcm[idx] ← dcm[idx] + 1
16                 si dcm[idx] = 2 alors                                  // i.e. m ∈ 𝒫
17                     cout ← cout + cf
18                     tm ← 0
19                 sinon                                  // dcm[idx] = 1, i.e m ∈ 𝒟
20                     tr[idx] ← 𝒲(α_{d_idx}, β_{d_idx}) ;      // Tirer temps de vie mode dégradé
21                     tm ← tm - evt
22     m ← base_10(dcm) ;                  // Traduction du vecteur dcm en base 10
```

Sorties : x=(m, t1, t2, t3, ta), cout ; tr

Ensuite, on détaille le script des transitions possibles du pod quand l'action choisie est une action de maintenance dans l'atelier, dans l'Algorithme 4. Il se base principalement sur la décomposition en base 3 de l'action a choisie sur le système (`dca`, ligne 3), et la décomposition en base 3 du mode `m` courant (`dcm`, ligne 2), ce qui permet ensuite de procéder aux opérations de maintenance composant par composant sur le vecteur des modes `dcm`.

Algorithme 4 : Déroulement des opérations en première semaine d'atelier

Entrées : x=(m, t1, t2, t3, ta), a, cout ; tr

1 **début**
2 dcm $\leftarrow base_3$(m)
3 dca $\leftarrow base_3$(a) ; // Décomposition de l'action a en base 3
4 idxe $\leftarrow \{1 \leq j \leq 3 : \text{dca[j]}=1\}$; // Indices composants à entretenir
5 idxr $\leftarrow \{1 \leq j \leq 3 : \text{dca[j]}=2\}$; // Indices composants à remplacer
6 cout $\leftarrow$ cout + ci
7 ta $\leftarrow$ 1
8 **pour** idx **dans** idxe **faire** // Éventuelles opérations d'entretien
9 dcm[idx] $\leftarrow$ 0
10 tr[idx] $\leftarrow \mathcal{W}(\alpha_{s_{\text{idx}}}, \beta_{s_{\text{idx}}})$
11 t[idx] $\leftarrow$ 0
12 cout $\leftarrow$ cout + ce
13 **pour** idx **dans** idxr **faire** // Éventuelles opérations de remplacement
14 dcm[idx] $\leftarrow$ 0
15 tr[idx] $\leftarrow \mathcal{W}(\alpha_{s_{\text{idx}}}, \beta_{s_{\text{idx}}})$
16 t[idx] $\leftarrow$ 0
17 cout $\leftarrow$ cout + cr
18 m $\leftarrow base_{10}$(dcm) ; // Traduction du vecteur dcm en base 10

Sorties : x=(m, t1, t2, t3, ta), cout ; tr

Enfin, on détaille l'évolution du compteur de temps passé dans l'atelier `ta`, quand l'action choisie est de rester en maintenance, dans l'Algorithme 5.

Algorithme 5 : Transition lorsque l'action choisie est en maintenance

Entrées : x=(m, t1, t2, t3, ta)

1 **début**
2 ta $\leftarrow$ ta + 1 ; // Rester une semaine dans l'atelier
3 **si** ta = da **alors**
4 ta $\leftarrow$ 0 // Réinitialiser le compteur ta ; sortir d'atelier

Sorties : x=(m, t1, t2, t3, ta)

3.3.2.2 Scripts de construction de trajectoires selon la politique choisie

Les Algorithmes 3 à 5 décrivant l'évolution du sytème pour chaque état et pour chaque action possibles étant explicités, on présente maintenant comment simuler une trajectoire du processus contrôlé par chacune des politiques de référence, en y faisant référence dans les scripts via les termes *mission*, *operations* et *maintenance* respectivement.

Par la suite, les simulations sont décrites en partant de l'état neuf hors atelier, c'est à dire avec le mode m , les temps de fonctionnement `t`, le `cout` ainsi que le compteur `ta` sont initialisés à 0 (ligne 2), et on initialise ensuite les temps de vie restants `tr` (ligne 4).

Ci-dessous on décrit la dynamique du pod sous la politique π_0, dans l'Algorithme 6, où la seule action possible à chaque date de décision est d'aller en mission.

Algorithme 6 : Dynamique du processus contrôlé par la politique π_0

Entrées : horizon

```
1  début
2      m, ta, cout, t ← 0, 0, 0, 0_3  ;                          // Initialisation
3      pour i de 1 à 3 faire          // Tirer temps de vie restant en mode stable
4          tr[i] ← W(α_{s_i}, β_{s_i})
5      pour date de décision de 0 à horizon-1 faire
6          (x, cout ; tr) ← mission (x, cout ; tr) ;              // voir Algorithme 3
```

Sorties : cout

On détaille ensuite Algorithme 7 la dynamique du pod contrôlé par la politique π_1.

Algorithme 7 : Dynamique du processus contrôlé par la politique π_1

Entrées : horizon

```
1  début
2      m, ta, cout, t ← 0, 0, 0, 0_3 ;                           // Initialisation
3      pour i de 1 à 3 faire          // Tirer temps de vie restant en mode stable
4          tr[i] ← W(α_{s_i}, β_{s_i})
5      pour date de décision de 0 à horizon-1 faire
6          si ta = 0 alors
7              si m ∉ P alors                          // i.e. mode stable ou dégradé
8                  (x, cout ; tr) ← mission(x, cout ; tr) ;          // voir Algorithme 3
9              sinon                                                  // i.e. mode panne
10                 a tel que remplacer composant en panne
11                 (x, cout ; tr) ← opérations(x, a, cout ; tr) ;    // voir Algorithme 4
12         sinon                                                // i.e. en maintenance
13             (x, cout ; tr) ← maintenance(x, a, cout ; tr) ;       // voir Algorithme 5
```

Sorties : cout

Ci-après, on décrit dans l'Algorithme 8 comment simuler une trajectoire du processus contrôlée par la politique de maintenance corrective π_2.

Algorithme 8 : Dynamique du processus contrôlé par la politique π_2

Entrées : horizon

début

  m, ta, cout, t $\leftarrow$ 0, 0, 0, 0_3 ; // Initialisation

  pour i **de** 1 **à** 3 **faire** // Tirer temps de vie restant en mode stable

    tr[i] $\leftarrow \mathcal{W}(\alpha_{s_i}, \beta_{s_i})$

  pour date de décision **de** 0 **à** horizon-1 **faire**

    si ta $= 0$ **alors**

      si m $\notin \mathcal{P}$ **alors** // i.e. mode stable ou dégradé

        (x, cout ; tr) $\leftarrow$ mission(x, cout ; tr) ; // voir Algorithme 3

      sinon // i.e. mode panne

        a tel que remplacer composant si panne et entretenir si dégradé

        (x, cout ; tr) $\leftarrow$ opérations(x, a, cout ; tr) ; // voir Algorithme 4

    sinon // i.e. en maintenance

      (x, cout ; tr) $\leftarrow$ maintenance(x, a, cout ; tr) ; // voir Algorithme 5

Sorties : cout

Ensuite, on détaille le pseudo-code permettant de simuler une trajectoire du processus contrôlé selon la politique de maintenance préventive π_3 dans l'Algorithme 9.

Algorithme 9 : Dynamique du processus contrôlé par la politique π_3

Entrées : horizon

début

  m, ta, cout, t $\leftarrow$ 0, 0, 0, 0_3 ; // Initialisation

  pour i **de** 1 **à** 3 **faire** // Tirer temps de vie restant en mode stable

    tr[i] $\leftarrow \mathcal{W}(\alpha_{s_i}, \beta_{s_i})$

  pour date de décision **de** 0 **à** horizon-1 **faire**

    si ta $= 0$ **alors**

      si m $\in \mathcal{S}$ **alors** // i.e. mode stable

        (x, cout ; tr) $\leftarrow$ mission(x, cout ; tr) ; // voir Algorithme 3

      sinon // i.e. mode dégradé ou panne

        a tel que remplacer composant si panne et entretenir si dégradé

        (x, cout ; tr) $\leftarrow$ opérations(x, a, cout ; tr) ; // voir Algorithme 4

    sinon // i.e. en maintenance

      (x, a, cout ; tr) $\leftarrow$ maintenance(x, a, cout ; tr) ; // voir Algorithme 5

Sorties : cout

Pour finir, on décrit la construction de trajectoires de la dynamique du système lorsque la politique de référence choisie est π_4, dans l'Algorithme 10.

On rappelle que cette politique de maintenance préventive dépend d'un paramètre $p \in [0,1]$, déterminant par le calcul des quantiles de temps de fonctionnement passés dans le mode dégradé (voir Équation 3.1), les seuils conditionnant les actions de maintenance.

Aussi, en plus des étapes d'initialisation précedemment décrites (lignes 2 et 4), on ajoute le calcul de ces seuils (ligne 5) en fonction du paramètre p donné.

Enfin, le dépassement de ce seuil pour au moins un composant en mode dégradé implique l'envoi du système en maintenance (ligne 12).

Algorithme 10 : Dynamique du processus contrôlé par la politique π_4

Entrées : horizon, paramètre $p \in [0,1]$

1 **début**
2 m, ta, cout, t $\leftarrow$ 0, 0, 0, 0_3 ; `// Initialisation`
3 **pour** i **de** 1 **à** 3 **faire**
4 tr[i] $\leftarrow \mathcal{W}(\alpha_{s_i}, \beta_{s_i})$; `// Tirer temps de vie restant en mode stable`
5 seuils[i] $\leftarrow$ quantile weibull($\mathcal{W}(\alpha_{d_i}, \beta_{d_i})$, p) ; `// Calcul seuils decision`
6 **pour** date de décision **de** 0 **à** horizon-1 **faire**
7 **si** ta $= 0$ **alors**
8 dcm $\leftarrow base_3$(m) ; `// calcul de m en base 3`
9 idxd $\leftarrow \{1 \leq j \leq 3 : \text{dcm[j]}=1\}$; `// indices composants dégradé`
10 **si** m $\in \mathcal{S}$ ou (m $\in \mathcal{D}$ et $\forall$ t[idxd] $<$ seuils[idxd])) **alors**
11 (x, cout ; tr) $\leftarrow$ mission(x, cout ; tr) ; `// voir Algorithme 3`
12 **sinon**
13 a tel que remplacer composant si panne et entretenir si dégradé
14 (x, cout ; tr) $\leftarrow$ opérations(x, a, cout ; tr) ; `// voir Algorithme 4`
15 **sinon** `// i.e. en maintenance`
16 (x, cout ; tr) $\leftarrow$ maintenance(x, a, cout ; tr) ; `// voir Algorithme 5`

Sorties : cout

Notre objectif est ensuite de calculer le coût de ces différentes politiques, par méthode de Monte-Carlo. En plus de calculer le coût moyen de chaque politique de référence, on s'intéresse également à la distribution des panne et à la décomposition du coût entre coûts de maintenance et pénalités liées à l'échec des missions, afin de vérifier la cohérence des résultats de simulation. Enfin, on étudie la dynamique du pod sur un `horizon` de 1 ans et 10 ans, c'est-à-dire de 52 et 520 dates de décisions. Aussi, ces scripts sont implémentés en `Python`, en utilisant notamment le module `Numba` [LPS15]. Ce dernier a permis d'une part de compiler le code "à la volée" pour augmenter la vitesse de calcul des fonctions, et d'autre part de paralléliser les répétitions de Monte-Carlo.

3.3.3 Résultats numériques et comparaison des politiques

Avant de calculer le coût de ces politiques de références, il est possible de les classer intuitivement selon les types de dynamiques qu'elles vont générer et les coûts associés.

La politique de référence π_0 ne permettant pas les opérations de maintenance, elle va laisser le système évoluer vers la panne, puis générer une pénalité d'échec pour chaque date où celui-ci ne pourra pas effectuer sa mission. À terme, le coût d'une telle stratégie devrait être très élevé et supérieur à celui des politiques de références π_1 à π_4. En effet, en autorisant des maintenances, les politiques π_1 à π_4 vont voir les coûts liés aux pénalités d'échec diminuer. Ces quatres politiques diffèrent ensuite entre elles par les modes sur lesquels elle autorisent l'envoi en atelier et les opérations de maintenance.

La politique d'intervention π_2 permet d'entretenir les composants qui se trouvent dans un état dégradé lorsque le système est envoyé en atelier pour une panne, alors que la politique d'intervention π_1 ne prend pas en compte ces dégradations. Par conséquent, la panne future de ces composants risquent de générer une pénalité d'échec et un surcoût de remplacement qu'une action d'entretien aurait pu éviter. Ainsi, les coûts d'interventions et d'échec seront plus élevés en moyenne pour la politique π_1 que pour la politique π_2. Ensuite, la politique de maintenance préventive π_3 permet d'intervenir sur le système dès la première dégradation d'un composant. Si cette politique peut permettre d'éviter une panne et la pénalité associée, elle risque néanmoins de proposer d'intervenir trop tôt sur un système qui aurait pu réussir encore plusieurs missions. Finalement, la meilleure politique est a priori la politique de référence π_4, puisqu'elle permet de pallier à ces défauts et de proposer les opérations de maintenance les plus adaptées à l'état du système, selon le seuil d'intervention choisi.

Enfin, concernant la politique π_4, on note que deux valeurs particulières du paramètre $p \in [0, 1]$ caractérisent des politiques de références déjà énoncées précédemment.

En effet, pour $p = 0$, la valeur du seuil est $s_p = f_q(0) = 0$. Ainsi, pour la politique π_4 évaluée avec le paramètre $p = 0$, le système est envoyé en atelier après l'entrée dans un mode de panne, ou dès qu'un composant se trouve en mode dégradé avec un temps de fonctionnement supérieur à 0. Le système est donc envoyé en atelier dès son passage en mode dégradé ou panne, ce qui correspond à la politique de maintenance préventive π_3.

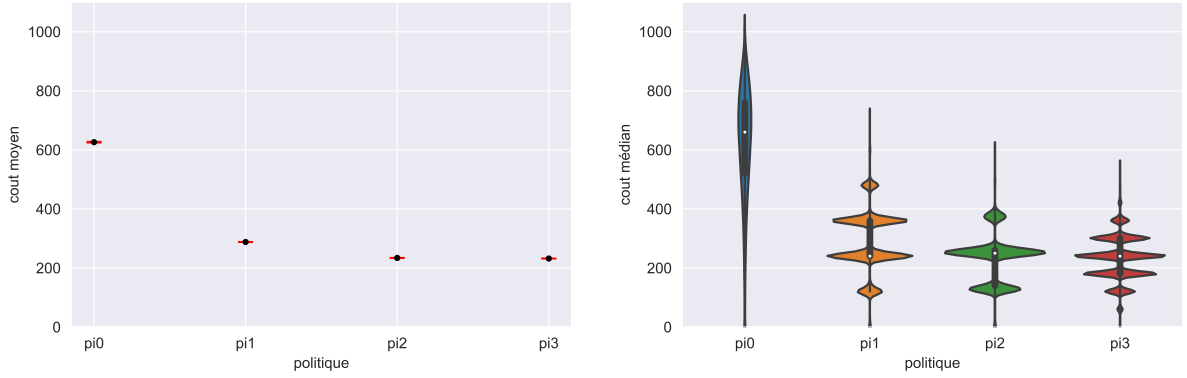
Aussi, pour $p = 1$, on obtient le seuil $s_p = f_q(1) = +\infty$. Ainsi, pour la politique π_4 évaluée avec le paramètre $p = 1$, le système est en envoyé en atelier après l'entrée dans un mode de panne, ou dès qu'un composant se trouve en mode dégradé avec un temps de fonctionnement supérieur à $+\infty$. Le système n'est donc envoyé en atelier qu'une fois en panne, ce qui correspond à la politique de maintenance corrective π_2.

3.3.3.1 Résultats numériques pour un horizon de 52 semaines

Grâce à des simulations exactes, on compare les performances de ces politiques en terme de coût. Leur coût moyen a été évalué via 10^5 simulations de Monte-Carlo.

Coûts des politiques π_0 à π_3. Le coût moyen des différentes politiques est illustré dans la Figure 3.7a alors que leur coût médian et une estimation à noyau de la distribution associée sont présentés dans la Figure 3.7b.

On observe que la politique π_0 est, comme attendu, celle qui réalise le coût le plus élevé. Ensuite, on constate que les politiques π_2 et π_3 réduisent les coûts de gestion du pod d'environ 19% par rapport à la politique π_1 et 62% par rapport à la politique π_0. Néanmoins, bien que le coût moyen des politiques π_2 et π_3 ne semblent pas significativement différents, on peut voir que leurs distributions diffèrent. Pour les distinguer, on va s'interesser d'une part aux statistiques des pannes du système, et d'autre part à la répartition du coût moyen entre coûts de maintenance et pénalités d'échec mission.



(a) Coût moyen des politiques.

(b) Coût médian et distribution associée.

FIGURE 3.7 – Coûts des politiques de référence π_0 à π_3 sur un horizon de 52 semaines.

Détails des côuts. On reprend la décomposition du coût moyen entre coûts de maintenance et pénalités d'échec mission dans la Table 3.5. On peut voir que la politique π_0 ne génère pas de coût de maintenance, étant donné qu'elles ne sont pas permises, et qu'au contraire elle engendre d'importantes pénalités d'échec. À l'inverse, les politiques π_1 à π_3 voient ces pénalités diminuer. Ensuite, la politique π_2 réduit à la fois les coûts de maintenances et d'échec par rapport à la politique π_1, en intervenant de façon opportuniste sur les composants en mode dégradé lors d'une visite en atelier. Enfin, si la politique π_3 annule effectivement les pénalités d'échec en intervenant sur le système avant la panne, elle engendre néanmoins davatange de coûts de maintenance que la politique π_2 en intervenant prémarurément sur le pod.

TABLE 3.5 – Détails des coûts des politiques π_0 à π_3, sur un horizon de 52 semaines.

	π_0	π_1	π_2	π_3
pénalité d'échec	624	49	37	0
coût maintenance	0	239	196	233
coût moyen	**624**	**288**	**233**	**233**
IC 95%	[622 ; 626]	[287 ; 289]	[232 ; 234]	[232 ; 234]

Statistiques de pannes des trajectoires. La répartition des coûts entre opérations et pénalités est à mettre en relation avec le nombre moyen de pannes et le pourcentage de trajectoires sans panne, donnés Table 3.6. En effet, un premier constat est que les politiques π_0 à π_2 ont le même pourcentage de trajectoires sans panne, puisqu'aucune d'elle n'intervient sur le système avant la panne. Ceci s'explique par le fait que ces politiques ne changent pas le pourcentage de trajectoires qui n'atteignent pas de mode de panne. Au contraire, en intervenant dès le passage du pod en mode dégradé, π_3 réalise 99% de trajectoires sans panne. Ensuite, on remarque que la politique π_0 observe en moyenne une seule panne, ce qui est cohérent puisque cette politique ne permet pas de maintenance sur le pod. De plus, π_2 réduit bien le nombre moyen de pannes par rapport à la politique π_1. Enfin, on note que le nombre moyen de pannes sous la politique π_3 est bien proche de 0 en intervenant avant la panne, mais n'est cependant pas nul, puisqu'il arrive que le pod passe en mode dégradé puis panne au cours d'une même mission.

TABLE 3.6 – Nombre moyen de pannes par trajectoire et pourcentage de trajectoires sans panne, pour les politiques π_0 à π_3, sur un horizon de 52 semaines.

	π_0	π_1	π_2	π_3
nombre de pannes	0.99	2.46	1.89	0.01
trajectoires sans panne	0.71	0.74	0.70	99.18

Répartition par mode de panne. Les pourcentages de panne du système, détaillés par mode de panne dans la Table 3.7, s'interprètent également au regard des paramètres de fiabilité donnés Table 3.2. Pour toutes les politiques, la majorité des pannes correspond à une panne du composant 1 (mode 18, de 24 à 50%), puis à une panne du composant 2 (mode 6, de 12 à 30%). Ceci s'explique par le fait que les composants 1 et 2 ont une durée de vie moyenne dans le mode dégradé plus faible que pour le composant 3, dont les pannes sont donc moins fréquentes. Ensuite, pour les politiques π_0 à π_2, deux autres modes sont régulièrement visités (modes 15 et 21, avec 24 à 50%), et correspondent aux modes où l'un des composants 1 ou 2 est en panne et l'autre est dégradé.

Concernant la politique π_3, on note que les modes de panne les plus visités (18, 6 et 2) correspondent aux pannes simples des composants 1, 2 et 3 respectivement, avec une majorité (50%) de panne du composant 1. En effet, s'il est possible qu'un composant stable passe en mode dégradé puis panne au cours d'une seule mission, il est par contre beaucoup moins probable que plusieurs composants passent en mode dégradé puis que l'un d'eux tombe en panne lors d'une même mission. Les autres modes de pannes sont plus rares, avec moins de 6% de trajectoires par mode.

TABLE 3.7 – Répartition, exprimée en pourcentage, des pannes de trajectoires contrôlées par les politiques π_0 à π_3, en fonction du mode de panne, sur un horizon de 52 semaines.

Mode	2	5	6	7	11	14	15	16	18	19	21	22
π_0	5.67	2.63	**11.92**	4.09	4.98	2.60	10.41	4.23	**27.95**	9.24	11.55	4.72
π_1	8.63	3.65	**14.02**	5.42	5.78	2.48	9.48	3.68	**24.69**	8.90	9.60	3.65
π_2	6	2.66	**12.91**	4.35	4.98	2.56	10.51	4.21	**27.12**	8.93	11.35	4.43
π_3	**10.74**	0.84	**32.69**	0.48	0.36	0	1.69	0	**51.15**	1.09	0.97	0

Coûts de la politique π_4. Le coût moyen pour différentes valeurs du paramètre p est illustré dans la Figure 3.8. On observe que cette politique réduit effectivement les coûts de gestion du pod, par rapport aux politiques π_0 à π_3, pour certaines valeurs de p. En particulier, le coût le plus faible est observé pour les valeurs $p = 10\%$ et $p = 20\%$, qui donnent toutes deux un gain relatif d'environ 16% par rapport aux politiques π_2 et π_3. Ces coûts se situent entre le coût de π_2 (233) pour $p = 1$ et celui de π_3 (233) pour $p = 0$.

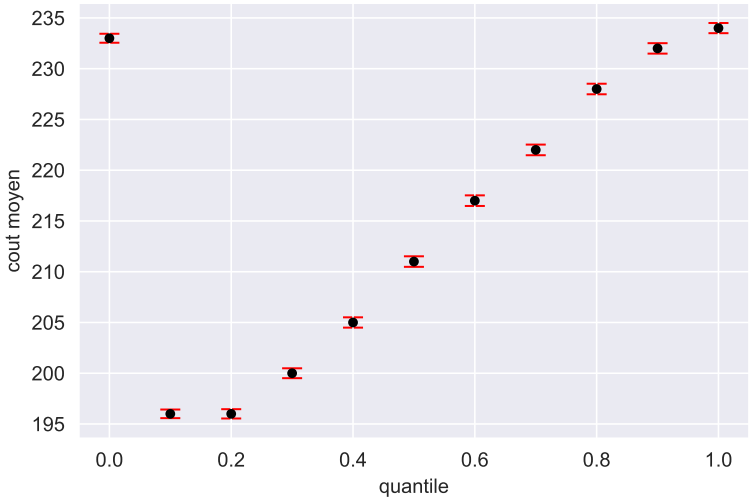


FIGURE 3.8 – Coût de la politique π_4, en fonction de p, sur un horizon de 52 semaines.

Détails des côuts pour π_4 : on reprend la décomposition du coût moyen entre coûts de maintenance et pénalités d'échec mission dans la Table 3.8. On note que la part du coût liée à la pénalité d'échec est croissante avec celle du paramètre p. Ceci s'explique par le fait que la probabilité de tomber en panne augmente avec le temps de fonctionnement passé dans le mode dégradé. À l'inverse, la part du coût liée aux opérations de maintenance n'est pas monotone. Elle réalise un compromis entre intervenir trop tôt sur le système qui aurait pu encore effectuer plusieurs missions avant la panne, et intervenir trop tard, ce qui nécessite des opérations correctives plus coûteuses que les opérations préventives. En particulier, on note que le coût pour les valeurs $p = 10\%$ et $p = 20\%$, pourtant égal, se répartit différemment entre coûts de maintenance et pénalités d'échec.

Remarque 3.3.5. Pour 10^5 répétitions de Monte-Carlo sur ces politiques, les intervalles de confiance ayant une longueur inférieure à 1, ils ne seront plus donnés.

TABLE 3.8 – Détails des coûts de la politique π_4 en fonction de p, sur un horizon de 52 semaines.

p	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
pénalité d'échec	0	7	13	18	22	26	29	32	34	36	37
coût maintenance	233	189	183	182	183	185	188	190	193	195	196
coût moyen	**233**	**196**	**196**	**200**	**205**	**211**	**217**	**222**	**227**	**231**	**233**

Statistiques de pannes des trajectoires pour π_4. La répartition des coûts entre opérations et pénalités est à mettre en relation avec le nombre moyen de pannes et le pourcentage de trajectoires sans panne, donnés dans la Table 3.9.

On y observe notamment que le nombre moyen de panne est croissant avec celui du paramètre p. On note également que le pourcentage de trajectoires sans panne est décroissant avec l'augmentation du paramètre p. Dans les deux cas, ceci s'explique par le fait que la probabilité de tomber en panne augmente en fonction du temps de fonctionnement passé dans le mode dégradé. En particulier, on remarque que le nombre moyen de panne et le pourcentage de trajectoires sans pannes diffèrent pour les paramètres $p = 10\%$ et $p = 20\%$. Ceci s'explique par la répartition différente entre coûts de maintenances et pénalités d'échec observés précedemment dans la Table 3.8.

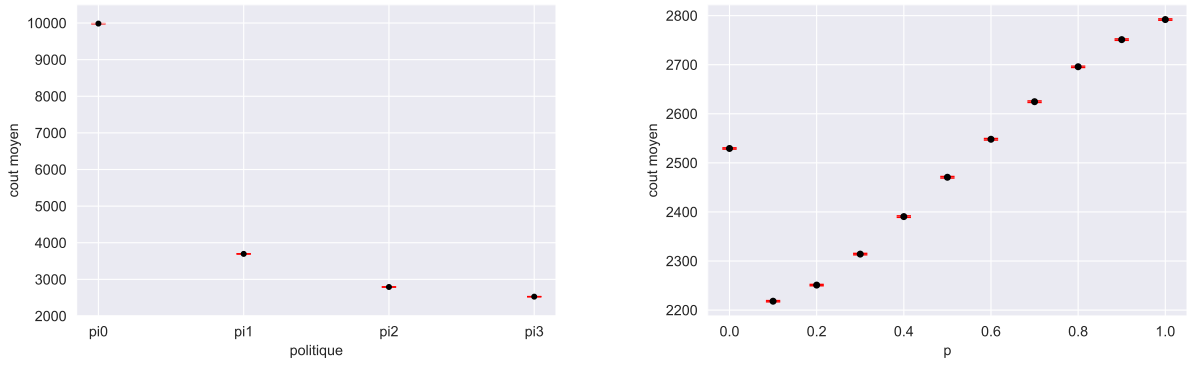
TABLE 3.9 – Nombre moyen de pannes et pourcentage de trajectoires sans panne de π_4, en fonction de la valeur du paramètre p, sur un horizon de 52 semaines.

p	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
nombre de pannes	0.01	0.39	0.67	0.91	1.11	1.30	1.47	1.61	1.73	1.83	1.89
trajectoires sans panne	99.24	66.24	47.96	34.91	25.63	18.12	12.75	8.55	5.29	2.82	0.71

3.3.3.2 Résultats numériques pour un horizon de 520 semaines

Grâce à des simulations exactes, on compare maintenant les performances de ces différentes politiques de références, sur un horizon de 520 semaines, c'est-à-dire 10 ans. Leur coût moyen est évalué au moyen de 10^5 simulations de Monte-Carlo.

Coûts des politiques π_0 à π_4. Le coût moyen des politiques π_0 à π_3 est illustré dans la Figure 3.9a alors que celui de la politique π_4, pour différentes valeurs du paramètre p est présenté dans la Figure 3.9b. On observe la même tendance que pour un horizon de 52 semaines. Premièrement, on constate que le coût de la politique π_0 est croissant avec l'horizon. Aussi, bien que le classement des politiques n'ait pas changé, on note néanmoins que la politique préventive π_3 réalise cette fois un meilleur coût que la politique corrective π_2. Plus précisément, elle permet un gain relatif de 9% par rapport à π_2, et de 74% par rapport à π_0. Concernant la politique π_4, le coût le plus faible est cette fois observé pour la valeur $p = 10\%$, qui donne un gain relatif d'environ 12% par rapport à la politique π_3.



(a) Coût moyen des politiques π_0 à π_3.

(b) Coût de π_4, pour différentes valeurs de p.

FIGURE 3.9 – Coûts des politiques de référence π_0 à π_4, sur un horizon de 520 semaines.

Détails des côuts pour π_0 à π_3. On reprend la décomposition du coût moyen entre coûts de maintenance et pénalités d'échec mission dans la Table 3.10, où les interprétations sont à nouveau les mêmes que pour les résultats à un horizon de 52 semaines.

TABLE 3.10 – Détails des coûts des politiques π_0 à π_3, sur un horizon de 520 semaines.

	π_0	π_1	π_2	π_3
pénalité échec	9984	617	439	1
coût maintenance	0	3079	2352	2527
coût moyen	**9984**	**3696**	**2791**	**2528**
IC 95%	[9982 ; 9986]	[3694 ; 3698]	[2789 ; 2793]	[2526 ; 2530]

Statistiques de pannes des trajectoires pour π_0 à π_3. La répartition des coûts entre opérations et pénalités s'interprète encore avec le nombre moyen de pannes et le pourcentage de trajectoires sans panne, donnés dans la Table 3.11. Comme attendu, la politique π_0 observe toujours en moyenne une seule panne. À l'inverse, on observe pour les politiques π_1 à π_3 qu'avec l'augmentation de l'horizon, le pourcentage de trajectoires sans panne devient nul et le nombre de panne augmente.

TABLE 3.11 – Nombre moyen de pannes par trajectoire et pourcentage de trajectoires sans panne, pour les politiques π_0 à π_3, sur un horizon de 520 semaines.

	π_0	π_1	π_2	π_3
nombre de pannes	1.0	30.85	22.0	0.08
trajectoires sans panne	0.0	0.0	0.0	91.89

Détails des côuts et statistiques de pannes des trajectoires pour π_4. On reprend la décomposition du coût moyen entre coûts de maintenance et pénalités d'échec mission dans la Table 3.12, ainsi que les statistiques de panne associées dans la Table 3.13.

On peut encore noter que la part du coût liée à la pénalité d'échec, ainsi que le nombre moyen de panne augmentent avec le paramètre p ; et qu'à l'inverse le pourcentage de trajectoires sans panne diminue. De nouveau, ceci s'explique par le fait que la probabilité de tomber en panne augmente avec le temps de fonctionnement passé en mode dégradé. Pour finir, la part du coût liée aux opérations de maintenance réalise une nouvelle fois un compromis entre intervenir trop tôt sur le système et tendre vers la panne.

TABLE 3.12 – Détails des coûts pour la politique π_4, en fonction du paramètre p, sur un horizon de 520 semaines.

p	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
pénalité échec	1	85	147	201	249	293	332	367	398	423	439
coût maintenance	2527	2132	2103	2113	2140	2177	2215	2256	2294	2328	2352
coût moyen	**2528**	**2217**	**2250**	**2314**	**2389**	**2470**	**2547**	**2623**	**2692**	**2751**	**2791**

TABLE 3.13 – Nombre moyen de pannes et pourcentage de trajectoires sans panne pour la politique π_4, en fonction du paramètre p, sur un horizon de 520 semaines.

p	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
nombre de pannes	0.08	4.26	7.37	10.06	12.48	14.67	16.63	18.4	19.91	21.18	22
trajectoires sans panne	92	1	0.03	0	0	0	0	0	0	0	0

3.3.3.3 Discussions sur les paramètres et temps de calculs

Au cours des expérimentations ayant conduit à la calibration des paramètres utilisés pour ce chapitre, on a pu observer que toutes les mesures effectuées précedemment (c'est-à-dire les coûts des politiques et leur répartition, et les statistiques de pannes des trajectoires) sont très sensibles à l'évolution des différents paramètres du modèle développé.

Influence des paramètres du modèle. En plus d'avoir testé différentes valeurs pour l'horizon d'étude du MDP, on a également fait varier les valeurs des paramètres de fiabilité ainsi que des différents coûts. On décrit ci-dessous quelques cas particuliers où la sensibilité des paramètres a conduit à des résultats numériques très différents de ceux présentés dans ce chapitre, et pour lesquels ces résultats sont informatifs sur le modèle.

Par exemple, supprimer la pénalité d'échec mission conduit à ce que la politique sans maintenance π_0 soit la moins coûteuse, puisqu'aucun coût (ni de maintenance, ni de pénalité) n'est généré : c'est la politique optimale pour le modèle associé.

Aussi, supprimer le coût d'immobilisation à payer lors d'une visite en atelier fait que la politique corrective *sans action de maintenance préventive* π_1 a un coût plus faible que les politiques π_2 et π_3, car elle évite d'effectuer des entretiens trop précoces trop souvent, sans être pénalisée par ces multiples passages en atelier, alors qu'un coût d'immobilisation incite au contraire à mutualiser les interventions.

Enfin, on a observé que le nombre de pannes décroit (resp. croit) avec l'augmentation (resp. la diminution) de la durée de vie des composants, et à l'inverse le pourcentage de trajectoires sans panne augmente (resp. diminue). En particulier, si la durée de vie moyenne en mode dégradé est plus petit ou de l'ordre de grandeur de la durée d'une mission, alors la politique préventive π_3 présente le coût le plus faible, et a fortiori si le coût de remplacement et/ou d'échec augmente, π_3 semble être la politique optimale pour le modèle associé.

Finalement, les paramètres de fiabilité ont été calibrés afin d'illustrer l'intéret de considérer des politiques qui dépendent du temps de fonctionnement des composants, menant à un problème non trivial d'optimisation de la stratégie de maintenance du pod, et les paramètres de coûts gardés sont ceux qui donnaient les résultats de simulations les plus cohérents avec la problématique industrielle étudiée.

Temps de calculs d'une trajectoire. Tous les résultats présentés dans ce chapitre sont basés sur la simulation de trajectoires et la méthode de Monte-Carlo pour estimer les quantités d'intéret. Aussi, une importante partie de ce travail a été consacrée à les accéler. D'une part, les algorithmes sont compilés via le module `Numba` de `Python`, dont les performances en temps de calculs sont connues pour être parfois comparables au langage compilé `C`. D'autre part, la methode de monte-Carlo se prétant bien à la parallélisation des trajectoires, le module `Numba` permet aussi de les paralléliser pour encore accéler les temps de calculs.

Finalement, les calculs de ce travail ont été effectués avec le support de la Plateforme MESO@LR de l'Université de Montpellier, afin de paralléliser les simulations sur un noeud de 28 coeurs. Néanmoins, la forte variance observée sur les simulations, conduisant à simuler un très grand nombre de trajectoires pour obtenir des intervalles de confiance précis sur les quantités d'intéret, a été et est restée une limitation dans mon travail, et a fortiori pour l'optimisation de la stratégie de maintenance du pod détaillée dans le Chapitre 4 suivant.

3.4 Conclusions

Dans ce chapitre, on a posé un problème d'optimisation de maintenance pour un système constitué de plusieurs composants. La dynamique du pod a été modélisée par un processus markovien décisionnel à espace d'état continu. Des trajectoires contrôlées du processus ont été simulées sous différentes politiques de maintenance corrective ou préventive. Ensuite, on a pu comparer leurs coûts ainsi que les statistiques de pannes des trajectoires, afin de vérifier la cohérence des résultats de simulations obtenus.

Cette modélisation de la dynamique du système a constitué un travail à part entière de la thèse. D'une part, les différents paramètres de coûts et de fiabilités du MDP ont été calibrés afin d'illustrer au mieux la problématique industrielle de Thales. D'autre part, les simulateurs implémentés pour simuler des trajectoires contrôlées par différentes politiques ont été optimisés pour être les moins coûteux possibles en temps de calculs et permettre d'obtenir des estimations précises des quantités d'intéret étudiées.

La prochaine étape de ce travail est de calculer la fonction valeur et une politique optimale associée, sur l'ensemble Π des stratégies admissibles. Il s'agit d'un problème d'optimisation non standard dans le cadre des MDP car le noyau de transition n'est pas explicite analytiquement, il est seulement simulable et de plus, l'espace d'état n'est pas fini. Ainsi, les techniques d'optimisation standard pour les MDP telles que la programmation dynamique ne s'appliquent pas.

L'étape suivante vers la résolution du problème d'optimisation consistera à discrétiser l'espace d'états du processus. On utilisera notamment la méthode de quantification décrite dans la section 2.3, et les coûts des politiques de référence seront utilisés pour évaluer l'impact de la discrétisation sur le calcul des coûts. Il doit y avoir un compromis entre précision, conduisant à un espace d'états avec une cardinalité élevée et complexité numérique conduisant à un espace d'états le plus petit possible. Sur cet espace d'états fini, le noyau de transition ne sera toujours pas explicite, et on utilisera des algorithmes d'optimisation basés sur la simulation de trajectoires pour résoudre ce problème d'optimisation, notamment les méthodes MRAS et ASA décrites dans les sections 2.2.2 et 2.2.3.

Chapitre 4

Approximation numérique de la fonction valeur

L'objectif de ce chapitre est d'approcher la fonction valeur du MDP défini section 3.2 du Chapitre 3, et de fournir une politique quasi-optimale associée via les méthodes MRAS et ASA, dont l'application requiert de prendre des décisions sur un ensemble fini d'états.

Pour ce faire, on propose d'explorer différents sous-ensembles de l'ensemble des politiques admissibles, afin de trouver un compromis entre gain de coût réalisé par l'approximation de la fonction valeur et interprétabilité de la politique associée. La démarche consiste à associer une même action à certains sous-ensembles d'états, puis d'effectuer la transition vers l'état suivant à partir du vrai état courant, ce qui permet de discrétiser la règle de décision sans discrétiser la dynamique du MDP, afin d'éviter le cumul d'erreurs de projections.

Une première proposition est de considérer des politiques qui ne dépendent que de certaines variables d'états. En particulier, on se restreint d'abord aux politiques dont les décisions se basent uniquement sur les modes et le compteur du temps passé en atelier, sans considérer les temps de fonctionnement des composants du pod. Néanmoins, rien n'indique qu'une politique optimale ne tienne pas compte de ces temps, et on a au contraire observé sous-section 3.3.3 que la politique de référence ayant le plus faible coût est celle qui différencie les maintenances selon les durées de vie des composants. Aussi, ces derniers étant à valeurs réelles, l'espace d'états du pod ainsi que l'ensemble des politiques à tester sont infinis.

La seconde proposition est de tenir compte des temps de fonctionnement des composants en mode dégradé via une discrétisation. Pour chaque composant, on utilise la quantification pour construire une partition adaptée à la loi de leur durée de vie en mode dégradé, et on associe une même action à tous les états dans un même mode et dont les temps passés en mode dégradé ont le même plus proche voisin, après projection sur la partition. La transition se fait ensuite sur le vrai état, ce qui permet encore de discrétiser la règle de décision sans discrétiser la dynamique du MDP et d'éviter le cumul d'erreurs de projection.

Sur ces deux sous-espaces de politiques, bien qu'une politique optimale soit a priori non-stationnaire, on s'intéressera également à la recherche d'une politique stationnaire, plus interprétable industriellement. Sur ces différents sous-problèmes, on ne peut toujours pas utiliser la programmation dynamique car le noyau de transition du processus n'est pas explicite, mais on peut appliquer les méthodes MRAS et ASA qui se basent sur les simulations.

L'approximation de la fonction valeur basée sur l'ensemble des politiques ne dépendant que des modes est décrite section 4.1, puis l'optimisation tenant compte de la durée de vie des composants est détaillée section 4.2. Dans les deux cas, les méthodes MRAS et ASA se basent sur l'initialisation d'une loi de probabilité sur les actions possibles par état, ainsi qu'un simulateur de trajectoires du processus contrôlé et plusieurs hyperparamètres à déterminer. Les résultats sont présentés sur un horizon de 52 semaines du MDP, partant de l'état neuf.

4.1 Recherche de politique optimale basée sur le mode

Dans cette section, on cherche à approcher la fonction valeur sur l'ensemble des politiques qui dependent uniquement du mode du pod et du compteur de temps passé en atelier.

Premièrement, l'objectif est de déterminer une politique stationnaire moins coûteuse que les politiques de références, qui associe des actions différentes à des composants dans un même mode, par exemple en fonction de ses paramètres de fiabilité. En effet, on a vu sous-section 3.3.3 que les pannes surviennent en majorité sur les composants 1 et 2. Aussi, différencier les décisions en fonction du composant pourrait être moins coûteux, par exemple en favorisant des opérations préventives des composants 1 et 2 et correctives du composant 3.

Ensuite, on cherche des politiques non stationnaires, dont les décisions se basent sur les mêmes variables d'états. Bien qu'a priori moins facilement interprétable, une politique non stationnaire peut illustrer l'existence de cycles liés à la fréquence de dégradation et défaillance des composants, et permettre d'adapter la fréquence des interventions à ces cycles.

Pour ce faire, on définit le problème d'optimisation en explicitant le sous-ensemble des politiques admissibles considéré, ainsi que le simulateur sur lequel se basent les méthodes MRAS et ASA, section 4.1.1. Ensuite, on précise le plan d'expériance permettant d'ajuster leurs hyperparamètres, section 4.1.2. Enfin, on applique les méthodes au cas stationnaire puis au cas non stationnaire, et on commente les résultats obtenus, section 4.1.3.

4.1.1 Définition du problème d'optimisation

Avant de s'intéresser en détails aux hyperparamètres des méthodes MRAS et ASA, on va distinguer des hyperparamètres qui sont propres à la méthode appliquée, et d'autres qui sont propres au MDP étudié. Pour ces derniers, les deux méthodes s'appuient d'une part sur la définition d'une loi de probabilité sur les actions possibles par état, et d'autre part sur un simulateur du processus contrôlé par les politiques admissibles du problème considéré.

Distribution de probabilités sur les actions possibles par état $P_0 \in \mathcal{M}_{\#\mathbb{X};\#\mathbb{A};H}([0;1])$. Cette distribution de probabilités est déterminée à partir de l'ensemble des actions possibles par état du pod, défini sous-section 3.1.4. Sur ce sous-ensemble de l'espace d'états du système, on a défini un ensemble fini de 27 modes (voir Table 3.1). Aussi, puisqu'une seule action est possible sur le pod lorsqu'il est dans l'atelier suite à une ou des maintenances, c'est-à-dire lorsque $1 < t_a < d_a - 1$, on garde un unique état associé ($n°27$). Finalement, on a $\#\mathbb{X} = 28$ états et $\#\mathbb{A} = 28$ actions (voir Table 3.3) à considérer. L'ensemble des actions possibles par états pour le système est résumé dans la Table 4.1, où les lignes du tableau correspondent aux états x, les colonnes correspondent aux actions a, et où on désigne par 1 (resp. 0) les actions possibles (resp. interdites) dans un état donné. Elle est ensuite initialisée à une loi uniforme (telle que tous les coefficients non nuls associés à un état x sont $\frac{1}{\#\mathbb{A}(x)}$).

TABLE 4.1 – Ensemble des actions possibles par mode du système.

x \ a	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	1	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1	1	0	1	1	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	1	0	1	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
15	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
16	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
18	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
19	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
21	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
22	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Remarque 4.1.1. Pour les états inatteignables, dont les lignes sont grisées Table 4.1, on fixe arbitrairement que la seule action possible est la maintenance qui le ramène à l'état neuf, correspondant aux colonnes grisées. Ceci permet d'utiliser numériquement la numérotation des modes introduite Table 3.1 pour décrire les simulations de trajectoires du MDP, et ces états/actions inatteignables ne seront jamais visités/générées.

Remarque 4.1.2. Si la politique optimale recherchée est stationnaire, alors la probabilité de choisir une action dans un état ne dépend plus de la date de décision, et dans ce cas on peut restreindre la distribution de probabilités initiale $P_0 \in \mathcal{M}_{\#\mathbb{X};\#\mathbb{A}}([0;1])$.

Remarque 4.1.3. Sur ce sous-problème, il existe $\prod_{x\in\mathbb{X}} \#\mathbb{A}(x) \approx 10^9$ politiques stationnaires, et 10^{9H} politiques non stationnaires possibles. Bien que l'espace des politiques non stationnaires soit beaucoup plus grand, chaque itération des algorithmes met à jour la distribution de probabilités P_k sur l'ensemble de ses dimensions, sans exploser en temps de calculs.

Simulateur associé. Pour appliquer les méthodes MRAS et ASA, il faut un simulateur prenant en argument toute politique de l'ensemble des politiques associé au problème d'optimisation considéré. On propose ici un script permettant de générer des trajectoires du MDP contrôlées par toute politique qui ne tient compte que des modes. Ce dernier s'appuie sur les variables et les algorithmes 3, 4 et 5 indroduits sous-section 3.3.2 pour décrire la transition du système selon l'action choisie parmi les trois types d'actions possibles (mission, action de maintenance, rester en atelier).

Algorithme 11 : Simulateur de trajectoires du MDP contrôlé par une politique basée sur les modes.

Entrées : horizon, politique $\pi \in \mathcal{M}_{\#\mathbb{X};H}([\#\mathbb{A}])$.

```
début
    m, ta, cout, t ← 0, 0, 0, 0_3 ; x=(m, t1, t2, t3, ta) ;        // Initialisation
    pour i de 1 à 3 faire        // Tirer temps de vie restant en mode stable
        tr[i] ← W(α_{s_i}, β_{s_i})
    pour date de décision de 0 à horizon-1 faire
        a ← π[m, date de décision] ;        // choix d'action selon la politique
        si a = 0 alors                                          // i.e. mission
            (x, cout ; tr) ← mission (x, cout ; tr) ;          // voir algorithme 3
        sinon si 0 < a < #A − 1 alors          // i.e. action de maintenance
            (x, cout ; tr) ← opérations(x, a, cout ; tr) ;     // voir algorithme 4
        sinon                                  // i.e. rester en maintenance
            (x, cout ; tr) ← maintenance(x, a, cout ; tr) ;    // voir algorithme 5
```

Sorties : cout

Contrairement aux scripts correspondants à la simulations de trajectoires du processus contrôlées par les politiques de références (voir sous-section 3.3.2), où les actions étaient déterminées "en dur" selon l'état observé du MDP, la dynamique est contrôlée par une politique sous forme d'une matrice $\pi \in \mathcal{M}_{\#\mathbb{X};H}([0, \#\mathbb{A} - 1])$, donnée en entrée du simulateur.

Remarque 4.1.4. Dans le cas d'une politique π stationnaire, l'action choisie dépend juste du mode m du processus et non de la date de décision. On peut voir la politique comme un vecteur $\pi \in \mathcal{M}_{\#\mathbb{X}}([0, \#\mathbb{A} - 1])$ et non plus une matrice, et pour choisir une action il suffit de remplacer dans l'algorithme 11 : $a \leftarrow \pi$[m, date de décision] par $a \leftarrow \pi[m]$ (ligne 6).

Remarque 4.1.5. De la même façon, si on recherche une politique optimale stationnaire (respectivement non stationnaire) avec les méthodes MRAS et ASA, alors la distribution de probabilités initiale sur les actions possibles par état s'écrit $P_0 \in \mathcal{M}_{\#\mathbb{X};\#\mathbb{A}}([0; 1])$ (respectivement $P_0 \in \mathcal{M}_{\#\mathbb{X};\#\mathbb{A};H}([0; 1])$).

Le cadre général de ce problème d'optimisation étant défini, l'optimisation de la stratégie de maintenance du pod via l'application des méthodes MRAS et ASA requiert ensuite l'ajustement de leurs hyperparamètres.

4.1.2 Plan d'expériences

On va distinguer des hyperparamètres qui peuvent être ajustés à partir de la connaissance du MDP étudié, et d'autres qui sont propres à la méthode appliquée et influencent leur coût computationnel ainsi que leur convergence. Dans les deux cas, certains sont communs aux deux méthodes. On explicite ces derniers section 4.1.2.1, puis on décrit les hyperparamètres spécifiques aux méthodes MRAS et ASA dans les sections 4.1.2.2 et 4.1.2.3.

4.1.2.1 Hyperaramètres communs aux deux algorithmes

Dans toutes les expériences présentées dans cette section, on fixe les paramètres suivants pour les méthodes MRAS et ASA.

Nombre initial de simulations pour la méthode de Monte-Carlo $M_0 \geq 1$. Ce nombre est fixé à 10 000, en se basant sur les résultats numériques obtenus sur les politiques de références décrits sous-section 3.3.3. En effet, on a observé pour 10^4 (resp. 10^5) répétitions de Monte-Carlo une longueur d'intervalle de confiance inférieure à 3 (resp. 1), soit approximativement 1.5% (resp. 0.5%) d'écart avec les meilleurs coûts observés sur les politiques de références. On choisit d'initialiser $M_0 = 10^4$ répétitions, ce qui permet d'obtenir des résultats précis sans trop impacter le temps de calcul des algorithmes. Par ailleurs, on rappelle que le nombre de répétitions de Monte-Carlo peut augmenter à chaque itération des deux algorithmes, selon un autre hyperparamètre propre à chacun.

Nombre initial de politiques candidates $N_0 \geq 2$. Ce nombre est fixé arbitrairement à 100. Les méthodes étant basées sur l'exploration de l'espace des politiques admissibles, il doit être suffisamment grand pour le visiter efficacement, sans faire exploser le temps de calculs. Ici, évaluer $N_0 = 100$ politiques via $M_0 = 10^4$ trajectoires nécéssite 10^6 simulations à la première itération, puis ce nombre augmente. Enfin, on rappelle que le nombre de politiques peut augmenter pour les deux méthodes, selon un autre hyperparamètre propre à chacune.

Nombre limite d'itérations $K \in \mathbb{N}$. On le fixe arbitrairement à 100. On verra sur les expériences que la dernière itération utile intervient souvent bien avant la 100-ième itération.

4.1.2.2 Hyperparamètres de la méthode MRAS

L'algorithme MRAS se base ensuite sur 6 autres hyperparamètres, rappelés ci-dessous.

— $\alpha > 1$: facteur par lequel on multiplie le nombre de politiques candidates à simuler à la prochaine itération de l'algorithme, lorsqu'une itération ne trouve pas de seuil meilleur qu'à l'itération précédente ;

— $\beta > 1$: facteur par lequel on multiplie le nombre de simulations pour la méthode de Monte-Carlo, à chaque itération de l'algorithme ;

— $\epsilon > 0$: tolérance pour la prise en compte de moins bonnes politiques dans la mise à jour de la matrice P_k ;

— $\nu \in]0; 1[$: coefficient qui détermine la pondération associée à l'estimation courante de la distribution de probabilité, dans la mise à jour de la matrice P_k ;

— $\lambda \in]0; 1[$: coefficient de mélange qui détermine la proportion de politiques candidates qui seront simulées selon la matrice initiale P_0 ;

— $\rho_0 \in]0; 1]$: Proportion initiale qui sert à calculer le seuil de performance, et détermine la proportion de politiques candidates gardées pour la mise à jour de la matrice P_k.

On choisit de fixer les valeurs de deux paramètres suivants, que les résultats numériques du Chapitre 3 permettent de calibrer.

Coefficients α et β. Les paramètres α et β contrôlent la croissance respectivement du nombre de politiques candidates et du nombre de répétitions pour le calcul du coût des politiques par méthode de Monte-Carlo. On fixe $\beta = 1.025$, ce qui implique que le coût des politiques candidates est évalué au moyen de 10^5 répétitions de Monte-Carlo aux dernières itérations de l'algorithme. On fixe $\alpha = 1.1$, ce qui donne au plus 10^6 politiques à tester dans le pire des cas où aucune des itérations n'a permis de déterminer un seuil de coût meilleur qu'à l'étape précédente, et un peu plus de 10^4 si la moitié des itérations permettent d'améliorer la performance.

Il n'existe pas de résultat théorique sur la façon d'ajuster la valeur de ces hyperparamètres de la méthode MRAS. Aussi, compte tenu des valeurs de paramètres fixés précédement, son exécution nécéssite de 2 à plus de 24 heures de calculs, avec une forte variance. Ainsi, il n'est pas envisageable de proposer pour chaque paramètre une grille fine de valeurs à tester, et de chercher la meilleure combinaison parmis la grille croisée des possibilités.

Une alternative réalisable en temps de calcul consiste à tester l'influence des paramètres un par un, sur un petit nombre de valeurs à tester. Pour ce faire, on choisit 4 valeurs à tester pour chacun de ces hyperparamètres. Le nombre de combinaisons à tester pour une grille croisée nécessiterait 256 exécutions de la méthode MRAS, ce qui n'est pas réalisable si chaque exécution de l'algorithme prend plusieurs heures de calculs, même malgré la forte parallélisation des simulations de Monte-Carlo. Finalement, on lance un plan d'expérience pour ajuster les paramètres sur les grilles de valeurs suivantes, choisies uniformément sur leurs ensembles de définition, qui demandent donc 16 exécutions de l'algorithme.

— $\epsilon \in \{0.1,\ 0.5,\ 1,\ 10\} \subset \mathbb{R}_+^*$;

— $\nu \in \{0.2,\ 0.4,\ 0.6,\ 0.8\} \subset]0;1[$;

— $\lambda \in \{0.2,\ 0.4,\ 0.6,\ 0.8\} \subset]0;1[$;

— $\rho_0 \in \{0.2,\ 0.4,\ 0.6,\ 0.8\} \subset]0;1]$.

Les résultats numériques associés sont ensuite détaillés dans la sous-section 4.1.3, dans le cas de la recherche d'une politique optimale stationnaire puis dans le cas non stationnaire.

4.1.2.3 Hyperparamètres de la méthode ASA

On rappelle que la distribution initiale P_0 est commune aux méthodes MRAS et ASA. L'algorithme ASA se base ensuite sur 3 hyperparamètres, rappelés ci-dessous.

— $\alpha_0 \in]0;1[$: coefficient initial déterminant la pondération associée à l'estimation courante de la distribution de probabilité, dans la mise à jour de la matrice P_k ;

— $\beta_0 \in]0;1]$: coefficient initial déterminant la proportion de politiques simulées selon la matrice initiale P_0 ;

— $T_0 \in \mathbb{R}_+^*$: température initiale, qui dirige la convergence de la méthode ;

D'après [CFHM13], on peut fixer les hyperparamètres $\alpha_0 = \dfrac{1}{100^{0,501}} \approx 0.1$ et $\beta_0 = 1$. On rappelle que la température initiale T_0 doit être ni trop élevée, pour ne pas ralentir la convergence de l'algorithme, ni trop proche de zéro, pour éviter d'être coincé dans un minimum local. On propose d'évaluer son infuence sur l'algorithme ASA avec la grille de valeurs arbitraires suivantes.

— $T_0 \in \{0.1,\ 1,\ 10,\ 100\} \subset \mathbb{R}_+^*$.

Les résultats numériques associés sont ensuite détaillés dans la sous-section 4.1.3, dans le cas de la recherche d'une politique optimale stationnaire puis dans le cas non stationnaire.

4.1.3 Résultats numériques

On cherche ici à approcher la fonction valeur sur l'ensemble des politiques dont les décisions sont basées uniquement sur les modes des composants du pod et le compteur de temps passé dans l'atelier, sans considérer les temps de fonctionnement des composants du pod. Pour ce faire, on évalue l'infuence des hyperparamètres des algorithmes MRAS et ASA, afin d'ajuster leurs valeurs parmi les grilles fixées dans la sous-section 4.1.2. En particulier, on applique ici une procédure de type ascendante, c'est-à-dire que tous les hyperparamètres sont initialisés arbitrairement à le plus petite valeur de la grille proposée, puis la valeur ayant conduit au *meilleur* coût lors d'une étape est gardée pour les tests suivants. Enfin, on étudie le cas stationnaire puis le cas non stationnaire, et on compare dans les deux cas les résultats numériques obtenus via les deux méthodes.

4.1.3.1 Résultats de la méthode MRAS pour le cas stationnaire

L'objectif de cette section est de déterminer une politique de contrôle stationnaire du pod, qui mène au plus faible coût possible, et dont les décisions sont basées uniquement sur les modes du pod et le compteur de temps passé dans l'atelier. Pour ce faire, on présente maintenant les résultats numériques de la méthode MRAS correspondants au plan d'expérience présenté dans la sous-section 4.1.2, en appliquant une procédure de type ascendante où tous les hyperparamètres sont initialisés arbitrairement à le plus petite valeur de la grille, puis la valeur ayant conduit au *meilleur* coût lors d'une étape est gardée pour les tests suivants.

Fonctionnement et critère d'arrêt de l'algorithme MRAS. On rappelle que la méthode MRAS se base sur l'évolution d'un seuil de performance, noté γ_k, pour mettre à jour la distribution sur les actions possibles par état $\hat{P}$. Ce seuil est un coût calculé à chaque itération de l'algorithme, et correspond au quantile tel qu'une proportion ρ_k de politiques ont un coût inférieur ou égal à γ_k et sont sélectionnées pour la mise-à-jour de $\hat{P}_k$. Si cette nouvelle distribution peut améliorer le seuil γ_{k+1} en générant des politiques dont les ρ_k meilleures ont un coût inférieur au seuil γ_k calculé à l'itération précédente, il faut parfois se restreindre à une proportion plus faible ρ_{k+1} de politiques pour abaisser ce quantile, ou encore générer un plus grand nombre de politiques si aucune n'a présenté un coût inférieur à γ_k. Aussi, on observe dans nos simulations que la dernière itération où le seuil de performance est mis-à-jour, appelée ensuite *dernière itération utile*, intervient toujours avant la 100-ième voire la 50-ième itération de l'algorithme. Ainsi, toutes ces itérations *inutiles* mènent à augmenter le nombre de politiques candidates, ce qui tend très rapidement à faire exploser le nombre de politiques à générer et à évaluer, sans forcément améliorer les résultats. Sachant que le nombre de trajectoires M_k utilisées pour calculer le coût d'une politique par méthode de Monte Carlo est strictement croissant au cours des itérations, les temps de calculs explosent.

Finalement, on choisit par la suite d'interrompre l'algorithme MRAS dès que 10 itérations **consécutives** ne permettent plus de mettre à jour le seuil de performance γ_k. Afin de comparer toutefois des résultats ayant la même précision, le coût de la meilleure politique déterminée à la dernière itération de la méthode est ensuite recalculé via 10^5 simulations.

Remarque 4.1.6. Un autre critère d'arrêt possible pour la méthode MRAS serait d'interrompre l'algorithme lorsqu'un certain nombre d'itérations consécutives n'ont pas permis de mettre à jour la distribution de probabilités $\hat{P}$. Néanmoins, comme on le verra tout au long de ce chapitre, de nombreux états sont finalement rarement atteints selon la politique appliquée. Il s'agit notamment des états de panne du pod, qui sont peu observés lorsque la politique choisie est préventive. Aussi, la mise à jour de $\hat{P}$ pour ces états non observés ne conduit pas à trouver une politique dont le coût est plus faible, c'est pourquoi on retient un critère d'arrêt basé sur l'évolution du seuil de performance plutôt que sur la mise à jour la distribution de probabilités $\hat{P}$.

Paramètre de tolérance ϵ. Les résultats numériques de la méthode MRAS en fonction du paramètre ϵ sont donnés Table 4.2, où on décrit le plus faible coût trouvé ainsi que la dernière itération utile de l'algorithme. Le premier constat est que dans tous les cas, l'exécution de l'algorithme MRAS a permis de construire des politiques stationnaires dont le coût est plus faible que celui des politiques de référence basées sur les modes. Le plus faible coût génère un gain relatif de 10% (resp. 27%) par rapport aux politiques π_2 et π_3 (resp. π_1). Par contre, ce coût est supérieur à celui de la politique π_4 tenant compte des temps de fonctionnement des composants. Aussi, le paramètre ϵ semble avoir peu d'influence, car tous les coûts (évalués via 10^5 simulations de Monte-Carlo) sont proches, et on observe que l'algorithme cesse de chercher après un faible nombre d'itérations. Par contre, pour la plus grande valeur testée (ϵ=10), on obtient un coût légèrement supérieur et une dernière itération utile plus précoce qu'avec les autres valeurs testées. C'est un résultat attendu, car si ϵ est trop grand, alors l'algorithme ne peut pas faire évoluer le seuil de performance. En effet, ϵ intervient lors de deux étapes de chaque itération de l'algorithme MRAS. D'une part, lors de la mise à jour du seuil de performance γ_k, il quantifie l'amélioration de performance requise par rapport à celle de l'itération précédente, c'est à dire que γ_k doit être inférieur ou égal à $\gamma_{k-1} - \epsilon$. Ainsi, l'algorithme fait en sorte que la mise à jour courante de la distribution P_k soit principalement faite avec des politiques ayant un coût plus faible que celles utilisées pour la mise-à-jour faite aux itérations précédentes. Il garantit la monotonie du seuil de performance. D'autre part, lors de la mise à jour de la distribution de probabilités P_k, l'algorithme fait intervenir les politiques dont le coût est inférieur au seuil de performance γ_k, mais tient également compte des politiques dont le coût se situe entre la précédente performance γ_{k-1} et la performance courante γ_k, via une certaine pondération (voir Figure 2.2). Finalement, on choisit de garder la valeur ϵ=1 pour les tests suivants, qui a conduit au coût le plus faible.

TABLE 4.2 – Résultats numériques en fonction de ϵ pour l'algorithme MRAS, avec $\nu = 0.2$, $\lambda = 0.2$, et $\rho_0 = 0.2$, dans le cas stationnaire basé sur les modes.

ϵ	Meilleur coût	Dernière itération utile
0.1	210.77	29
0.5	210.75	15
1	**210.70**	20
10	211.42	4

Coefficient ν. Les résultats numériques en fonction de ν sont résumés dans la Table 4.3. Ici encore, l'exécution de l'algorithme MRAS permet un gain relatif de près de 10% sur celui des politiques de référence. On observe aussi que la méthode cesse rapidement de chercher, mais on a une tendance décroissante avec ν sur la dernière itération utile. Ceci s'explique par le fait qu'il faut plus d'itérations pour prendre en compte les meilleures politiques trouvées si on leur accorde un poids plus faible dans la mise à jour de P_k. Enfin, on remarque que les seuils obtenus sont en fait croissants avec ν, ce qui s'interprète par le fait que l'algorithme a donné un poids très élevé aux politiques meilleures que le seuil aux premières itérations, au détriment d'autres politiques qui auraient pu donner de meilleures performances mais dont les probabilités associées ne permettent plus de les tirer au sort. Ces résultats laissent à penser qu'il est préférable de garder la plus faible valeur de ν, qui permet de s'orienter plus lentement mais plus prudemment vers de meilleures performances, et on garde $\nu = 0.2$.

TABLE 4.3 – Résultats numériques en fonction de ν pour l'algorithme MRAS, avec $\epsilon = 1$, $\lambda = 0.2$, et $\rho_0 = 0.2$, dans le cas stationnaire basé sur les modes.

ν	Meilleur coût	Dernière itération utile
0.2	**210.70**	20
0.4	210.81	14
0.6	211.02	10
0.8	213.70	7

Coefficient de mélange λ. Les résultats numériques en fonction de λ sont repris Table 4.4. Encore une fois, les coûts trouvés sont dans un même intervalle de confiance. On note que le coût le plus faible est obtenu pour la plus petite valeur de λ. Ce coefficient déterminant la proportion de politiques candidates simulées selon la matrice initiale P_0, il est cohérent de choisir une faible valeur suite au choix d'une paramètre ν petit également : si l'algorithme s'éloigne lentement de la distribution initiale au profit de meilleures solutions, il a moins besoin d'y revenir au cours des itérations. On garde la valeur $\lambda = 0.2$.

TABLE 4.4 – Résultats numériques en fonction de λ pour l'algorithme MRAS, avec $\epsilon = 1$, $\nu = 0.2$, et $\rho_0 = 0.2$, dans le cas stationnaire basé sur les modes.

λ	Meilleur coût	Dernière itération utile
0.2	**210.71**	20
0.4	210.88	16
0.6	210.83	19
0.8	211.40	9

Proportion initiale ρ_0. Les résultats numériques en fonction de ρ_0 sont donnés Table 4.5. Les coûts trouvés sont encore du même ordre de grandeur. Rappelons que cette proportion sert à calculer le seuil de performance initial de l'algorithme, qui est un coût tel que les $\rho_0\%$ politiques candidates ayant un coût inférieur sont gardées pour la mise à jour de la distribution de probabilités P_k. Aussi, il est cohérent de choisir une valeur élevée pour ρ_0, bien que prendre en compte trop de politiques puisse conduire à converger moins rapidement vers les meilleures, ceci permet d'apprendre à partir d'un plus grand nombre de politiques pour mettre à jour P_k. Cette proportion décroit ensuite pour estimer P_k via des politiques ayant le plus faible coût possible, de sorte que seules quelques politiques sont encore considérées et que le seuil de performance est finalement le coût de la meilleure politique trouvée. Pour finir, on garde la valeur $\rho_0 = 0.8$, qui a mené au plus faible coût sur les tests effectués.

TABLE 4.5 – Résultats numériques en fonction de ρ_0 pour l'algorithme MRAS, avec $\epsilon = 1$, $\nu = 0.2$, et $\lambda = 0.2$, dans le cas stationnaire basé sur les modes.

ρ	Meilleur coût	Dernière itération utile
0.2	210.71	20
0.4	211.01	19
0.6	210.90	21
0.8	**210.62**	36

Conclusions. Finalement, l'exécution de l'algorithme MRAS a permis de construire dans tous les cas des politiques stationnaires dont le coût est plus faible que celui des politiques de référence, avec un gain relatif de près de 10%, bien que ses différents hyperparamètres ne semblent pas avoir de grande influence sur les résultats numériques obtenus dans ce cas. Aussi, le fait que toutes les exécutions de l'algorithme mènent à des coûts se trouvant tous dans un même intervalle de confiance peut laisser penser qu'il a convergé vers un minimum pour ce problème. Par contre, on ne peut apporter aucune garantie sur cette hypothèse, car la méthode MRAS est asymptotique, et les algorithmes ont arrêté de chercher après un faible nombre d'itérations, bien qu'on puisse supposer qu'il s'arrête sur la solution du problème.

Interprétations des résultats. On étudie maintenant plus en détails les différentes sorties numériques de la méthode MRAS. En particulier, on souhaite expliciter la meilleure politique obtenue avec la distribution de probabilités terminale $\hat{P}_{st}$, notée par la suite π^*_{st}. Notre objectif est de l'interpréter industriellement. Pour ce faire, on la compare aux politiques de références afin d'identifier les modes pour lesquels choisir une action différente a permis de réduire le coût de la stratégie de maintenance. Aussi, on détaille les statistiques telles que la répartition des coûts entre pénalités et opérations de maintenance, ainsi que le nombre et la répartition des pannes observées. Du point de vue de l'application de la méthode MRAS, on s'intéresse également à l'évolution du seuil de performance au cours des itérations de l'algorithme, qui donne une indication sur les coûts des politiques visitées au cours des itérations. Pour terminer, on étudie aussi la convergence de la distribution terminale $\hat{P}_{st}$, qui peut indiquer si les explorations numériques doivent être poursuivies.

Politique stationnaire optimale basée sur les modes. Globalement, c'est de la politique de référence π_2, qui est une politique de maintenance corrective avec actions opportunistes de maintenances préventives, que la politique optimale basée sur les modes trouvée dans le cas stationnaire se rapproche le plus. Dans le détail, elle s'en distingue par exactement 6 actions qui sont différentes. En particulier, c'est dans deux sous-groupes de modes qu'elle préconise de choisir des actions différentes.

Premièrement, dans les modes sans panne avec des dégradations multiples (modes 4, 10, 12, 13), elle indique d'aller dans l'atelier pour entretenir tous les composants qui sont dégradés, là où la politique π_2 indiquait d'attendre la panne d'au moins l'un d'entre eux pour s'y rendre. Ce résultat est parfaitement interprétable industriellement : lorsque plusieurs composants se trouvent dans un mode dégradé, il est préférable de mutualiser les opérations préventives en atelier, plutôt que de risquer que l'un d'entre eux tombe prochainement en panne, et entraîne des pénalités ainsi que des opérations de maintenances plus coûteuses.

Ensuite, dans le mode 7 (resp. 11), qui correspond à une panne du composant 2 et une dégradation du 3 (resp. une panne du composant 3 et une dégradation du 1), la politique optimale indique de n'intervenir que sur le composant en panne, là où la politique π_2 indiquait d'intervenir sur les deux composants lors d'une visite en atelier. Ceci peut s'expliquer par le fait que les paramètres de fiabilité des composants leurs permettent d'effectuer plusieurs mission dans le mode dégradé avant de tomber en panne, et conduit donc à observer des dégradations multiples pour lesquels la politique va ensuite envoyer le système en atelier. Sinon, cela peut provenir du fait que ces modes sont en fait très peu visités lors des simulations, et a fortiori si la politique optimale indique d'intervenir préventivement dès que plusieurs composants du système se trouvent dans un mode dégradé : ils sont visités dans moins de 0.5% des cas (voir la Table 4.8 pour la répartition des modes de panne observés sous cette politique). La politique est détaillées en Annexe A.1, Table A.1.

Détails des côuts de la politique π^*_{st}. On détaille la décomposition du coût moyen dans la Table 4.6. La politique π^*_{st} réduit à la fois les coûts de maintenances et les coûts d'échec par rapport aux politiques de références π_1 à π_3, en mutualisant les actions d'entretien sur les composants en mode dégradé en cas de dégradations multiples. Ceci indique que le fait d'attendre l'apparition de dégradations multiples ne conduit pas forcément à subir davantage de pannes, et donc pas à payer plus de pénalités et d'opérations plus coûteuses.

TABLE 4.6 – Détails des coûts de la politique stationnaire π^*_{st}, sur un horizon de 52 semaines.

	π^*_{st}	π_1	π_2	π_3
pénalité échec	20	49	37	0
coût maintenance	190	239	196	233
coût moyen	**210**	288	233	233
IC 95%	[209 ; 211]	[287 ; 289]	[232 ; 234]	[232 ; 234]

Statistiques de pannes. Le nombre moyen de pannes et le pourcentage de trajectoires sans panne sont donnés Table 4.7. On note que le nombre de pannes est réduit par rapport aux politiques correctives π_1 et π_2. Aussi, bien qu'elle diminue le pourcentage de trajectoires sans panne par rapport à la politique préventive π_3, elle mène à un coût plus faible.

TABLE 4.7 – Nombre moyen de pannes par trajectoire et pourcentage de trajectoires sans panne, pour la politique π^*_{st}, sur un horizon de 52 semaines.

	π^*_{st}	π_1	π_2	π_3
nombre de pannes	**1.03**	2.46	1.89	0.01
% trajectoires sans panne	**26.59**	0.74	0.70	99.18

Répartition par mode de panne pour la politique π^*_{st}. La répartition par mode de panne du pod est détaillée dans la Table 4.8. On note que les modes de panne les plus visités (18, 6 et 2) correspondent aux pannes simples des composants 1, 2 et 3 respectivement, avec une majorité (56%) de pannes du composant 1. Cette répartition montre que les autres modes de pannes sont plus rarement observés. En particulier, les modes 5, 7, 11, 14, 16 et 22 sont très peu visités, avec moins de 0.5% de visites sur les trajectoires simulées.

TABLE 4.8 – Répartition des pannes de trajectoires contrôlées par la politique π^*_{st}, en fonction du mode de panne, sur un horizon de 52 semaines.

Mode	2	5	6	7	11	14	15	16	18	19	21	22
π^*_{st}	**12.77**	0.29	**26.90**	0.41	0.43	0.01	0.93	0.02	**56.12**	0.85	1.27	0.02

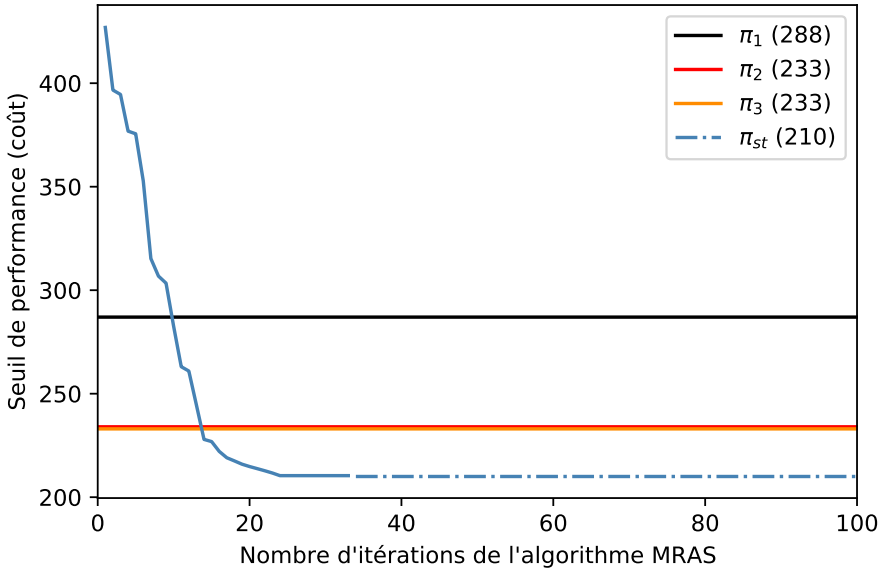


FIGURE 4.1 – Évolution du seuil de performance γ_k en fonction du nombre d'itérations de la méthode MRAS, dans le cas stationnaire basé sur les modes.

Distribution de probabilités $\hat{P}_{st}$. Globalement, on note que ces distributions semblent avoir convergé vers une seule action possible par mode, excepté pour 6 d'entre eux. Dans le détail, il s'agit des modes 5, 7, 11, 14, 16 et 22, qui correspondent tous à des modes où un composant du pod est en panne et un ou plusieurs composants sont dans un mode dégradé. Ceci s'explique par le fait que ces états ont une probabilité très faible d'être visités si la politique optimale indique d'intervenir préventivement dans les modes de dégradations multiples. Ainsi, les états étant trop rarement visités sont très peu mis à jour dans la distribution de probabilités $\hat{P}_{st}$ au cours des itérations de l'algorithme, mais ceci indique également qu'ils ont finalement une influence très faible sur le coût généré par ce type de politique.

Évolution du seuil de performance. On présente maintenant l'évolution du seuil de performance γ_k en fonction du nombre d'itérations de l'algorithme MRAS, Figure 4.1. Il est particulièrement intéressant de noter qu'à la première itération, $1 - \rho_0 = 20\%$ des politiques ont un coût supérieur à 427, c'est à dire que l'algorithme MRAS a considéré des politiques nettement plus coûteuses que les politiques de références, faisant jusqu'à 83% de perte relative. L'algorithme s'oriente ensuite rapidement vers les politiques qui réalisent près de 10% de gain.

Finalement, cette politique réduit les coûts de gestion du pod mais fournit également une stratégie très facilement interprétable, cohérente et applicable d'un point de vue industriel. L'ensemble des résultats présentés laisse penser que l'algorithme a pu converger vers une solution du problème, après avoir suffisamment exploré l'espace des politiques stationnaires basées sur les modes du pod.

4.1.3.2 Résultats de la méthode ASA pour le cas stationnaire

On compare ici les résultats obtenus via la méthode MRAS avec ceux de la méthode ASA, sur 52 semaines. Pour ce faire, on utilise la même distribution initiale P_0 et on évalue l'infuence du paramètre T_0 sur les résultats de l'algorithme ASA, via le plan d'expérience décrit section 4.1.2.3. On rappelle que la méthode ASA tient compte de toutes les politiques simulées à chaque itération pour mettre à jour la distribution $\hat{P}$, qui est donc mise à jour à chaque itération, et on donne pour résultat de la méthode ASA le coût de la meilleure politique trouvée : on dit que c'est la politique optimale renvoyée par l'algorithme.

Température initiale T_0. Les résultats numériques de la méthode ASA en fonction de T_0 sont donnés Table 4.9. On obtient des coûts similaires à ceux obtenus via l'algorithme MRAS, ce qui permet de trouver le même minimum, mais pas de meilleur. La valeur de T_0 le réalisant illustre bien un compromis entre température trop élevée et trop faible.

TABLE 4.9 – Résultats en fonction de T_0 pour ASA, dans le cas stationnaire sur les modes.

T_0	0.1	1	10	100
Meilleur coût	216.32	212.70	**210.62**	215.44

Politique stationnaire optimale basée sur les modes. Cette politique est très proche de celle déterminée via l'algorithme MRAS, et s'en différencie par exactement 6 actions. Il s'agit des modes 5, 7, 11, 14, 16 et 22, pour lesquels la distribution $\hat{P}_st$ déterminée via la l'algorithme MRAS n'a pas convergé vers une seule action. Ceci s'explique encore par la très faible probabilité de visiter ces états, et cela confirme que l'action choisie dans ces états a une influence négligeable sur le coût généré par ces politiques, puisque les deux politiques explicitées ont le même coût. La politique est détaillées en Annexe A.1, Table A.1.

Statistiques de la politique. Les statistiques présentées pour la politique trouvée via la méthode MRAS sont similaires pour celle déterminée via la méthode ASA, aussi elles ne sont pas détaillées.

Distribution de probabilités $\hat{P}_{st}$. Globalement, on note que ces distributions semblent avoir convergé vers une seule action possible par mode, excepté pour 6 d'entre eux : les modes 5, 7, 11, 14, 16 et 22, déjà évoqués comme peu visités et n'ayant aucun impact sur le côut.

Finalement, l'ensemble des résultats de cette section confortent l'idée que la fonction valeur, dans le cas **stationnaire** basé sur les modes, a pu être atteinte, avec les algorithmes MRAS et ASA. Aussi, les politiques associées (puisqu'il n'y a pas unicité) permettent d'expliciter une stratégie de maintenance qui est interprétable et applicable industriellement.

4.1.3.3 Résultats de l'algorithme MRAS pour le cas stationnaire, sur un horizon de 520 semaines, via la distribution déterminée pour 52 semaines

Dans cette section, on propose d'utiliser la distribution $\hat{P}_{st}$ déterminée via la méthode MRAS pour un horizon de 52 semaines, afin d'optimiser la stratégie de maintenance du système sur un horizon de 520 dates de décisions, soit 10 ans. En effet, on rappelle que la distribution P_0 peut être initialisée à une loi uniforme sur les actions possibles par état, où bien à partir d'un a priori différent. L'objectif de cette démarche, qui s'apparente à la méthode de *splitting-initializing* parfois utilisée en quantification pour construire une partition de Voronoï à $K+1$ points à partir d'une partition optimale à K points, est d'illustrer un passage à l'échelle possible pour résoudre un MDP sur un horizon plus long.

Pour ce faire, on exécute l'algorithme MRAS avec les hyperparamètres $\epsilon = 1$, $\nu = 0.2$, $\lambda = 0.2$ et $\rho_0 = 0.8$, qui ont conduit au coût le plus faible lors des exécutions précédentes. Ensuite, on compare les résultats obtenus via une distribution initiale P_0 uniforme et ceux réalisés via une distribution P_0 initialisée à partir de l'estimation $\hat{P}_{st}$. Enfin, on compare les performances obtenues avec celles des politiques de références détaillées dans la section 3.3.3.2 du Chapitre 3.

Fonction valeur. Les meilleurs coûts trouvés via l'algorithme MRAS, selon la distribution initale utilisée, sont donnés dans la Table 4.10. On note que les deux approches ont conduit à un coût similaire : la différence relative entre les deux coûts est de l'ordre de 0.33%.

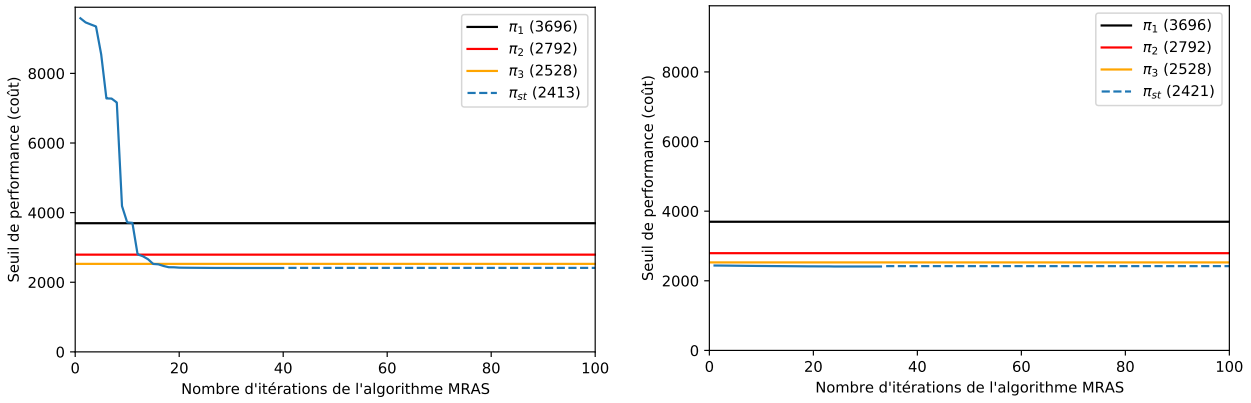
TABLE 4.10 – Résultats numériques en fonction de l'a priori sur la distribution initiale P_0, dans le cas stationnaire basé sur les modes, sur un horizon de 520 semaines.

A priori	uniforme	basé sur $\hat{P}_{st}$
Meilleur coût	2413	2421

Politiques associées. Dans le cas où la distribution P_0 est initialisée à partir de la distribution $\hat{P}_{st}$, la politique trouvée par l'algorithme MRAS correspond à la politique π^*_{st} explicitée dans la section précédente. Comparée aux coûts des politiques de référence, elle génère un gain relatif de 75% (resp. 35%, 13% et 4%) par rapport à la politique de référence π_0 (resp. π_1, π_2 et π_3). Son coût est très proche de celui de la politique stationnaire préventive π_3. Par contre, il est supérieur au plus faible coût réalisé via la politique π_4 qui tient compte des temps de fonctionnement passés dans le mode dégradé pour déclencher les opérations à effectuer sur les composants du pod, qui est de 2217, ce qui représente environ 9% d'écart relatif. Cette différence s'explique par le fait que la politique de référence π_4 n'appartient pas au sous-ensemble de politiques étudié, qui est ici uniquement basé sur les modes du pod.

Ensuite, dans le cas où la distribution initiale P_0 est uniforme sur les actions possibles par état, le gain relatif par rapport aux politiques de références est du même ordre de grandeur. Aussi, la politique trouvée par l'algorithme MRAS se distingue de la politique π^*_{st} par 5 actions différentes. Il s'agit des modes 7 et 11, où un composant du pod est en panne et un composant est dans un mode dégradé. Dans ces modes, la politique indique d'aller en atelier pour intervenir sur les deux composants, là où la politique π^*_{st} indiquait de n'intervenir que sur le composant en panne. Ces derniers ont été évoqués dans le paragraphe précédent comme étant très peu observés lorsqu'on applique la politique π^*_{st}. Aussi, le gain provient du fait qu'en étudiant le système sur un horizon plus long, ces modes sont plus souvent atteints, ce qui conduit à mieux estimer l'action à effectuer dans ces états.

Évolution du seuil de performance. On s'intéresse ensuite à l'évolution du seuil de performances au cours des itérations de l'algorithme, présentée dans la Figure 4.2. On note que les deux approches mènent à considérer initialement des politiques dont les coûts sont très différents. On observe qu'avec une distribution P_0 initialisée via $\hat{P}_{st}$, l'algorithme s'oriente en moins d'itérations vers son minimum. Par contre, avec une distribution initiale P_0 uniforme, bien que les premières politiques candidates aient un coût nettement plus élevé, on parvient finalement à déterminer une politique dont le coût est légèrement plus faible.



(a) Distribution initiale P_0 uniforme.

(b) Distribution initiale P_0 basée sur $\hat{P}_{st}$.

FIGURE 4.2 – Évolution du seuil de performances (coût) en fonction du nombre d'itérations de l'algorithme MRAS, dans le cas stationnaire basé sur les modes, à horizon 520 semaines.

Finalement, utiliser l'approche de *splitting-initializing* permet de retrouver la politique optimale π^*_{st}, ce qui est attendu puisque $\hat{P}_{st}$ concentre majoritairement les probabilités sur une seule action par mode. Si elle mène à un coût sous-optimal, puisqu'on trouve un coût meilleur de 0.33% via une distribution P_0 uniforme, la faible différence montre que la politique π^*_{st} passe à l'échelle pour un plus grand nombre de décisions.

Tous les résultats de ce chapitre sont ensuite donnés pour un horizon de 52 semaines, et on poursuit les recherches d'approximation de la fonction valeur sur l'ensemble des politiques non stationnaires basées sur les modes du système.

4.1.3.4 Résultats de la méthode MRAS pour le cas non stationnaire

On évalue à présent l'infuence de 4 hyperparamètres de l'algorithme MRAS sur sa dynamique, afin d'ajuster leurs valeurs, dans le cas non stationnaires basées sur les modes. On rappelle que dans ce cas, $P_0 \in \mathcal{M}_{\#\mathbb{X};\#\mathbb{A};H}([0;1])$, et est encore initialisée à une loi uniforme.

Robustesse des résultats. Si les résultats présentés dans le cas stationnaire sont très stables à paramètres fixées, on observe dans le cas non stationnaire que certaines combinaisons de paramètres mènent à de la variabilité sur les résultats. Ceci s'explique par le fait que la méthode explore un espace de politiques beaucoup plus grand, et serait plus sensible aux premières politiques candidates. À titre d'exemple, on note que si ν est grand et λ petit, une variance s'observe sur le coût, la dernière itération utile, et le nombre de politiques candidates à la dernière itération, qui évoluent conjointement au cours des itérations. Les valeurs de ces indicateurs, obtenues via plusieurs exécutions de l'algorithme MRAS, lancées les mêmes paramètres arbitrairement fixés pour les premiers essais, sont données Table 4.11.

Les coûts sont globalement décroissants avec le nombre de politiques, lui même croissant avec la dernière itération utile : chercher pendant plus d'itérations via un plus grand nombre de politiques permet de mieux explorer l'espace des politiques. Aussi, on peut supposer que si le seuil de performance n'est plus mis à jour, c'est qu'il a appris de politiques peu performantes lors des premières itérations, auxquelles l'algorithme serait sensible. Ceci s'illustre sur l'exécution 1, qui présente une moins bonne performance que les exécutions 2 et 3, tout en ayant fait davantage d'itérations et bien qu'ayant effectué ses recherches sur davantage de politiques : si ν est grand et λ petit, les paramètres ne permettent pas de s'orienter vers de meilleurs coûts, et les temps de calculs explosent, ce qui légitime le critère d'arrêt fixé.

Idéalement, en plus d'évaluer l'influence des paramètres via le plan d'expérience, il faudrait lancer plusieurs exécutions pour chaque combinaison de paramètres afin d'évaluer la sensibilité, mais les temps de calculs élevés ne permettent pas de le réaliser sur le problème étudié. Notons toutefois qu'en dehors du coût de 188, la variance sur les résultats est de l'ordre de 1%, ce qui confirme l'idée d'une tendance observée à paramètres fixés. Notons enfin que la meilleure performance (188) réalise un gain relatif de 20% (resp. 10%) sur les politiques de référence basées sur les modes (resp. au minimum trouvé dans le cas stationnaire).

TABLE 4.11 – Résultats numériques en fonction de l'exécution de l'algorithme MRAS, dans le cas non stationnaire basé sur les modes, avec $\epsilon = 1$, $\nu = 0.4$, $\lambda = 0.05$, $\rho_0 = 0.2$.

Exécution	1	2	3	4
Meilleur coût	210.39	209.08	207.54	**188.04**
Dernière itération utile	28	13	18	61
Nombre de politiques candidates	1141	270	437	20120

Finalement, on réutilise le plan d'expérience présenté paragraphe 4.1.2.2, afin de mesurer si, comme dans le cas stationnaire, une tendance apparait sur les hyperparamètres. Via l'ajustement des hyperparamètres, l'objectif reste d'explorer l'espace des politiques non stationnaires basées sur les modes, pour approcher la fonction valeur ainsi qu'une politique qui la réalise, afin d'en interpréter une stratégie de maintenance applicable industriellement.

Paramètre de tolérance ϵ. Les résultats numériques en fonction du paramètre ϵ sont donnés Table 4.12, où on décrit le plus faible coût trouvé par la méthode MRAS ainsi que la dernière itération où ce seuil a été mis à jour. Contrairement au cas stationnaire, on note que la dernière itération utile intervient bien plus tard, ce qui peut notamment s'expliquer par le fait que la méthode explore un espace de politiques beaucoup plus grand. Aussi, en tenant compte de politiques non stationnaires, on obtient dans le meilleur des cas un gain relatif de 15% par rapport à la meilleure politique de référence basée sur les modes, et de 6% par rapport au minimum trouvé dans le cas stationnaire. Par contre, on n'observe pas de tendance particulière avec ce paramètre, bien que les résultats soient nettement influencés par sa valeur. Par exemple, on note pour ϵ=10 qu'aucune itération n'a permis de mettre à jour le seuil de performance, qui est nettement supérieur aux autres, et même supérieur au minimum trouvé dans le cas stationnaire, alors qu'approcher la fonction valeur sur un espace de politiques plus grand conduit a priori à obtenir de meilleures performances. Ceci illustre l'intéret de bien choisir les hyperparamètres de l'algorithme. Finalement, on choisit de garder la valeur $\epsilon = 0.5$ pour les tests suivants, qui a conduit au coût le plus faible.

TABLE 4.12 – Résultats numériques en fonction de ϵ pour l'algorithme MRAS, avec $\nu = 0.2$, $\lambda = 0.2$, et $\rho_0 = 0.25$, dans le cas non stationnaire basé sur les modes.

ϵ	Meilleur coût	Dernière itération utile
0.1	203.76	62
0.5	**197.46**	70
1	198.58	54
10	214.43	0

Coefficient ν. Les résultats numériques en fonction de ν sont résumés Table 4.13. On observe ici des seuils du même ordre de grandeur que pour les tests effectués avec ϵ, mais on peut noter que les seuils de performance sont croissants avec la valeur du paramètre ν, comme observé précedemment dans le cas stationnaire. Aussi, le nombre d'itérations utiles est encore décroissant avec ν. Ceci conduit à nouveau à garder la plus faible valeur, $\nu = 0.2$, pour toutes les expériences qui suivent. Enfin, on observe que, cette valeur étant fixée, les résultats qui suivent semblent beaucoup moins variables que ceux présentés Table 4.11, ce qui indique à nouveau qu'un choix de ν et λ faibles mène à de meilleures performances.

TABLE 4.13 – Résultats numériques en fonction de ν pour l'algorithme MRAS, avec $\epsilon = 0.5$, $\lambda = 0.2$, et $\rho_0 = 0.25$, dans le cas non stationnaire basé sur les modes.

ν	Meilleur coût	Dernière itération utile
0.2	**197.46**	70
0.4	207.32	20
0.6	208.15	11
0.8	214.39	13

Coefficient de mélange λ. Les résultats numériques en fonction de λ sont repris dans la Table 4.14. Encore une fois, on a des performances du même ordre de grandeur, mais on observe toutefois une légère amélioration globale : tous les seuils observés dans cette expérience sont plus proches de la meilleure performance observée jusqu'ici, qui est de 197.46, et tous sont meilleurs que le minimum trouvé dans le cas stationnaire. On note aussi que le coût le plus faible est obtenu pour la plus petite valeur du paramètre λ, comme observé dans le cas stationnaire, ce qui conduit à nouveau à choisir la plus faible valeur, $\lambda = 0.2$.

TABLE 4.14 – Résultats numériques en fonction de λ pour l'algorithme MRAS, avec $\epsilon = 0.5$, $\nu = 0.2$, et $\rho_0 = 0.25$, dans le cas non stationnaire basé sur les modes.

λ	Meilleur coût	Dernière itération utile
0.2	**197.46**	70
0.4	199.82	45
0.6	204.76	24
0.8	199.39	75

Proportion initiale ρ_0. Les résultats numériques en fonction de ρ_0 sont donnés dans la Table 4.15. À nouveau, les coûts obtenus sont ici du même ordre de grandeur. Ce paramètre détermine la proportion initale de politiques candidates gardées pour la mise à jour de $\hat{P}$, et présente ici la meilleure performance pour la plus petite valeur de $\rho_0 = 0.2$, contrairement à ce qu'on a observé dans le cas stationnaire. Ceci peut s'expliquer par le fait que l'algorithme peut s'orentier plus rapidement vers de meilleures solutions s'il n'apprend que d'une petite proportion des meilleures politiques candidates pour mettre à jour $\hat{P}$, plutôt que de tenir compte de beaucoup de politiques moins performantes lors des mises à jour. Ceci est aussi lié au fait qu'à chaque itération, une proportion λ de politiques candidates sont simulées depuis la distribution initiale P_0 pour continuer à explorer l'espace des politiques et corriger une éventuelle mauvaise orientation prise au cours des itérations, surtout lorsqu'on a calibré une faible valeur pour le coefficient de mise à jour ν. Pour finir, on garde la plus petite valeur $\rho_0 = 0.2$, qui a conduit au coût le plus faible observé sur l'ensemble des tests effectués.

TABLE 4.15 – Résultats numériques en fonction de ρ_0 pour l'algorithme MRAS, avec $\epsilon = 0.5$, $\nu = 0.2$, et $\lambda = 0.2$, dans le cas non stationnaire basé sur les modes.

ρ	Meilleur coût	Dernière itération utile
0.2	**197.46**	70
0.4	198.17	56
0.6	201.16	40
0.8	202.08	45

Conclusions du plan d'expérience. L'exécution de la méthode MRAS sur ce plan d'expérience a permis de construire une politique non stationnaire basée sur les modes générant un gain relatif de 15% par rapport aux meilleures politiques de référence basées sur les modes, et de 6% par rapport à l'optimum trouvé via la méthode MRAS dans le cas stationnaire. Aussi, son coût d'environ 197 est similaire au meilleur côut obtenu via la politique de référence π_4 tenant compte des durées de vie des composants, qui est de 196. Ceci suggère que, le système partant toujours de l'état neuf dans mes simulations, les politiques non stationnaires parviennent à capturer une partie d'information sur la durée de fonctionnement des composants, en différenciant l'action à choisir sur le pod selon la date de décision.

D'après ces résultats numériques, les hyperparamètres de l'algorithme MRAS semblent avoir eu ici une plus grande influence sur les performances obtenues que dans le cas stationnaire, bien qu'on observe globalement les mêmes tendances pour les paramètres ν et λ. Mais à nouveau, rien ne garantit que le minimum trouvé au cours de ces expériences soit la fonction valeur de ce problème. Finalement, on s'intéresse aux interprétations que peuvent fournir ces résultats en terme d'application industrielle.

Interprétations des résultats. On détaille maintenant les sorties numériques obtenues via la méthode MRAS. En particulier, on s'intéresse à la meilleure politique obtenue avec la distribution $\hat{P}_{nst}$, à l'évolution du seuil de performance au cours des itérations de l'algorithme, ainsi qu'à la convergence de la distribution $\hat{P}_{nst}$. Notre objectif est alors d'expliciter la meilleure politique obtenue avec $\hat{P}_{nst}$, notée par la suite π^*_{nst}, afin de l'interpréter industriellement. Pour cela, l'objectif de ce chapitre étant d'approcher la fonction valeur et une politique associée permettant d'expliciter des recommandations industrielles, on détaille les résultats associés à la meilleure performance obtenue, c'est-à-dire ceux ayant conduit au coût de **188**, bien que le plan d'expérience n'ait pas mené à le retrouver. Aussi, on présente les statistiques telles que la répartition du coût moyen de la politique entre pénalités et opérations de maintenance, ainsi que le nombre et la répartition des pannes observées, et on les compare avec les résultats obtenus dans le cas stationnaire.

Politique non stationnaire optimale basée sur les modes. Le premier constat est que la politique trouvée est effectivement non stationnaire : le choix de l'action à effectuer dans un mode dépend de la date de décision, ce qui la rend moins facilement interprétable.

Premièrement, on observe des effets de bord : à l'horizon, il y a plusieurs modes pour lesquels la politique indique de ne pas aller en atelier en cas de dégradation ou de panne. Pour ces modes (2, 4, 5, 9, 11 et 15), ceci peut s'expliquer par le fait qu'une visite en atelier coûte plus cher que la pénalité d'indisponibilité associée aux modes de panne : le coût d'un passage en atelier est au minimum de 60, alors que la pénalité n'est que de 20. Toutefois, on peut se demander pourquoi ce comportement n'est pas partagé par tous les modes : peut-être parce que la politique conduit à observer très rarement ces autres modes. On note ensuite la même tendance à éviter d'aller en atelier lorsque la date de décision se rapproche de l'horizon, dans la plupart des modes, a priori pour la même raison.

Ensuite, la principale différence observée par rapport au cas stationnaire concerne les modes de dégradation multiples. Bien qu'on observe globalement le même comportement, c'est-à-dire d'entretenir tous les composants qui sont en mode dégradé dans ces cas, on note aussi une propension à choisir parfois une action de maintenance qui ne ramène pas le système à l'état neuf, voir à ne pas revenir du tout en atelier, selon la date de décision. La politique présente ce qui semble être des cycles, bien qu'aucune fréquence ne soit particulièrement régulière ni commune aux modes pour être facilement interprétable. Elle est détaillée en Annexe A.2, Table A.2. On peut toutefois supposer que, le système partant toujours de l'état neuf dans mes simulations, cette politique capte une part d'information sur la durée de vie des composants. C'est ce qu'on tentera d'interpréter plus finement dans la section 4.2.

Détails des côuts de la politique π^*_{nst}. On détaille la décomposition du coût moyen de la politique π^*_{nst} dans la Table 4.16. La politique π^*_{nst} réduit encore à la fois les coûts de maintenances et les coûts d'échec par rapport aux politiques de références et à la politique stationnaire optimale π^*_{st}. Ceci indique a priori que le nombre moyen de pannes a encore pu être réduit par rapport au cas stationnaire, sans que les opérations préventives augmentent les coûts de maintenance.

TABLE 4.16 – Détails des coûts de la politique optimale non stationnaire basée sur les modes du pod π^*_{nst}, sur un horizon de 52 semaines.

	π^*_{nst}	π^*_{st}
pénalité échec	10	20
coût maintenance	178	190
coût moyen	**188**	210
IC 95%	[187 ; 189]	[209 ; 211]

Statistiques de pannes. Le nombre moyen de pannes et le pourcentage de trajectoires sans panne observés sous la politique π^*_{nst} sont donnés dans la Table 4.17. On note que le nombre moyen de pannes est réduit d'environ 68.9% par rapport à la politique stationnaire optimale π^*_{st}. De plus, on observe que le pourcentage de trajectoires sans panne est lui nettement augmenté, plus que doublé, par rapport à la politique π^*_{st}.

TABLE 4.17 – Nombre moyen de pannes par trajectoire et pourcentage de trajectoires sans panne, pour la politique π^*_{st}, sur un horizon de 52 semaines.

	π^*_{nst}	π^*_{st}
nombre de pannes	**0.32**	1.03
% trajectoires sans panne	**71.6**	26.59

Répartition par mode de panne pour la politique π^*_{nst}. La répartition par mode de panne du pod est détaillée Table 4.18. Les modes de panne les plus visités (18, 6 et 2) correspondent encore aux pannes simples des composants 1, 2 et 3 respectivement, avec une majorité (50%) de pannes du composant 1. Cette répartition montre que les autres modes de pannes sont plus rarement observés, et en particulier les modes 5, 7, 11, 14, 16 et 22. On note toutefois que les pourcentages ont légèrement augmenté dans les modes 11, 15, 19 et 21, qui correspondent aux cas où un composant est en panne et un autre est dégradé. Ceci s'explique par le fait que si les dégradations multiples ne sont pas systématiquement traitées, alors ces cas pourraient apparaitre légèrement plus souvent, bien que cela n'ait pas conduit à un coût plus élevé que celui de la politique stationnaire.

TABLE 4.18 – Répartition des pannes de trajectoires contrôlées par la politique π^*_{nst}, en fonction du mode de panne, sur un horizon de 52 semaines.

Mode	2	5	6	7	11	14	15	16	18	19	21	22
π^*_{nst}	**13.78**	0.86	**17.81**	1.11	2.48	0.13	4.10	0.14	**50.88**	3.92	4.59	0.20

Évolution du seuil de performance. L'évolution du seuil de performance en fonction du nombre d'itérations de l'algorithme est représenté dans la Figure 4.3. On note que dès les premières itérations, la méthode a considéré des politiques moins coûteuses que les politiques de références, et moins coûteuses que l'optimum trouvé dans le cas stationnaire. L'évolution observée indique qu'une politique non stationnaire est rapidement moins coûteuse qu'une politique stationnaire, puisque le meilleur coût obtenu dans le cas stationnaire est battu en quelques itérations. Aussi, la décroissance du coût est ensuite moins rapide vers le minimum trouvé, et ce dernier cesse de s'améliorer après 60 itérations, bien que la recherche se fasse sur un très grand nombre de politiques candidates.

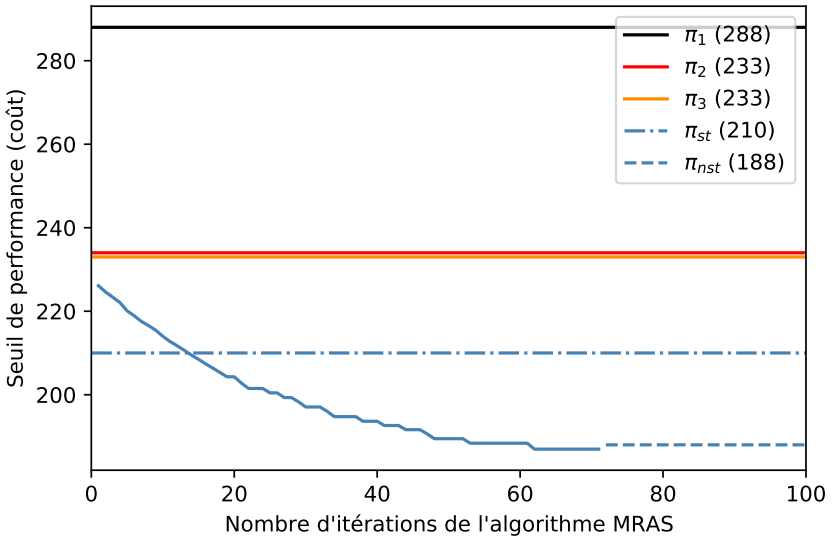


FIGURE 4.3 – Évolution du seuil de performance en fonction du nombre d'itérations de l'algorithme MRAS, dans le cas non stationnaire basé sur les modes.

Distribution de probabilités $\hat{P}_{nst}$. On s'intéresse maintenant à la distribution de probabilités $\hat{P}_{nst}$ obtenue à l'issue des itérations de la méthode MRAS. On note que pour tous les modes et toutes les dates de décisions, la distribution est loin de la distribution uniforme donnée en entrée de l'algorithme : de nombreux coefficients sont réduits à une valeur proche de 0. Dans la grande majorité des cas, les probabilités se concentrent sur une seule action par mode, bien qu'il y ait pour chaque date de décision un ou deux modes pour lesquels les probabilités sont réparties uniformément entre deux actions possibles.

On compare ensuite ces résultats avec ceux réalisés via la méthode ASA, en reprenant le plan d'expérience décrit dans la sous-section 4.1.2.

4.1.3.5 Résultats de la méthode ASA pour le cas non stationnaire

On évalue maintenant l'infuence de la température initiale T_0 sur la dynamique de l'algorithme ASA, pour ajuster sa valeur parmi la grille des 4 valeurs fixées uniformément.

Les résultats en fonction de T_0 sont résumés Table 4.19. On note que le seuil de performance est croissant avec T_0, et ces coûts sont similaires à ceux obtenus avec l'exécution de la méthode MRAS. Au mieux, cela permet de réaliser un gain de 15% par rapport aux politiques de références, et de 6% par rapport à l'optimum trouvé dans le cas stationnaire.

TABLE 4.19 – Résultats numériques en fonction du paramètre T_0 pour ASA, dans le cas non stationnaire basé sur les modes.

T_0	0.1	1	10	100
Meilleur coût	**197.29**	204.54	210.36	211.28

Politique associée. Globalement, cette politique sest proche de celle déterminée via la méthode MRAS. Elle est non stationnaire et présente des effets de bord similaires, c'est-à-dire d'éviter les retours en atelier à l'approche de l'horizon. Ensuite, on note la même tendance à entretenir tous les composants en mode dégradé en cas de dégradations multiples, bien qu'on note encore une propension à choisir parfois une action de maintenance qui ne ramène pas le système à l'état neuf, selon la date de décision. Néanmoins, à nouveau, les cycles observés ne présentent pas de fréquence particulière qui permettrait d'identifier une stratégie facilement interprétable. Enfin, la politique préconise de mutualiser les opérations si le pod est en panne avec un ou deux composants dégradés : si les dégradations multiples ne sont pas systématiquement traitées, ces cas pourraient apparaitre légèrement plus souvent et conduire à préciser la stratégie pour ces cas. Elle est détaillée en Annexe A.2, Table A.3.

Distribution de probabilités $\hat{P}$. Contrairement à ce qu'on a observé avec l'algorithme MRAS, les distributions ne semblent pas avoir convergé vers une seule action possible pour la plupart des modes. La méthode ASA serait plus lente à converger, ce qui peut expliquer la différence de coût obtenue, bien que les deux politiques associées soient assez proches. Finalement, les résultats de la méthode ASA ne permettent pas d'interpréter davantage la stratégie que ceux de la méthode MRAS.

Abandon des recherches avec la méthode ASA. Pour la méthode MRAS, on a pu fixer un critère d'arrêt empirique basé sur l'arrêt de l'évolution du seuil de performance pendant 10 itérations consécutives, afin d'éviter une explosion en temps de calcul de l'algorithme MRAS. Aussi, la méthode ASA ne faisant pas évoluer de seuil de performance, on n'a pas d'autre critère que le nombre d'itérations (initialisé à 100) pour interrompre l'algorithme, ce qui a conduit à des temps de calculs très longs, de l'ordre de 24h par exécution pour les résultats résumés dans la Table 4.19. Considérant que les calculs ont été plus coûteux sans pour autant mener à de meilleures performances que la méthode MRAS, on décide de ne pas poursuivre nos recherches via la méthode ASA.

Conclusions. Finalement, les recherches ont conduit à une performance de 188, réalisant un gain relatif d'environ 20% (resp. 10%) par rapport à la meilleure politique de référence basée sur les modes (resp. au minimum trouvé dans le cas stationnaire). La politique associée permet plusieurs interprétations : la politique optimale est non stationnaire, adapte les actions à l'approche de l'horizon, et a une tendance à mutualiser les interventions plutôt que d'intervenir précocément dès la première dégradation, tout en évitant les pannes.

Ensuite, on poursuit nos recherches sur l'approximation de la fonction valeur sur l'ensemble des politiques basées sur une partition des temps de fonctionnement des composants du système. L'objectif est alors d'améliorer les performances mais également de tenter d'interpréter une politique associée en terme de recommandations applicables industriellement.

4.2 Approximation de la fonction valeur basée sur une discrétisation des temps de fonctionnement

La deuxième partie de ce travail consiste à approcher la fonction valeur en tenant compte des temps de fonctionnement via une partition de leur domaine de définition, c'est à dire d'associer une même action à tous les états dans un même mode et dont les temps de fonctionnement ont le même plus proche voisin. Ceci donne un espace d'états fini pour les prises de décisions, sans discrétiser la dynamique du MDP ni cumuler d'erreurs de projection. Les meilleures politiques trouvées lors des recherches tenant compte du mode du pod semblant tenir compte de ces temps de fonctionnements, sans qu'on puisse facilement l'interpréter, l'objectif est ici non seulement d'approcher la fonction valeur sur un espace de politiques plus grand, mais aussi de pouvoir interpréter une politique associée.

L'espace des politiques étant différent de celui dont les politiques ne se basent que sur les modes, la mise en oeuvre de l'algorithme MRAS pour ce problème d'optimisation requiert d'expliciter le sous-ensemble de politiques considéré, ainsi que le simulateur permettant de simuler des trajectoires contrôlées par toute politique de cet ensemble. Aussi, on s'intéresse d'abord au cas stationnaire afin d'interpréter plus facilement les politiques déterminées via la méthode MRAS en terme d'application industrielle, puis au cas non stationnaire qui peut permettre d'améliorer les performances et compléter les interprétations.

Dans les deux cas, on définit le problème d'optimisation en explicitant l'espace des politiques considéré, ainsi que le simulateur associé à cet ensemble de politiques, section 4.2.1. Le choix d'une partition est ensuite détaillé dans la section 4.2.2. Enfin, on précise le plan d'expérience permettant d'ajuster les hyperparamètres de l'algorithme MRAS dans la section 4.2.3, et on présente les résultats obtenus, ainsi que leur comparaison avec les politiques de référence et les résultats des recherches précédentes.

4.2.1 Définition du problème d'optimisation

Pour approcher la fonction valeur en tenant compte des temps de fonctionnement des composants du pod via une partition de leur domaine de définition pour la prise de décisions, on explicite l'espace d'états associé. Ceci permet ensuite d'expliciter la distribution de probabilités initiale P_0, définie par l'ensemble des actions possibles par état sur l'espace considéré, et qui détermine l'espace des politiques sur lequel la méthode MRAS cherche à approcher la fonction valeur.

L'objectif de cette section est alors de préciser d'une part le sous-espace d'états fini choisi pour les prises de décisions faites sur le système, ainsi que la distribution de probabilités sur les actions possibles par état de l'espace considéré, et d'autre part un simulateur du processus contrôlé par les politiques admissibles du problème considéré.

Sous-espace d'états fini pour la prise de décisions. Afin de clarifier ma démarche, je désigne par *régime* les éléments du sous-ensemble d'états considéré. Aussi, comme défini lors des recherches basées sur les modes, puisqu'une seule action est possible lorsque le pod est dans l'atelier suite à une ou plusieurs maintenances, c'est-à-dire lorsque $1 < t_a < d_a - 1$, on garde un unique régime associé. De même, étant donné qu'une seule action est possible pour un composant en mode stable, on garde un unique régime associé par composant, sans tenir compte du temps passé dans ce mode. De plus, puisqu'un composant en panne ne fonctionne plus, aucun temps de fonctionnement n'est à considérer dans ce mode, ce qui revient également à garder un unique régime associé par composant. Finalement, la partition des temps de fonctionnement des composants se porte sur ceux étant dans un mode dégradé.

Notons que le nombre de points de cette partition est un hyperparamètre à déterminer. Pour ce faire, on choisit d'une part de se restreindre à une unique grille par composant commune à toutes les dates de décisions, et d'autre part de considérer le même nombre de points pour chaque composant dans la grille associée. On a finalement un unique hyperparamètre à déterminer, le nombre de points pour discrétiser la durée de vie de chacun des composants, qu'on note par la suite D. Ceci permet alors de déterminer le nombre de régimes du sous-espace d'états fini basé sur cette partition, en fonction de D, noté par la suite $\mathbb{X}_D$.

Simplement, plutôt que de baser les décisions sur 3 modes par composant (stable, dégradé, panne), on considère un certain nombre de régimes : 1 stable, D dégradés, et 1 de panne par composant, auquel s'ajoute le régime global en atelier. Ceci donne $\#\mathbb{X}_D < (1 + D + 1)^3 + 1$ régimes sur lesquels baser les prises de décisions, puisque certains sont inatteignables.

Numérotation des régimes et projection au plus proche voisin. On peut ainsi se ramener à un problème de la même forme qu'à la section 4.1, si les points de la partition sont triés par ordre croissant : on associe un régime 0 à un composant stable, l'indice de son plus proche voisin après projection sur la partition pour un composant dégradé, et un régime $D+1$ à un composant en panne. On peut numéroter les régimes avec l'ordre lexicographique utilisé Table 3.1, et utiliser à nouveau la bijection entre le triplet de régimes des composants et le numéro de régime du pod.

Distribution de probabilités sur les actions possibles par régime $P_0 \in \mathcal{M}_{\#\mathbb{X}_D;\#\mathbb{A};H}([0;1])$. Cette distribution est déterminée par l'ensemble des actions possibles par régime du pod. Notons que l'espace d'actions est inchangé. Aussi, les contraintes sur les actions possibles par état définies sous-section 3.1.4 sont, pour un mode donné, les mêmes quelle que soit la durée de fonctionnement dans ce mode. L'ensemble des actions possibles par régimes du système est alors similaire à celui résumé dans la Table 4.1, bien que ses dimensions soient différentes. La distribution initiale P_0 est ensuite initialisée à une loi uniforme (telle que tous les coefficients non nuls associés à un régime $r \in \mathbb{X}_D$ sont $\frac{1}{\#\mathbb{A}(r)}$).

Simulateur sur l'ensemble des politiques admissibles associé à $\mathbb{X}_D$. On propose ici un pseudo-code permettant de générer des trajectoires contrôlées par toute politique qui tienne compte du temps de fonctionnement des composants en mode dégradé via une partition. Ce dernier s'appuie encore sur le fait que trois types d'action sont possibles sur le pod (aller en mission, effectuer une action de maintenance, rester en atelier), et sur les algorithmes décrivants la dynamique du MDP selon l'action choisie, indroduits sous-section 3.3.2.

Aussi, les simulations se basent sur la projection de l'état courant du MDP sur l'ensemble des régimes du sous-espace d'états fini $\mathbb{X}_D$. Ainsi, l'objectif est d'associer à tous les états dans un même régime la même action à effectuer sur le système, sans jamais discrétiser la dynamique du MDP ni cumuler d'erreurs de projection. Pour ce faire, les régimes sont numérotés, ce qui permet à nouveau d'utiliser alternativement : le régime projeté r pour la prise de décision et l'état réel x lors des transitions qui en résultent.

Algorithme 12 : Trajectoire du processus discrétisé

Entrées : Horizon, politique π, partition à D points.

début
- m, ta, cout, t $\leftarrow$ 0, 0, 0, 0_3 ; x=(m, t1, t2, t3, ta) ; `// Initialisation`
- **pour** i **de** 1 **à** 3 **faire** `// Tirer temps de vie restant en mode stable`
  - tr[i] $\leftarrow \mathcal{W}(\alpha_{s_i}, \beta_{s_i})$
- **pour** date de décision **de** 0 **à** horizon-1 **faire**
  - $r \leftarrow$ **Projection**(x, partition)
  - a $\leftarrow \pi[r,$ date de décision$]$; `// choix d'action selon la politique`
  - **si** a $= 0$ **alors** `// i.e. mission`
    - (x, cout ; tr) $\leftarrow$ mission (x, cout ; tr) ; `// voir algorithme 3`
  - **sinon si** $0 <$ a < 27 **alors** `// i.e. action de maintenance`
    - (x, cout ; tr) $\leftarrow$ opérations(x, a, cout ; tr) ; `// voir algorithme 4`
  - **sinon** `// i.e. en maintenance`
    - (x, cout ; tr) $\leftarrow$ maintenance(x, a, cout ; tr) ; `// voir algorithme 5`

Sorties : cout

Remarque 4.2.1. À nouveau, si on considère une politique stationnaire, π est un vecteur au lieu d'être une matrice, et il suffit de remplacer $a \leftarrow \pi[$r, date de décision$]$ par $a \leftarrow \pi[r]$. De la même façon, si on recherche une politique optimale stationnaire (respectivement non stationnaire) avec la méthode MRAS, alors la distribution de probabilités initiale sur les actions possibles par état s'écrit $P_0 \in \mathcal{M}_{\#\mathbb{X}_D;\#\mathbb{A}}([0;1])$ (respectivement $P_0 \in \mathcal{M}_{\#\mathbb{X}_D;\#\mathbb{A};H}([0;1])$).

Ce simulateur permet de générer des trajectoires contrôlées sur l'ensemble des politiques admissibles associée à $\mathbb{X}_D$, mais requiert d'avoir sélectionné au préalable l'hyperparamètre D ainsi qu'une partition. Pour ce faire, le choix d'une partition est détaillé sous-section 4.2.2.

4.2.2 Choix d'une partition

L'objectif de cette section est de détailler le choix de partition utilisé dans ce travail. Puisqu'il s'agit de discrétiser le temps passé dans un mode dégradé, on va discrétiser la distribution des variables aléatoires de Weibull qui déterminent la passage d'un mode dégradé à un mode de panne, de chacun des composants. Pour ce faire, on va utiliser la méthode de quantification d'une variable aléatoire, présentée section 2.3. À partir de cette partition de Voronoï, dont le nombre de points D est à déterminer, on va associer à tous les états dégradés ayant le même plus proche voisin, la même action lors de la prise de décision.

Afin de déterminer le nombre de points D de la partition, on choisit de sélectionner celui qui réalise un compromis entre minimiser l'erreur commise lors de la projection des états sur $\mathbb{X}_D$, qui conduit à un nombre de points élevé ainsi qu'à une cardinalité élevée de l'espace des régimes, et complexité numérique, qui conduit à considérer un espace de régimes le plus petit possible.

Pour mesurer l'erreur commise, on intègre cette étape de projection des états sur $\mathbb{X}_D$ au simulateur de trajectoires contrôlées par la politique π_4, ce qui permet de comparer les coûts obtenus avec et sans projection (résumés dans la Table 3.8). On rappelle que la politique π_4 s'appuie sur le dépassement d'un quantile du temps passé dans un mode dégradé pour déclencher le retour préventif en atelier du système. Aussi, sa version restreinte à l'espace $\mathbb{X}_D$ consiste à projeter le temps de fonctionnement des composants dégradés sur la partition, puis à comparer si le temps projeté est plus élevé que le quantile, plutôt que d'utiliser le temps de fonctionnement réel pour déclencher le retour préventif en atelier. La transition se fait ensuite sur le vrai état, ce qui permet de discrétiser la règle de décision sans discrétiser la dynamique du MDP, afin d'éviter le cumul d'erreurs de projection. Les politiques de référence π_0 à π_3 ne se basant que sur le mode du pod pour effectuer les prises de décisions, elles associent la même action à tous les régimes dégradés, ce qui ne génère pas d'erreur de projection et conduit à se concentrer sur la politique de référence π_4, évaluée pour différentes valeurs du paramère $p \in [0, 1]$.

Plus précisément, l'objectif de ce chapitre étant d'approcher la fonction valeur du MDP, l'erreur est basée sur la plus grande différence observée entre le coût associé à un quantile pour la politique π_4 et le coût de sa version projetée, puis exprimée en pourcentage d'erreur relative. Aussi, on évalue cette erreur en estimant le coût des politiques par méthode de Monte-Carlo via 10^5 simulations, pour différentes valeurs du paramètre D, afin de déterminer un nombre de points qui réalise ce compromis entre faible erreur d'approximation de la fonction valeur et complexité numérique. Enfin, étant donné que la quantification est connue pour sa sensibilité à son initialisation, qui est stochastique, on choisit de générer (arbitrairement) 20 partitions pour chaque valeur de D et on garde celle qui a mené à la plus petite erreur correspondant à ce critère. Les résultats sont résumés dans la Table 4.20.

TABLE 4.20 – Erreur de projection en fonction du nombre de points D de la partition.

D	% erreur	$\#\mathbb{X}_D$
2	18.17	65
3	12.86	126
4	2.86	217
5	1.68	344
6	**0.56**	**513**
7	0.53	730
8	0.57	1001
9	0.72	1332
10	0.56	1729

On observe que l'erreur est décroissante en fonction du nombre de points jusqu'à $D = 7$, puis ne diminue plus tandis que la taille de l'espace d'états augmente. On choisit une partition à $D = 6$ points, qui conduit à une erreur relative inférieure à 1% et permet de garder des résultats numériques précis pour l'évaluation des politiques candidates, avec $\#\mathbb{X}_D = 513$.

Remarque 4.2.2. On rappelle que sur les 27 modes, seuls 20 sont effectivement atteignables par le processus. De la même façon, sur les 513 régimes de $\mathbb{X}_D$, seuls 491 sont atteignables. Sur ce sous-problème, il existe alors $\prod_{r \in \mathbb{X}_D} \#\mathbb{A}(r) \approx 10^{180}$ politiques stationnaires et 10^{180H} politiques non stationnaires possibles.

Les déciles de temps de fonctionnement ainsi que les 6 points de la partition choisie, représentés en fonction de la proportion de temps de fonctionnement passée dans le mode dégradé et les intervalles associés pour les prises de décisions et pour chaque composant, sont représentés dans la Figure 4.4. Comparée aux déciles de temps de fonctionnement utilisés comme seuils de déclenchement des maitenances pour la politique π_4, la grille choisie permet de considérer une même gamme de temps de fonctionnement en le moins de points possibles, conduisants aux mêmes coûts.

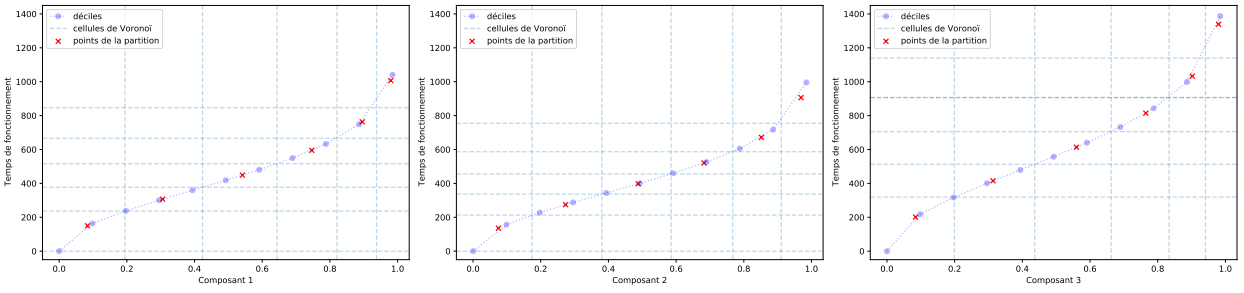


FIGURE 4.4 – Déciles de temps de fonctionnement et points de la partition choisie, en fonction de la proportion de temps de fonctionnement passée dans le mode dégradé, et intervalles associés pour les prises de décisions.

4.2.3 Plan d'expériences et résultats numériques

On s'intéresse maintenant à l'approximation de la fonction valeur sur l'ensemble des politiques basées sur les temps de fonctionnement des composants, via la partition choisie. Pour ce faire, on garde les valeurs d'hyperparamètres précédement fixées : le nombre d'itérations maximal est fixé à $K = 100$, le nombre de politiques candidates initial à $N_0 = 100$, le nombre initial de simulations à $M_0 = 10^4$. Pour finir, on garde les coefficients $\alpha = 1.1$ et $\beta = 1.025$.

Aussi, le plan d'expérience présenté pour l'optimisation basée sur les modes dans la sous-section 4.1.2 étant très coûteux en temps de calcul, on se restreint par la suite à effectuer une recherche d'hyperparamètres sur des grilles de 3 valeurs, recalibrées à partir des résultats obtenus lors des recherches basées sur les modes du pod.

Ainsi, la plus grande valeur du paramètre ϵ ayant donné, comme attendu, la moins bonne performance, on ne garde par la suite que les 3 plus faibles valeurs testées. Ensuite, on a observé pour les paramètres ν et λ une tendance à donner des performances croissantes, aussi, on va concentrer nos recherches sur des valeurs plus faibles de ces hyperparamètres. Enfin, le paramètre ρ_0 n'ayant pas montré de comportement particulier, on se restreint à 3 valeurs réparties uniformément. Finalement, on lance un plan d'expérience sur les grilles de valeurs suivantes, et on présente les résultats numériques obtenus.

— $\epsilon \in \{0.1,\ 0.5,\ 1\} \subset \mathbb{R}_+^*$;

— $\nu \in \{0.10,\ 0.20,\ 0.30\} \subset]0; 1[$;

— $\lambda \in \{0.05,\ 0.10,\ 0.20\} \subset]0; 1[$;

— $\rho_0 \in \{0.25,\ 0.50,\ 0.75\} \subset]0; 1]$.

4.2.3.1 Résultats de l'algorithme MRAS pour le cas stationnaire

On présente maintenant les résultats numériques correspondants à ce plan d'expérience, dans le cas stationnaire. À nouveau, tous les paramètres sont initialisés arbitrairement à leur plus petite valeur de la grille, puis la valeur ayant conduit au plus faible coût lors d'une étape est ensuite gardée pour tous les tests suivants.

Paramètre de tolérance ϵ. Les résultats numériques en fonction du paramètre ϵ sont donnés Table 4.21. Le coût est croissant, et la dernière itération utile décroissante avec ϵ. Aussi, ces coûts sont plus élevés que le minimum trouvé dans le cas non stationnaire basé sur les modes, bien que similaire à celui issu du plan d'expérience. Ceci peut s'expliquer par le fait qu'une politique stationnaire perd le bénéfice des cycles et effets de bord, et qu'approcher la fonction valeur sur un ensemble plus grand conduit à l'explorer moins facilement. Enfin, on garde la valeur $\epsilon = 0.1$ pour les tests suivants, qui a conduit au plus faible coût.

TABLE 4.21 – Résultats en fonction de ϵ pour l'algorithme MRAS, avec $\nu = 0.1$, $\lambda = 0.05$, et $\rho_0 = 0.25$, dans le cas stationnaire basé sur les temps de fonctionnement.

ϵ	Meilleur coût	dernière itération utile
0.1	**197.90**	66
0.5	198.12	27
1	199.46	19

Coefficient ν. Les résultats numériques en fonction de ν sont donnés dans la Table 4.22. On obtient des seuils du même ordre de grandeur que pour les tests effectués avec ϵ. On note encore que les coûts sont croissants, et le nombre d'itérations utiles décroissant avec ν. Ceci conduit à garder la plus faible valeur, $\nu = 0.1$, pour toutes les expériences qui suivent.

TABLE 4.22 – Résultats en fonction de ν pour l'algorithme MRAS, avec $\epsilon = 0.1$, $\lambda = 0.05$, et $\rho_0 = 0.25$, dans le cas stationnaire basé sur les temps de fonctionnement.

ν	Meilleur coût	dernière itération utile
0.1	**197.90**	66
0.2	198.08	46
0.3	198.57	29

Coefficient de mélange λ. Les résultats numériques en fonction du paramètre λ sont donnés Table 4.23. Les performances du même ordre de grandeur et présentent la même tendance, ce qui conduit à choisir la plus faible valeur, $\lambda = 0.05$.

TABLE 4.23 – Résultats en fonction de λ pour l'algorithme MRAS, avec $\epsilon = 0.1$, $\nu = 0.1$, et $\rho_0 = 0.25$, dans le cas stationnaire basé sur les temps de fonctionnement.

λ	Meilleur coût	dernière itération utile
0.05	**197.90**	66
0.10	198.20	36
0.20	198.43	28

Proportion initiale ρ_0. Les résultats numériques en fonction de ρ_0 sont donnés Table 4.24. Les coûts sont encore du même ordre de grandeur que ceux des tests précédents, mais légèrement améliorés en prenant une valeur plus élevée pour ρ_0. Il est cohérent de prendre en compte davantage de politiques pour que la mise à jour de P_k mène à de meilleures performances, étant donné que l'espace à explorer est très grand. Pour finir, on garde la plus grande valeur $\rho_0 = 0.75$, qui a présenté le coût le plus faible sur l'ensemble des tests effectués.

TABLE 4.24 – Résultats en fonction de ρ_0 pour l'algorithme MRAS, avec $\epsilon = 0.1$, $\nu = 0.1$, et $\lambda = 0.05$, dans le cas stationnaire basé sur les temps de fonctionnement.

ρ	Meilleur coût	Dernière itération utile
0.25	197.90	66
0.50	198.51	36
0.75	**196.71**	58

Conclusions. Les différentes exécutions de la méthode MRAS sur ce plan d'expérience ont permis de construire une politique non stationnaire basée sur une partition des temps de fonctionnement, qui génère un gain relatif d'environ 15% par rapport aux meilleures politiques de référence basées sur les modes, de 6% par rapport à la politique stationnaire basée sur les modes. Aussi, son coût est similaire à celui de la politique de référence π_4. Ceci indique a priori que différencier les actions selon la partition des temps de fonctionnement passés pour les composants dans un mode dégradé serait similaire au fait de différencier les actions selon les quantiles des temps de fonctionnement passés dans un mode dégradé pour les composants du pod.

Néanmoins, il faut noter que ce coût est plus élevé d'environ 5% que le minimum trouvé dans le cas non stationnaire basé sur les modes du pod. Ceci peut s'expliquer d'une part par le fait qu'une politique stationnaire perd le bénéfice des cycles et effets de bord en choisissant la même action dans un mode donné pour toutes les dates de décisions, et d'autre part par le fait qu'approcher la fonction valeur sur un espace d'états plus grand conduit à l'explorer moins facilement. Finalement, on s'intéresse aux interprétations que peuvent fournir ces résultats en terme d'application industrielle.

Interprétations des résultats. On détaille maintenant les différentes sorties numériques obtenues via l'exécution de la méthode MRAS. En particulier, on s'intéresse à la meilleure politique obtenue avec la distribution de probabilités terminale $\hat{P}_{st6}$, à l'évolution du seuil de performance au cours des itérations de l'algorithme, ainsi qu'à la convergence de $\hat{P}_{st6}$.

Notre objectif est d'expliciter la meilleure politique obtenue, notée par la suite π^*_{st6}, afin de préciser une stratégie de maintenance applicable industriellement. Pour ce faire, on détaille à nouveau les statistiques telles que la répartition du coût moyen de la politique entre pénalités d'échec et opérations de maintenance, ainsi que le nombre moyen et la répartition des différents types de pannes observés.

Enfin, on les compare avec l'ensemble des résultats numériques obtenus précedemment, et plus particulièrement avec la politique de référence préventive π_4, afin d'identifier les différences et points communs qui ont conduit à obtenir le même coût.

Politique stationnaire optimale basée sur les modes et une discrétisation des temps de fonctionnement des composants du pod. Le premier constat est que cette politique différentie effectivement les actions selon le régime dégradé du système observé. Aussi, la principale différence avec les politiques de références explicitée via les politiques π^*_{st} et π^*_{nst}, qui est d'entretenir tous les composants dégradés pour les modes sans panne avec des dégradations multiples, se retrouve dans cette politique. Ensuite, elle s'en distingue essentiellement en différenciant l'action selon le régime de dégradation observé. Enfin, la politique étant sous la forme d'un vecteur de taille $\#\mathbb{X}_D = 513$, on tente ici de l'expliciter plus en détails en distinguant des sous-groupes de régimes, notés r.

— si $r \in \mathcal{S}$, l'action choisie, qui est la seule action possible pour le système dans ce mode, est un envoi en mission.

— si $r \in \mathcal{D}$ avec un seul régime dégradé : pour les deux premiers régimes dégradés de chaque composant, la tendance est d'aller en mission, puis d'effectuer un entretien préventif pour les deux régimes suivants, et pour les deux derniers, on ne note pas de tendance particulière. Dans le premier cas, on l'explique par le fait que le composant peut encore effectuer plusieurs missions avant la survenue d'une panne. Dans le deuxième, ceci correspond à appliquer une stratégie de maintenance basée sur un dépassement de seuil, comme pour la politique π_4. Enfin, pour le dernier cas, on suppose à nouveau qu'il s'agit de modes très rarement observés si la politique conduit à intervenir sur les régimes dégradés précédents.

— si $r \in \mathcal{D}$ avec deux ou trois régimes dégradés : on observe un comportement similaire à celui décrit pour un unique mode dégradé. Lorsque tous les composants dégradés le sont dans le premier régime, l'envoi en mission est encore choisi, mais dès qu'un composant se trouve dans un autre régime dégradé, le risque de panne devient vraisemblablement trop grand pour retarder l'entretien préventif de tous les composants dégradés. Aussi, les actions à effectuer pour les dégradations multiples dans ces derniers régimes peut encore être un envoi en mission, ce qui peut encore s'expliquer par le fait que ces modes ne sont plus atteints via cette politique.

— si $r \in \mathcal{P}$ avec un ou deux régimes dégradés : la politique indique d'intervenir sur tous les composants, là où les politiques π^*_{st} et π^*_{nst} indiquaient de n'intervenir que sur le composant en panne lors d'une visite en atelier. Néanmoins, on peut encore supposer que ces actions ne sont pas forcément les actions optimales à effectuer dans ces modes, puisque ces derniers sont très rarement atteignables via cette politique préventive.

Finalement, bien que l'ensemble des politiques admissibles associé à ce problème d'optimisation semble n'avoir été que très partiellement exploré, l'explicitation de la politique fournit des recommandations interprétables et applicables industriellement, et on peut supposer que l'action préconisée pour les régimes rarement atteints n'a que peu d'impact sur son coût.

Détails des côuts. On détaille la décomposition du coût moyen de la politique π^*_{st6} dans la Table 4.25. Cette politique a réduit les coûts d'échec mais pas les coûts de maintenances par rapport à la politique non stationnaire π^*_{nst} basée sur les modes. Ceci s'explique par une plus importante part de maintenances préventives effectuées via la politique π^*_{st6}, qui différencie des régimes de dégradations des composants pour déclencher les retours en atelier.

TABLE 4.25 – Détails des coûts de la politique optimale non stationnaire basée sur les modes et une partition des temps de fonctionnement π^*_{st6}, sur un horizon de 52 semaines.

	π^*_{st6}	π^*_{nst}	π^*_{st}
pénalité échec	**8**	10	20
coût maintenance	189	**178**	190
coût moyen	197	**188**	210
IC 95%	[196 ; 198]	[187 ; 189]	[209 ; 211]

Statistiques de pannes. Le nombre moyen de pannes et le pourcentage de trajectoires sans panne pour la politique π^*_{st6} sont donnés Table 4.26. Si le taux de panne est légèrement supérieur à celui observé pour la politique π^*_{nst}, mais que la part de pénalité d'échec est moindre, la seule explication possible vient de maintenances faites à l'horizon, où π^*_{nst} conduit à payer la pénalité d'échec plutôt que les maintenances.

TABLE 4.26 – Nombre moyen de pannes par trajectoire et pourcentage de trajectoires sans panne, pour la politique π^*_{st6}, sur un horizon de 52 semaines.

	π^*_{st6}	π^*_{nst}	π^*_{st}
nombre de pannes	0.44	**0.32**	1.03
% trajectoires sans panne	62.93	**71.6**	26.59

Répartition des pannes. La répartition par mode de panne est détaillée Table 4.27. Les modes de panne les plus visités sont encore des pannes simples des composants, et cette répartition montre que les autres modes de pannes restent plus rarement observés.

TABLE 4.27 – Répartition des pannes de trajectoires contrôlées par la politique π^*_{st6}, en fonction du mode de panne, sur un horizon de 52 semaines.

Mode	2	5	6	7	11	14	15	16	18	19	21	22
π^*_{st6}	**16.8**	0.38	**25.88**	0.45	0.63	0.02	0.97	0.02	**52.87**	0.83	1.13	0.03
π^*_{nst}	**13.78**	0.86	**17.81**	1.11	2.48	0.13	4.10	0.14	**50.88**	3.92	4.59	0.20
π^*_{st}	**12.77**	0.29	**26.90**	0.41	0.43	0.01	0.93	0.02	**56.12**	0.85	1.27	0.02

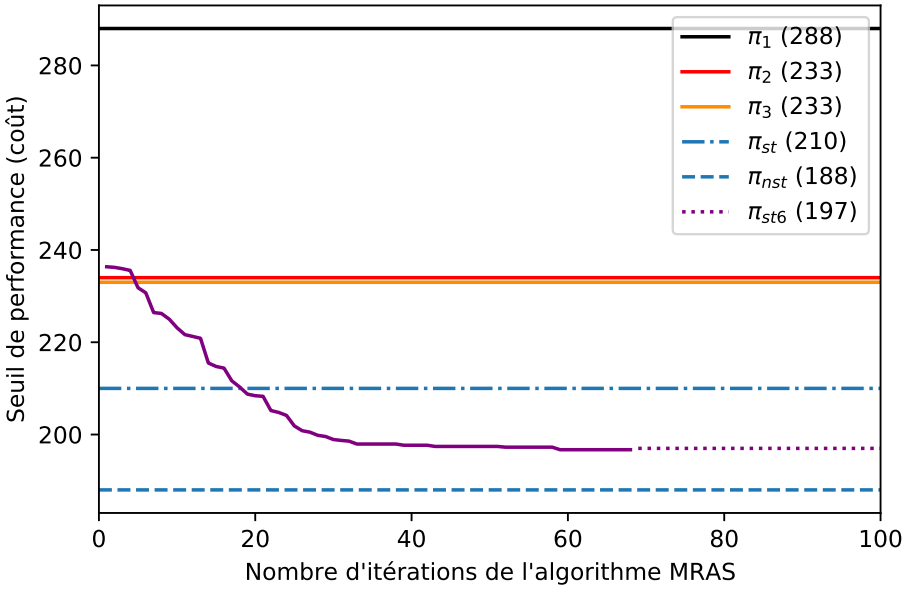


FIGURE 4.5 – Évolution du seuil de performance en fonction du nombre d'itérations de la méthode MRAS, dans le cas stationnaire basé sur les temps de fonctionnement.

Distribution de probabilités $\hat{P}_{st6}$. On s'intéresse maintenant à la distribution de probabilités $\hat{P}_{st6}$ obtenue à l'issue des itérations de la méthode MRAS. On note que pour certains groupes de régimes, la distribution est loin de la distribution uniforme P_0 donnée en entrée de l'algorithme : de nombreux coefficients sont réduits à une valeur proche de 0. La distribution étant sous la forme d'une matrice de dimensions $(\#\mathbb{X}_D, \#\mathbb{A}) = (513, 28)$, on reprend certains des sous-groupes de régimes distingués précedemment.

— si $r \in \mathcal{D}$ avec un seul régime dégradé : les probabilités se concentrent sur une seule action encore possible par mode, pour les 3 à 4 premiers régimes, et sur plusieurs actions possibles pour les régimes suivants, compte tenu de leur moindre observabilité dans les simulations via cette politique.

— si $r \in \mathcal{D}$ avec plusieurs régimes dégradés ou si $r \in \mathcal{P}$ avec un ou deux régimes dégradés, les probabilités sont aussi globalement réparties sur plusieurs actions, car ces régimes sont encore plus rarement atteignables dans les simulations via cette politique.

Évolution du seuil de performance. L'évolution du seuil de performance en fonction du nombre d'itérations de la méthode MRAS est représenté Figure 4.5. On note à nouveau que dès les premières itérations, la méthode a considéré des politiques de coût similaire à celui des meilleures politiques de références, mais plus coûteuses que le minimum trouvé dans le cas stationnaire basé sur les modes. On en déduit que tenir compte des durées de vie des composants pour conditionner les maintenances mène rapidement à réduire les coûts de la politique. Le seuil décroit ensuite lentement jusqu'à la dernière itération utile.

4.2.3.2 Résultats de l'algorithme MRAS pour le cas non stationnaire

On évalue à présent l'infuence de 4 hyperparamètres de l'algorithme MRAS sur sa dynamique, afin d'ajuster leurs valeurs, dans le cas non stationnaire basé sur les temps de fonctionnement. Ces expériences permettant a priori d'approcher au mieux la fonction valeur en explorant un espace de politiques plus grand, on choisit de les laisser tourner sur 100 itérations, même si la dernière itération utile intervenait avant.

Paramètre de tolérance ϵ. Les résultats numériques en fonction du paramètre ϵ sont donnés Table 4.28. Aussi, tenir compte des durées de vie pour les politiques produit au mieux seulement 1% de gain relatif par rapport à politique de référence π_4. Par contre, on a une perte de 4% par rapport au minimum trouvé sur les modes dans le cas non stationnaire, alors qu'approcher la fonction valeur sur un espace de politiques plus grand conduit a priori à de meilleures performances : ceci peut s'expliquer par le fait que l'espace des politiques à explorer est ici beaucoup plus grand. On garde $\epsilon = 0.5$.

TABLE 4.28 – Résultats en fonction de ϵ pour l'algorithme MRAS, avec $\nu = 0.1$, $\lambda = 0.05$, et $\rho_0 = 0.25$, dans le cas non stationnaire basé sur les temps de fonctionnement

ϵ	meilleur coût
0.1	197.52
0.5	**196.15**
1	207.36

Coefficient ν. Les résultats numériques en fonction de ν sont donnés Table 4.29. On observe ici des seuils qui sont du même ordre de grandeur que pour les tests effectués avec le paramètre ϵ, avec une légère amélioration globale des performances : le meilleur coût est plus faible que dans le cas précédent, mais toutefois encore moins bon que ceux obtenus lors des recherches basées sur les modes. Par contre, on ne retrouve pas la croissance observée jusqu'ici sur ce paramètre : on illustre un compromis menant à ne pas accorder de poids trop faible aux mises à jour. Ceci conduit encore à garder la valeur $\nu = 0.2$, pour toutes les expériences qui suivent, comme dans les cas précédents.

TABLE 4.29 – Résultats en fonction de ν pour l'algorithme MRAS, avec $\epsilon = 0.5$, $\lambda = 0.05$, et $\rho_0 = 0.25$, dans le cas non stationnaire basé sur les temps de fonctionnement.

ν	meilleur coût
0.1	196.15
0.2	**194.85**
0.3	198.80

Coefficient de mélange λ. Les résultats numériques en fonction de λ sont donnés dans la Table 4.30. On observe encore une légère amélioration. Ceci semble indiquer que le plan d'expérience permet effectivement de s'orienter vers de meilleures performances via l'ajustement des hyperparamètres. Par contre, on ne retrouve pas la croissance observée jusqu'ici sur ce paramètre : il reste utile de continuer à simuler une certaine proportion de politiques candidates simulées selon P_0, étant donné que l'espace à explorer est très grand. Finalement, le coût le plus faible est obtenu pour $\lambda = 0.2$.

TABLE 4.30 – Résultats en fonction de λ pour l'algorithme MRAS, avec $\epsilon = 0.5$, $\nu = 0.2$, et $\rho_0 = 0.25$, dans le cas non stationnaire basé sur les temps de fonctionnement.

λ	meilleur coût
0.05	194.85
0.10	196.90
0.20	**194.26**

Proportion initiale ρ_0. Les résultats numériques en fonction de ρ_0 sont donnés Table 4.31. À nouveau, les coûts sont du même ordre de grandeur, mais l'amélioration conduit finalement à trouver un coût de l'ordre de grandeur du minimum obtenu lors des recherches basées sur les modes, sans toutefois en trouver de meilleur. Ceci représente un gain de 4% par rapport à la politique π_4. Aussi, il semble à nouveau cohérent d'apprendre à partir d'un grand nombre de politiques candidates pour estimer la distribution de probabilités $\hat{P}$.

TABLE 4.31 – Résultats en fonction de ρ_0 pour l'algorithme MRAS, avec $\epsilon = 0.5$, $\nu = 0.2$, et $\lambda = 0.2$, dans le cas non stationnaire basé sur les temps de fonctionnement.

ρ	meilleur coût
0.25	194.26
0.50	190.66
0.75	**188.15**

Conclusions. Les différentes exécutions de l'algorithme MRAS sur ce plan d'expérience ont permis de construire une politique non stationnaire générant un gain relatif de près de 20% par rapport aux meilleures politiques de référence basées sur les modes, de 4% par rapport à la politique préventive π_4, et dont le coût est de l'ordre de grandeur du minimum trouvé dans le cas non stationnaire basé sur les modes. D'après ces résultats, les hyperparamètres de l'algorithme MRAS semblent avoir eu une influence sur les résultats numériques, et leur calibration permis d'améliorer les performances obtenues. Finalement, on s'intéresse aux interprétations que peuvent fournir ces résultats en terme d'application industrielle.

Interprétations des résultats. On détaille maintenant les différentes sorties numériques obtenues via l'exécution de la méthode MRAS. En particulier, on s'intéresse à la meilleure politique obtenue avec la distribution $\hat{P}_{nst6}$, à l'évolution du seuil de performance au cours des itérations de l'algorithme, ainsi qu'à la convergence de la distribution $\hat{P}_{nst6}$. Notre objectif est de tenter d'expliciter la meilleure politique obtenue, notée par la suite π^*_{nst6}, afin de l'interpréter et l'appliquer industriellement. Aussi, on présente à nouveau les statistiques telles que la répartition du coût moyen de la politique entre pénalités et opérations de maintenance, ainsi que le nombre et la répartition des pannes observées, et les comparons avec l'ensemble des résultats obtenus précedemment.

Politique non stationnaire optimale basée sur les modes et une partition des temps de fonctionnement des composants du pod. Le premier constat est que cette politique déterminée via l'algorithme MRAS est effectivement non stationnaire, en plus d'être différentiée selon le régime de dégradation des composants dégradés. En effet, le choix de l'action à effectuer dans un régime dépend de la date de décision, ce qui rend la politique π^*_{nst6} nettement moins facilement interprétable que la politique π^*_{st6}. On peut toutefois noter qu'on a encore remarqué certaines tendances.

Premièrement, on observe des effets de bord similaires à ceux explicités pour la politique π^*_{nst} : à l'horizon, et à l'approche de l'horizon, il y a un certain nombre de régimes pour lesquels la politique indique de ne pas entretenir ou remplacer les composants en cas de dégradation ou de panne. Ceci peut à nouveau s'expliquer par le fait qu'une visite en atelier coûte plus cher que la pénalité associée aux modes de panne : le coût d'une visite est au minimum de 60 alors que la pénalité n'est que 20.

Ensuite, la principale différence observée par rapport au cas stationnaire concerne les régimes de dégradations simples et multiples. Bien qu'on observe globalement le même comportement, c'est-à-dire d'entretenir tous les composants qui sont en régime dégradé dans ces cas, on note aussi une certaine propension à choisir parfois une action de maintenance qui ne ramène pas le système à l'état neuf, voir à ne pas revenir du tout en atelier, selon la date de décision. Néanmoins, aucune fréquence n'est particulièrement régulière ni commune aux régimes pour être facilement interprétable sur la politique étudiée.

Finalement, les dimensions $(\#\mathbb{X}_D,\ \#\mathbb{A},\ H) = (513,\ 28,\ 52)$ de cette politique non stationnaire font que le gain obtenu via son application n'est pas forcément plus explicitable que celui obtenu via la politique non stationnaire π^*_{nst} basée sur les modes du pod. On peut toutefois rappeler que, cette politique étant sous la forme d'un tableau qui associe à chaque régime et à chaque date de décision une action à effectuer sur le pod, elle peut être appliquée numériquement pour conduire à un coût de gestion du pod le plus faible possible, bien que la complexité du problème traité ne permette pas de l'expliciter plus en détails.

Détails des côuts. On détaille la décomposition du coût moyen de la politique π^*_{nst} dans la Table 4.32. La politique π^*_{nst6} réduit à la fois les coûts de maintenances et d'échec par rapport à la politique stationnaire optimale π^*_{st6}, et conduit finalement à des coûts très proches de ceux de la politique π^*_{nst}.

TABLE 4.32 – Détails des coûts de la politique optimale non stationnaire basée sur les modes et une partition des temps de fonctionnement du pod π^*_{nst6}.

	π^*_{nst6}	π^*_{st6}	π^*_{nst}	π^*_{st}
pénalité échec	10	**8**	10	20
coût maintenance	178	189	178	190
coût moyen	**188**	197	**188**	210
IC 95%	[187 ; 189]	[196 ; 198]	[187 ; 189]	[209 ; 211]

Statistiques de pannes. Le nombre moyen de pannes et le pourcentage de trajectoires sans panne observés sous la politique π^*_{nst6} sont donnés Table 4.33. On note que le nombre moyen de pannes et le pourcentage de trajectoires sans panne sont eux aussi très similaires à ceux observés pour des trajectoires contrôlées via la politique π^*_{nst}.

TABLE 4.33 – Nombre moyen de pannes par trajectoire et pourcentage de trajectoires sans panne, pour la politique π^*_{nst6}, sur un horizon de 52 semaines.

	π^*_{nst6}	π^*_{st6}	π^*_{nst}	π^*_{st}
nombre de pannes	0.36	0.44	**0.32**	1.03
% trajectoires sans panne	68.52	62.93	**71.6**	26.59

Répartition des pannes. La répartition par mode de panne est détaillée Table 4.34. Les modes de panne les plus visités sont encore des pannes simples des composants, et cette répartition montre que les autres modes de pannes restent plus rarement observés.

TABLE 4.34 – Répartition des pannes de trajectoires contrôlées par la politique π^*_{nst6}, en fonction du mode de panne, sur un horizon de 52 semaines.

Mode	2	5	6	7	11	14	15	16	18	19	21	22
π^*_{nst6}	**16.49**	0.71	**21.1**	0.85	1.32	0.08	2.71	0.13	**51.16**	2.04	3.30	0.10
π^*_{st6}	**16.80**	0.38	**25.88**	0.45	0.63	0.02	0.97	0.02	**52.87**	0.83	1.13	0.03
π^*_{nst}	**13.78**	0.86	**17.81**	1.11	2.48	0.13	4.10	0.14	**50.88**	3.92	4.59	0.20
π^*_{st}	**12.77**	0.29	**26.90**	0.41	0.43	0.01	0.93	0.02	**56.12**	0.85	1.27	0.02

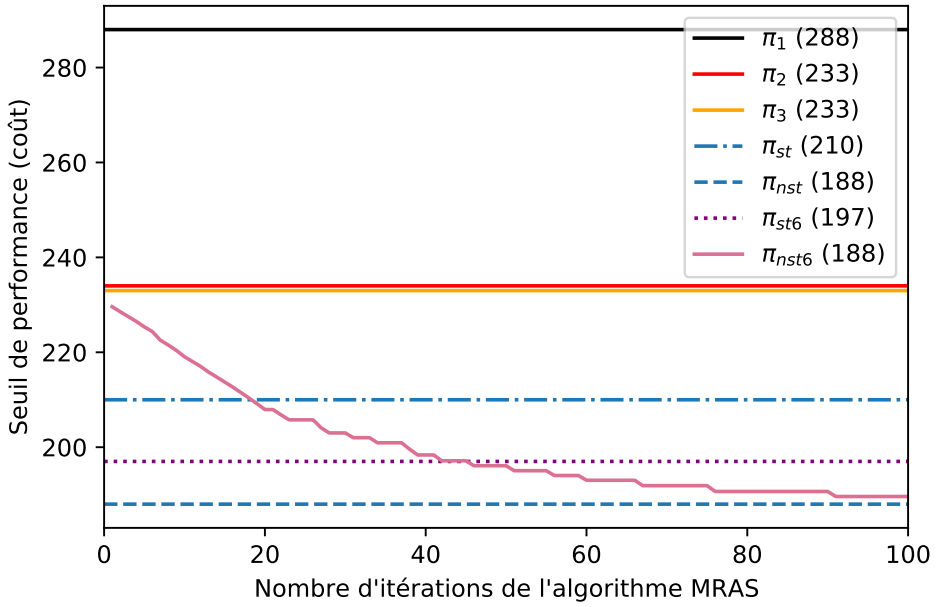


FIGURE 4.6 – Évolution du seuil de performance en fonction du nombre d'itérations de l'algorithme MRAS, dans le cas non stationnaire basé sur les temps de fonctionnement.

Distribution de probabilités $\hat{P}_{nst6}$. On s'intéresse ensuite à la distribution $\hat{P}_{nst6}$ obtenue via la méthode MRAS. On note que pour la plupart des dates de décisions, et pour les régimes qui semblent être suffisamment observés lors des simulations, la distribution terminale $\hat{P}_{nst6}$ s'éloigne de la distribution initiale P_0 : la plupart des coefficients sont réduits à une valeur proche de 0, de sorte que les probabilités se concentrent sur une seule action par régime. Sinon, pour les régimes qui semblent être très rarement observés via la politique optimale π^*_{nst6}, les probabilités sont encore réparties assez uniformément entre plusieurs actions possibles.

Évolution du seuil de performance. L'évolution du seuil de performance en fonction du nombre d'itérations de la méthode MRAS est représenté Figure 4.6. On note que dès les premières itérations, la méthode a considéré des politiques de l'ordre de grandeur des politiques de références. L'évolution montre que contrairement à ce qu'on a pu noter lors des recherches basées sur les modes, la décroissance du coût est moins rapide vers le minimum trouvé, et on observe plusieurs paliers pour lesquels le seuil de performances est constant. Enfin, la recherche s'arrête après 100 itérations, dont la dernière utile est la 92-ième. Par contre, rien ne garantit que le minimum trouvé soit la fonction valeur, et ce graphique semble indiquer que les recherches mériteraient d'être poursuivies sur davantage d'itérations puisque l'algorithme n'a pas été interrompu avant la 100ème itération. Les temps de calculs ne permettant pas de prolonger ces recherches dans le temps de cette thèse, relancer la méthode MRAS pour déterminer une politique encore moins coûteuse est une perspective à ce travail.

4.3 Conclusions

Pour la première partie de ce travail, on a cherché à approcher la fonction valeur du MDP introduit au Chapitre 3, sur l'ensemble des politiques dont les décisions se basent uniquement sur le mode du pod.

La mise en œuvre des méthodes MRAS et ASA a permis de montrer qu'on peut construire une politique stationnaire qui réduit à la fois le nombre de pannes et les coûts de gestion du système, par rapport aux politiques de références. Expliciter la stratégie stationnaire associée π^*_{st} a permis d'interpréter que le gain provient principalement du fait d'attrendre la survenue de plusieurs dégradations sur le système, qui restent imperceptibles par l'utilisateur, pour l'envoyer en atelier et mutualiser les opérations préventives. Cette stratégie stationnaire réalise un gain relatif de 66% (resp. 27% et 10%) par rapport aux politiques de référence π_0 (resp. π_1 et π_3), inclues dans l'ensemble des politiques stationnaires basées sur les modes. On montre enfin que cette politique passe à l'échelle pour un plus grand nombre de décisions.

Ensuite, le gain observé via π^*_{nst}, la meilleure politique non stationnaire basée sur les modes, provient du fait de différencier les actions selon la date de décision, et notamment de tenir compte de l'approche de l'horizon d'étude dans la stratégie de maintenance du pod. Cette stratégie non stationnaire réalise un gain relatif de 70% (resp. 35% et 19%) par rapport aux politiques π_0 (resp. π_1 et π_3), et de 10% par rapport à la politique π^*_{st}. Enfin, la politique π^*_{nst} réalise un gain relatif de 4% par rapport à la politique stationnaire π_4 qui tient compte des durées de vie des composants dégradés, et appartient donc à un espace de politiques beaucoup plus grand (et même infini) que celui exploré dans cette première partie.

Pour la seconde partie de ce travail, on a cherché à approcher la fonction valeur du MDP sur l'ensemble des politiques dont les décisions se basent sur le mode du pod et une discrétisation du temps de fonctionnement de ses composants.

L'application de la méthode MRAS a permis de construire une politique stationnaire qui réalise un gain relatif de 6% par rapport à la meilleure politique stationnaire basée sur les modes π^*_{st}, et dont le coût est similaire à celui de la politique π_4, qui n'appartient pas au sous-ensemble fini de politiques étudiées. Le gain obtenu provient essentiellement du fait de différencier les actions selon le niveau de dégradation des composants du pod. Ceci permet d'éviter la survenue de pannes en intervenant au juste moment sur le système, c'est à dire dès l'observation d'une dégradation avancée ou de dégradations multiples, afin d'envoyer le pod en atelier juste avant la panne ou de mutualiser les opérations de maintenance préventives.

Ensuite, l'amélioration apportée par la meilleure politique non stationnaire provient encore du fait de différencier les actions selon la date de décision. Son coût est égal à celui de la politique π^*_{nst}, et les statistiques de pannes des politiques sont semblables. Ceci suggère que les recherches méritent d'être poursuivies pour déterminer une politique non stationnaire moins coûteuse tenant compte des temps de fonctionnement du pod via une discrétisation.

Dans tous les cas étudiés, les algorithmes ont permis de déterminer des politiques dont les performances sont meilleures que celles des politiques de références stationnaires étudiées. Si la non stationnarité des meilleures politiques trouvées π^*_{nst} et π^*_{nst6} n'est pas facilement interprétable, il s'agit de stratégies déterministes markoviennes sous la forme d'un tableau qui donne la décision à prendre selon l'état du système et la date de décision, ce qui les rend utilisables industriellement. Enfin, ces stratégies sont en grande partie interprétables industriellement. Lorsque plusieurs composants se trouvent dans un mode dégradé, il est préférable de mutualiser les opérations préventives dans l'atelier, plutôt que de risquer que l'un d'entre eux tombe prochainement en panne, et entraîne des pénalités ainsi que des opérations de maintenances plus coûteuses. De la même façon, quand un composant se trouve dans un mode dégradé, tenir compte de son temps de fonctionnement permet d'intervenir au juste moment et au meilleur coût, pour garantir la réussite des missions tout en évitant les pannes.

Chapitre 5

Conclusions et perspectives

5.1 Conclusions

Optimiser une stratégie de maintenance prédictive pour des systèmes complexes à plusieurs états et plusieurs composants équipés de HUMS, développés et entretenus par Thales, est un des principaux objectifs de cette thèse. En particulier, il s'agit d'optimiser le rapport disponibilité/coût d'exploitation du système en déterminant une stratégie qui réalise un compromis entre intervenir trop tôt, ce qui conduit à payer des maintenances sur un système encore apte à assurer ses missions, et intervenir trop tard, menant à des pénalités d'indisponibilité et à des opérations de maintenances plus coûteuses.

Modélisation de la dynamique du pod par un MDP à espace d'états continu. Pour ce faire, on a développé un premier modèle stochastique de la dynamique du pod, via le formalisme d'un MDP à espace d'états continu ([CBdSD19]). Le problème d'optimisation associé est de déterminer, pour chaque date de décision et chaque état du système, l'action qui minimise la somme des coûts générés sur tout l'horizon d'étude. Pour ce modèle préliminaire, plusieurs politiques de référence stationnaires préventives et correctives basées sur le mode des composants du pod ont été proposées et comparées. Évaluer une politique tenant compte du temps passé en mode dégradé a permis d'observer que la politique ayant le plus faible coût est celle qui consiste à systématiquement entretenir un composant passant en mode dégradé, lorsque ce dernier caractérise un état où une panne est imminente et survient dès la première mission suivant ce changement d'état.

Suite à ces travaux, une importance particulière a été donnée à la calibration des paramètres de fiabilité et de coûts du MDP, dans le Chapitre 3, afin d'illustrer la problématique de Thales. Pour ce modèle, on a montré via la comparaison de plusieurs politiques de référence stationnaires préventives et correctives que tenir compte de la durée de vie passée dans un mode dégradé pour la prise de décision conduit à réduire à la fois le nombre moyen de pannes, et les coûts de maintenance et d'indisponibilité, en intervenant au bon moment sur le pod. Ainsi, les maintenances ne sont pas prévues selon une fréquence fixe, mais différentiées selon l'état du système et de ses composants. Ceci permet de ne pas envoyer le système en mission lorsque sa réussite ne peut pas être garantie, afin d'éviter la survenue d'une panne générant une indisponibilité opérationnelle et des coûts élevés.

Approximation numérique de la fonction valeur et de politiques quasi-optimales. Ensuite, deux méthodes d'optimisation stochastiques ont été mises en œuvre et comparées afin d'approcher la fonction valeur du MDP, et d'expliciter une politique associée qui soit applicable industriellement, dans le Chapitre 4. L'espace d'états du système étant infini et son noyau de transition n'étant pas explicite analytiquement, les méthodes classiques telles que la programmation dynamique ne peuvent pas être appliquées pour optimiser la stratégie de maintenance de ce système. Aussi, on a exploré différents sous-ensembles finis de l'ensemble des politiques admissibles associé à ce MDP, en utilisant des méthodes d'optimisation basées sur la simulation de trajectoires contrôlées du processus. La démarche proposée est alors d'associer une même action à des sous-ensembles d'états, puis d'effectuer la transition vers l'état suivant à partir du vrai état courant, ce qui permet de discretiser la règle de décision sans discretiser la dynamique du MDP, afin d'éviter le cumul d'erreurs de projections.

Premièrement, on a d'abord cherché à approcher la fonction valeur en basant les décisions uniquement sur le mode des composants du pod, sans considérer leurs durées de fonctionnement. L'ensemble des politiques associé contient les politiques de références π_0 à π_3, et son exploration montre qu'il existe des politiques stationnaires basées sur les modes qui réduisent à la fois le nombre de pannes et les coûts de gestion du système, par rapport à ces politiques. Les expliciter a permis d'interpréter que le gain provient de la mutualisation des opérations préventives, lorsque plusieurs composants sont dans un mode dégradé. En effet, les dégradations restent imperceptibles par l'utilisateur et mutualiser les opérations permet d'intervenir avant la panne pour garantir la réussite des missions tout en minimisant les coûts.

Deuxièmement, on a exploré un espace de politiques plus grand, contenant les politiques π_0 à π_3 mais pas π_4, en basant les décisions sur une discrétisation des durées de fonctionnement des composants en mode dégradé. Sur cet espace de politiques fini, le fait d'associer une même action à différents états qui ont le même mode et dont les temps de fonctionnement ont le même plus proche voisin, permet finalement d'explorer encore un sous-espace fini des politiques admissibles. Expliciter la meilleure stratégie trouvée sur cet espace a permis d'in-

terpréter que le gain est obtenu en différentiant les actions selon le niveau de dégradation d'un composant, ce qui permet d'éviter des pannes en attendant la survenue de plusieurs dégradations sur le système pour l'envoyer en atelier et mutualiser les opérations préventives.

Dans les deux cas, on s'est d'abord restreint à étudier les politiques stationnaires, qui fournissent des stratégies plus facilement interprétables, puis à considérer des politiques non stationnaires, qui réduisent encore les coûts d'exploitation et complètent les interprétations, en différentiant les actions selon la date de décision et notamment en tenant compte de l'horizon d'étude du système pour adapter la stratégie de maintenance. Bien que ces politiques ne soient pas toujours facilement interprétables, il s'agit en fin de compte de stratégies déterministes markoviennes sous la forme d'un tableau qui donne la décision à prendre selon chaque état et chaque date de décision, ce qui les rend utilisables industriellement.

5.2 Perspectives

Enfin, on évoque différentes perspectives, à court et à long terme, possibles pour ce travail.

5.2.1 Perspectives à court terme

Poursuite de l'optimisation de politiques basées sur les temps de fonctionnement. Une suite directe à ces travaux serait de poursuivre les recherches numériques pour approcher la fonction valeur du MDP, toujours en tenant compte des temps de fonctionnement des composants du système. Bien que les premiers résultats soient interprétables, applicables industriellement et permettent de réduire les coûts et le taux de pannes du pod, la méthode MRAS est une méthode stochastique asymptotique qui dépend de nombreux paramètres à déterminer. Aussi, rien ne garantit que ces résultats ne puissent pas être améliorés, ils sont seulement une approximation de la fonction valeur et d'une stratégie optimale associée, qui sont définis sur un espace de politiques infini. Finalement, ce manuscrit a été rédigé de sorte que les travaux de modélisation, de simulation et d'optimisation de la gestion de ce type de systèmes industriels par des MDP puissent être repris et poursuivis.

Relâchement des hypothèses simplificatrices du modèle. Un second type de perspectives est d'enrichir le modèle pour approcher au mieux la réalité industrielle. En effet, plusieurs hypothèses simplificatrices ont été faites pour ce modèle : la durée passée dans l'atelier est la même peu importe le type et le nombre d'opérations à effectuer et les coûts de maintenances sont différenciés selon l'opération mais pas selon le composant traité. Aussi, les travaux de [YWC+17] ont considéré, par exemple, des durées d'intervention variables et dépendantes de la machine à traiter. Ainsi, bien que notre modèle MDP soit très sensible à la définition de ces paramètres, changer leurs valeurs ne change pas la structure des espaces d'états et d'actions, ni du simulateur présenté dans ce travail.

Ensuite, on a considéré des maintenances parfaites, qui ramènent le système à l'état neuf. Une perspective serait de considérer que les opérations préventives peuvent être imparfaites, les opérations correctives étant supposées être des remplacements par un composant neuf. Pour cela, plusieurs hypothèses classiquement étudiées semblent pouvoir s'intégrer au modèle.

Une adaptation basée sur le modèle de Brown-Proschan [BP83] nécessiterait, par exemple, de modifier le noyau de transition du MDP lorsque l'action choisie sur un composant est une opération préventive. Ainsi, avec probabilité $p \in [0, 1]$, il s'agit d'une maintenance parfaite (AGAN), comme présenté dans ce travail. Ensuite, avec probabilité $1 - p$, il s'agit d'une maintenance minimale (ABAO), c'est à dire que le composant passe en mode dégradé avec un temps de fonctionnement identique à celui précédant la défaillance, et sa prochaine date de panne est déterminée selon un tirage conditionnel à ce temps.

Les modèles plus généraux d'âge virtuel [Kij89, DG04, MC19] supposent que l'efficacité d'une maintenance est proportionnelle à une valeur $p \in [0, 1]$ qui dépend de la durée écoulée depuis la dernière maintenance. En dehors des valeurs $p = 0$ et $p = 1$ qui correspondent aux hypothèses AGAN et ABAO, il faut définir si le composant est en mode stable ou dégradé suite à une opération, et selon quelle intensité déterminer son prochain changement de mode.

Ces modifications ne changeant pas la structure de l'espace d'états et d'actions, ni celle du simulateur, la procédure d'optimisation basée sur les simulations pourrait encore s'appliquer.

5.2.2 Perspectives à long terme

Changements d'échelle pour le nombre de composants et de dates de décisions. Enfin, plusieurs changements d'échelle peuvent être envisagés. Les premiers concernent le nombre de composants du système physique et le nombre de dates de décisions du modèle. Nos premiers résultats de simulations montrent qu'une politique stationnaire optimale basée sur les modes du système peut passer à l'échelle pour un plus grand nombre de décisions. Ensuite, bien que les résultats numériques aient été donnés pour un système à 3 composants, le modèle ainsi que le simulateur s'étendent à plus de composants. En effet, le modèle numérique est basé principalement sur une numérotation lexicographique des modes (resp. actions) et la correspondance entre le numéro d'un mode (resp. action) et sa décomposition composant par composant. Ceci permet d'automatiser l'encodage des modes (resp. actions) et la description du noyau de transition. Ainsi, le nombre de composants du système devient un paramètre du modèle numérique.

Aussi, pour optimiser une politique non stationnaire sur un plus grand nombre de composants et de dates de décision, il faudrait privilégier la recherche de politiques basées sur les modes. En effet, compte tenu de la combinatoire du nombre de politiques candidates à évaluer après leur discrétisation, considérer les temps de fonctionnement pourrait mener à des temps de calculs prohibitifs, alors que rechercher une politique restreinte au mode peut

déjà permettre de réduire les coûts, en un temps de calcul raisonnable. Dans tous les cas, le principal avantage de notre démarche est qu'elle est constructive. Ainsi, le calcul d'une politique optimale n'est à faire qu'une seule fois, et cette politique étant sous la forme d'un tableau qui donne la décision à prendre pour chaque état du système et chaque date de décision, elle est utilisable industriellement.

Politique d'emploi et de maintenance de flotte d'équipements multicomposants. Finalement, Thales souhaiterait optimiser la gestion globale d'une flotte d'équipements à plusieurs composants équipés de HUMS, plutôt qu'un équipement unique. Aussi, à l'issue des travaux de [Bay13, Gee17] sur l'optimisation d'instants puis de décisions de maintenance pour des caméras thermiques, Thales a souhaité changer d'équipement et d'échelle dans la problématique d'optimisation de la maintenance de ses équipements. En effet, les méthodes développées précédemment et basées sur des modèles de processus markovien déterministes par morceaux (PDMP) souffrent du fléau de la dimension et ne permettent pas de passer à l'échelle lorsque le nombre de décisions, le nombre de cycles de vie et de composants ou encore le nombre d'équipements augmentent. L'utilisation d'autres modèles mathématiques s'impose. Le choix des MDP a été fait pour permettre le passage à l'échelle de la flotte, et semble pouvoir s'y prêter pour déterminer des stratégies optimisées qui restent interprétables et applicables.

La principale différence avec le modèle développé dans cette thèse serait l'ajout d'actions liées au choix du ou des systèmes à envoyer en mission et une notion de disponibilité globale de la flotte à prendre en compte. Un autre aspect d'intéret serait de pouvoir considérer des missions de durées et de niveaux de priorités différents. L'objectif est alors de construire une politique de gestion des ressources qui garantisse la disponibilité et l'emploi optimisé des équipements pour les missions, tout en minimisant les coûts de maintenance de la flotte.

Dans le cas d'une flotte d'équipements, les travaux de la littérature traitent certains aspects spécifiques : la maintenance d'équipements à un seul [DMLL17, DML+14] ou plusieurs composants [SBJ11, BYS19], l'affectations de ressources [LZXZ18, DC87, OCO17], ou encore le dimensionnement de la flotte [WHW05, FK18] permettant d'assurer la disponibilité. Les travaux de [Rob19] traitent l'optimisation conjointe des maintenances et des missions d'une flotte, tenant compte de contraintes opérationnelles, mais pour des systèmes monolhitiques. L'objectif à long terme de notre travail est d'optimiser conjointement l'emploi et la maintenance d'équipements à plusieurs composants, en déterminant une politique optimale différenciée selon le type de mission, la date de décision, et selon l'état des équipements et de leurs composants, pour garantir leur disponibilité tout en minimisant les coûts de gestion de la flotte.

Annexe A

Politiques de référence et politiques quasi-optimales

Dans ce chapitre, on donne maintenance le détail des différentes politiques optimales stationnaires et non stationnaires déterminées par l'application des méthodes MRAS et ASA, lorsque les décisions sont basées uniquement sur les modes des composants du pod.

Aussi, les politiques déterminées à l'aide des méthodes MRAS et ASA lorsqu'on tient également compte des temps de fonctionnement des composants du système via une discrétisation, ne sont pas données ici, compte tenu de leurs dimensions.

A.1 Politiques stationnaires

Concernant le cas des politiques stationnaires, les politiques sont sous la forme d'un vecteur qui indique l'action à effectuer selon le mode des composants du système. Cette action est différenciée selon l'état du pod mais est la même pour toutes les dates de décisions. Les politiques de références décrites section 3.3, ainsi que les politiques quasi-optimales déterminées via les méthodes MRAS et ASA sous-section 4.1.3, sont détaillées dans la Table A.1.

A.2 Politiques non stationnaires

Dans le cas des politiques non stationnaires, il s'agit d'une matrice qui précise l'action à effectuer, différenciées pour chaque date de décision, et en fonction de chaque mode des composants du système. La politique déterminée via la méthode MRAS (resp. ASA) dans la section 4.1.3.4 (resp. 4.1.3.5) est détaillée dans la Table A.2 (resp. la Table A.3).

TABLE A.1 – Détails des actions à effectuer selon le mode du système, pour les politiques de référence stationnaires π_0 à π_3, et pour les politiques stationnaires optimales basées sur les modes du pod, déterminée via les méthodes MRAS (π_{st}) et ASA ($\pi_{st'}$). Les 28 colonnes représentent les différents modes considérés pour le pod, dont celles grisées sont les modes inatteignables du système.

Politique	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
π_0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	17	0	0	20	0	0	23	24	25	26	27
π_1	0	0	2	0	0	2	6	6	8	0	0	2	0	0	2	6	6	17	18	18	20	18	18	23	24	25	26	27
π_2	0	0	2	0	0	5	6	7	8	0	0	11	0	0	14	15	16	17	18	19	20	21	22	23	24	25	26	27
π_3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
π_{st}	0	0	2	0	4	5	6	6	8	0	10	2	12	13	2	15	16	17	18	19	20	21	22	23	24	25	26	27
$\pi_{st'}$	0	0	2	0	4	2	6	7	8	0	10	11	12	13	5	15	0	17	18	1	20	21	21	23	24	25	26	27

TABLE A.2 – Détails des actions à effectuer selon le mode du système et selon la date de décision, pour la politique non stationnaire optimale basée sur les modes du pod, déterminée via la méthode MRAS. Les 52 lignes représentent les 52 dates de décisions, et les 28 colonnes représentent les différents modes considérés pour le pod, dont celles grisées sont les modes inatteignables.

0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	17	0	0	20	0	0	23	24	25	26	27
0	0	2	0	3	5	0	7	8	0	10	0	9	10	0	6	7	17	18	18	20	18	21	23	24	25	26	27
0	0	2	3	1	5	6	0	8	0	1	0	3	13	2	6	16	17	0	19	20	0	0	23	24	25	26	27
0	0	0	0	1	2	0	6	8	9	10	11	12	4	5	15	0	17	18	0	20	21	22	23	24	25	26	27
0	0	0	0	0	2	6	7	8	0	0	0	12	0	2	0	7	17	18	18	20	21	22	23	24	25	26	27
0	0	0	0	0	0	6	6	8	0	0	2	12	13	0	6	15	17	0	19	20	18	19	23	24	25	26	27
0	1	2	0	4	5	0	0	8	0	10	0	0	12	0	6	0	17	0	19	20	18	21	23	24	25	26	27
0	1	0	3	0	5	6	7	8	0	10	2	12	4	5	15	7	17	18	0	20	18	19	23	24	25	26	27
0	0	2	3	1	0	0	7	8	9	0	11	12	10	11	6	15	17	18	0	20	21	21	23	24	25	26	27
0	0	0	0	0	0	0	0	8	0	0	2	12	0	14	0	0	17	0	19	20	0	0	23	24	25	26	27
0	0	0	0	4	2	6	6	8	0	10	0	0	0	2	6	6	17	0	0	20	21	22	23	24	25	26	27
0	1	2	0	0	2	0	7	8	0	10	2	12	3	2	6	7	17	18	19	20	21	19	23	24	25	26	27
0	1	2	3	1	0	0	0	8	0	1	11	0	13	14	6	6	17	18	19	20	0	21	23	24	25	26	27
0	0	2	0	1	5	6	0	8	0	10	2	12	4	2	15	16	17	18	18	20	18	21	23	24	25	26	27
0	0	0	0	3	2	6	7	8	0	10	2	12	3	14	0	15	17	18	0	20	18	22	23	24	25	26	27
0	0	2	0	3	0	6	0	8	0	10	2	12	9	11	6	15	17	18	0	20	0	19	23	24	25	26	27
0	0	2	0	0	5	6	0	8	9	0	2	12	3	5	0	6	17	18	0	20	21	18	23	24	25	26	27
0	0	2	3	4	5	0	6	8	0	10	0	12	10	0	0	16	17	18	18	20	18	18	23	24	25	26	27
0	0	0	0	1	0	6	7	8	0	10	11	12	13	0	15	7	17	0	0	20	21	18	23	24	25	26	27
0	0	0	0	4	2	6	6	8	0	10	11	12	12	2	0	7	17	18	18	20	18	0	23	24	25	26	27
0	0	2	0	4	2	0	0	8	0	10	2	12	13	5	0	16	17	18	19	20	0	22	23	24	25	26	27
0	0	0	0	4	0	6	7	8	0	0	11	12	10	11	6	15	17	18	19	20	0	22	23	24	25	26	27
0	1	2	0	0	2	6	7	8	0	10	0	12	13	11	6	15	17	0	19	20	18	22	23	24	25	26	27
0	0	2	0	4	0	6	6	8	0	0	0	12	12	5	0	15	17	18	19	20	21	19	23	24	25	26	27
0	1	2	3	1	2	6	7	8	0	0	11	12	13	5	15	7	17	18	19	20	0	18	23	24	25	26	27
0	0	0	3	3	5	6	7	8	9	10	2	12	12	14	15	0	17	18	0	20	21	0	23	24	25	26	27
0	0	2	0	3	5	0	0	8	0	10	0	12	4	5	6	16	17	0	19	20	21	0	23	24	25	26	27
0	0	2	0	0	5	6	7	8	0	9	2	12	12	2	6	0	17	18	19	20	0	22	23	24	25	26	27
0	1	0	0	4	5	0	7	8	0	10	0	0	12	11	0	15	17	18	19	20	18	0	23	24	25	26	27
0	0	0	3	3	5	0	0	8	0	10	0	12	1	0	6	6	17	0	19	20	18	19	23	24	25	26	27
0	0	2	3	4	0	0	6	8	0	0	0	12	10	5	15	0	17	18	18	20	0	22	23	24	25	26	27
0	0	0	0	1	0	0	6	8	0	10	0	0	0	11	6	15	17	0	0	20	0	18	23	24	25	26	27
0	0	2	0	3	5	0	0	8	0	10	2	12	13	0	15	7	17	18	18	20	21	19	23	24	25	26	27
0	0	0	0	0	2	6	7	8	0	10	0	12	4	2	6	0	17	0	0	20	18	21	23	24	25	26	27
0	0	2	0	4	5	6	7	8	9	0	0	12	0	2	15	6	17	18	19	20	0	0	23	24	25	26	27
0	0	2	3	4	0	0	0	8	0	10	0	12	13	2	15	0	17	0	19	20	21	22	23	24	25	26	27
0	0	2	0	0	5	6	7	8	0	10	11	12	13	5	15	16	17	0	19	20	21	0	23	24	25	26	27
0	1	2	0	4	5	0	0	8	0	10	2	12	10	11	15	16	17	0	19	20	18	19	23	24	25	26	27
0	0	2	3	0	0	6	6	8	0	0	2	0	1	2	6	6	17	18	0	20	0	19	23	24	25	26	27
0	0	2	3	4	2	6	0	8	9	10	2	12	0	0	15	7	17	18	19	20	18	19	23	24	25	26	27
0	0	0	3	0	5	0	7	8	0	0	11	12	1	11	0	6	17	18	18	20	21	19	23	24	25	26	27
0	0	2	0	4	2	0	6	8	0	10	11	9	10	2	6	16	17	18	18	20	0	0	23	24	25	26	27
0	0	2	0	4	2	6	0	8	0	1	11	0	1	14	6	6	17	0	0	20	0	18	23	24	25	26	27
0	0	2	0	0	0	6	6	8	0	1	2	12	4	14	6	7	17	18	19	20	0	22	23	24	25	26	27
0	0	2	0	1	2	6	6	8	0	9	2	12	13	14	0	6	17	18	18	20	18	0	23	24	25	26	27
0	0	2	0	3	5	6	7	8	0	0	0	9	0	0	0	7	17	18	19	20	18	21	23	24	25	26	27
0	0	2	0	0	5	0	0	8	0	1	2	3	1	0	0	15	17	18	0	20	21	0	23	24	25	26	27
0	0	2	0	0	2	0	6	8	0	10	11	3	9	14	15	0	17	0	0	20	0	0	23	24	25	26	27
0	0	0	0	0	5	0	6	8	0	0	11	0	0	5	6	0	17	0	18	20	21	0	23	24	25	26	27
0	0	0	0	0	0	0	0	8	0	0	0	0	1	11	6	6	17	0	19	20	0	19	23	24	25	26	27
0	0	0	0	0	0	0	0	8	0	0	0	0	1	14	15	0	17	0	18	20	0	19	23	24	25	26	27
0	0	2	3	4	0	6	6	8	9	9	2	3	13	0	0	16	17	0	19	20	18	19	23	24	25	26	27

TABLE A.3 – Détails des actions à effectuer selon le mode du système et selon la date de décision, pour la politique non stationnaire optimale basée sur les modes du pod, déterminée via la méthode ASA. Les 52 lignes représentent les 52 dates de décisions, et les 28 colonnes représentent les différents modes considérés pour le pod, dont celles grisées sont les modes inatteignables.

0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	17	0	0	20	0	0	23	24	25	26	27
0	0	2	0	1	0	0	0	8	0	10	2	12	13	11	15	0	17	18	0	20	18	21	23	24	25	26	27
0	0	2	3	3	2	0	7	8	0	1	11	0	9	5	15	15	17	0	19	20	0	19	23	24	25	26	27
0	1	0	3	4	0	6	7	8	0	1	11	12	1	14	15	15	17	18	18	20	21	19	23	24	25	26	27
0	1	0	0	0	0	0	7	8	0	9	11	12	9	5	0	0	17	18	0	20	0	19	23	24	25	26	27
0	1	0	0	4	5	6	6	8	9	9	0	0	1	5	6	15	17	0	0	20	21	19	23	24	25	26	27
0	0	2	3	4	2	0	7	8	0	1	2	0	9	2	6	15	17	18	19	20	0	0	23	24	25	26	27
0	0	2	0	4	2	6	0	8	0	10	11	12	9	5	15	15	17	0	19	20	18	19	23	24	25	26	27
0	0	0	3	4	2	6	0	8	0	10	2	12	10	14	15	0	17	18	18	20	18	22	23	24	25	26	27
0	0	0	0	1	2	6	6	8	0	10	2	9	12	0	15	16	17	18	18	20	18	21	23	24	25	26	27
0	0	2	3	1	2	0	7	8	0	9	0	0	13	5	0	15	17	18	0	20	18	0	23	24	25	26	27
0	0	0	3	4	5	0	0	8	0	10	11	12	1	0	0	7	17	18	0	20	21	21	23	24	25	26	27
0	0	2	0	4	5	6	6	8	0	10	11	12	13	5	0	0	17	18	0	20	18	22	23	24	25	26	27
0	1	0	0	4	2	6	6	8	0	0	11	12	4	2	0	6	17	18	19	20	21	0	23	24	25	26	27
0	0	0	0	1	5	6	0	8	0	0	0	12	12	0	0	0	17	18	19	20	0	22	23	24	25	26	27
0	1	2	3	4	0	6	0	8	0	10	2	0	9	2	6	15	17	18	19	20	0	21	23	24	25	26	27
0	1	0	3	1	2	0	7	8	0	0	11	12	13	14	15	7	17	18	19	20	0	0	23	24	25	26	27
0	0	0	0	4	2	6	0	8	0	10	2	9	1	14	6	0	17	18	18	20	21	0	23	24	25	26	27
0	0	0	0	1	5	0	7	8	0	10	11	12	3	5	0	15	17	0	19	20	18	21	23	24	25	26	27
0	0	0	0	0	0	0	0	8	0	9	2	12	12	11	0	7	17	18	18	20	0	0	23	24	25	26	27
0	0	2	3	0	5	0	7	8	9	0	0	12	4	5	0	7	17	18	19	20	0	21	23	24	25	26	27
0	0	2	0	4	2	6	0	8	0	10	11	12	13	0	0	7	17	18	0	20	21	0	23	24	25	26	27
0	0	0	3	4	5	6	7	8	0	0	2	12	13	5	6	16	17	18	18	20	0	19	23	24	25	26	27
0	1	0	3	1	2	0	7	8	0	0	11	0	3	2	6	6	17	0	19	20	21	22	23	24	25	26	27
0	0	2	3	1	5	6	0	8	0	0	0	0	9	14	15	15	17	18	18	20	18	22	23	24	25	26	27
0	1	2	3	1	2	0	0	8	0	10	0	12	13	0	6	16	17	18	18	20	18	0	23	24	25	26	27
0	0	0	0	1	0	0	6	8	9	0	2	12	13	0	0	16	17	18	0	20	0	0	23	24	25	26	27
0	0	2	3	1	5	6	6	8	0	9	2	9	10	2	0	6	17	18	0	20	0	19	23	24	25	26	27
0	0	0	3	0	0	0	0	8	9	9	0	0	10	0	0	7	17	18	19	20	21	0	23	24	25	26	27
0	1	2	0	1	2	6	6	8	0	10	2	12	12	2	15	6	17	0	19	20	21	18	23	24	25	26	27
0	0	2	0	0	2	0	0	8	0	0	0	0	3	5	15	15	17	18	18	20	21	0	23	24	25	26	27
0	0	2	0	4	2	0	7	8	0	0	0	12	4	11	15	15	17	18	19	20	0	21	23	24	25	26	27
0	0	2	0	3	0	6	0	8	0	9	0	12	1	11	0	0	17	0	0	20	18	19	23	24	25	26	27
0	0	2	3	1	2	6	0	8	9	10	0	12	3	5	15	0	17	18	0	20	18	22	23	24	25	26	27
0	0	0	0	0	5	6	0	8	0	0	0	3	1	2	0	16	17	18	19	20	21	22	23	24	25	26	27
0	0	2	0	4	5	6	7	8	0	10	11	12	3	11	15	15	17	0	0	20	18	18	23	24	25	26	27
0	0	0	0	4	5	6	6	8	0	10	11	12	12	5	0	6	17	0	18	20	18	0	23	24	25	26	27
0	0	2	3	3	5	6	0	8	0	0	11	0	1	0	6	0	17	18	18	20	21	21	23	24	25	26	27
0	0	0	3	0	2	0	0	8	0	0	0	12	4	14	6	15	17	0	18	20	0	21	23	24	25	26	27
0	1	2	3	3	2	0	6	8	9	10	11	12	13	5	0	16	17	18	19	20	21	21	23	24	25	26	27
0	1	0	3	4	0	0	0	8	0	0	11	0	0	14	6	16	17	18	0	20	18	0	23	24	25	26	27
0	1	0	3	4	0	0	0	8	0	10	0	0	12	11	0	15	17	0	0	20	18	22	23	24	25	26	27
0	0	2	3	3	5	0	6	8	0	10	0	0	13	2	0	15	17	0	18	20	0	21	23	24	25	26	27
0	0	2	0	4	0	6	0	8	0	0	2	3	3	14	15	7	17	0	19	20	21	0	23	24	25	26	27
0	0	2	0	4	0	6	6	8	0	0	11	9	3	2	15	6	17	0	0	20	18	0	23	24	25	26	27
0	0	0	0	1	2	0	6	8	0	1	0	0	10	0	15	0	17	18	0	20	21	0	23	24	25	26	27
0	0	2	0	1	5	6	7	8	0	0	11	3	1	5	0	16	17	18	0	20	21	22	23	24	25	26	27
0	0	2	0	1	0	0	0	8	0	9	2	9	3	14	15	15	17	18	19	20	21	19	23	24	25	26	27
0	0	0	0	0	2	6	7	8	0	0	11	3	4	11	0	16	17	0	18	20	21	0	23	24	25	26	27
0	0	0	0	0	0	0	7	8	0	0	2	9	13	11	6	7	17	0	18	20	0	19	23	24	25	26	27
0	0	0	0	0	0	0	6	8	0	0	0	0	9	2	15	7	17	0	0	20	21	0	23	24	25	26	27
0	1	2	3	0	5	0	7	8	9	9	11	3	9	0	15	15	17	18	0	20	21	21	23	24	25	26	27

Bibliographie

[ADPR16] J. Anselmi, F. Dufour, and T. Prieto-Rumeau. Computable approximations for continuous-time markov decision processes on borel spaces based on empirical measures. Journal of Mathematical Analysis and Applications, 443(2) :1323–1361, 2016.

[AKOTY11] D. Ait-Kadi, M. Ouali, L. Tadj, and S. Yacout. A survey of replacement models with minimal repair. London : Springer Series in Reliability Engineering., 2011.

[ARC18] N. Aslam, M. Roy, and C. Chowdhury. Designing transmission strategies for enhancing communications in medical iot using markov decision process. Sensors (Basel, Switzerland), 18, 2018.

[AX17] S. Alaswad and Y. Xiang. A review on condition-based maintenance optimization models for stochastically deteriorating system. Reliability Engineering and System Safety, 157 :54–63, 2017.

[BABC11] K. Bouvard, S. Artus, C. Bérenguer, and V. Cocquempot. Condition-based dynamic maintenance operations planning & grouping. application to commercial heavy vehicles. Reliability Engineering and System Safety, 96 :601–610, 2011.

[Bay13] Camille Baysse. Analyse et optimisation de la fiabilité d'un équipement opto-électrique équipé de HUMS. Theses, Université de Bordeaux, France, 2013.

[BBG06] A. Barros, C. Bérenguer, and A. Grall. A maintenance policy for two-unit parallel systems based on imperfect monitoring information. Reliability Engineering and System Safety, 91 :131–136, 2006.

[Bel57a] R. Bellman. Dynamic Programming. Princeton, 1957.

[Bel57b] R. Bellman. A markovian decision process. Journal of mathematics and mechanics, 6(5) :679–684, 1957.

[Ber87] D. Bertsekas. Dynamic Programming : Deterministic and Stochastic Models. Prentice Hall, 1987.

[BGB03] A. Barros, A. Grall, and C. Bérenguer. A maintenance policy optimized with imperfect and/or partial monitoring. Annual Reliability and Maintainability Symposium, 2003., pages 406–411, 2003.

[BH61] R. Bellman and R. Howard. Dynamic programming and markov processes. Mathematics of Computation, 15 :217, 1961.

[BP83] M. Brown and F. Proschan. Imperfect repair. Journal of Applied Probability, 20(4) :851–859, 1983.

[BP03] V. Bally and G. Pagès. A quantization algorithm for solving multi-dimensional discrete-time optimal stopping problems. Bernoulli, 9(6) :1003–1049, 2003.

[BR11] N. Bauerle and U. Rieder. Markov Decision Processes with Applications to Finance. Universitext. Springer, Heidelberg., 2011.

[BRASK17] E. I. Basri, I. H. A. Razak, H. Ab-Samat, and S. Kamaruddin. Preventive maintenance (pm) planning : a review. Journal of Quality in Maintenance Engineering, 23 :114–143, 2017.

[BYS19] S. Barde, S. Yacout, and H. Shin. Optimal preventive maintenance policy based on reinforcement learning of a fleet of military trucks. Journal of Intelligent Manufacturing, 30 :147–161, 2019.

[CBdSD19] T. Cherchi, C. Baysse, B. de Saporta, and F. Dufour. Optimal predictive maintenance policy for multi-component systems. In ESREL 2019 - 29th European Safety and Reliability Conference, Hanovre, Germany, September 2019.

[CC18] A. Cabarbaye and A. Cabarbaye. Mise en œuvre des hums sur les systèmes embarques - application aux drones. In Congrès Lambda Mu 21 “ Maîtrise des risques et transformation numérique : opportunités et menaces ”, Reims, France, October 2018.

[Cer85] V. Cerný. Thermodynamical approach to the traveling salesman problem : An efficient simulation algorithm. Journal of Optimization Theory and Applications, 45 :41–51, 1985.

[CFHM07] H. Chang, M. Fu, J. Hu, and S. Marcus. An asymptotically efficient simulation-based algorithm for finite horizon stochastic dynamic programming. IEEE Transactions on Automatic Control, 52 :89–94, 2007.

[CFHM13] H. Chang, M. Fu, J. Hu, and S. Marcus. Simulation-Based Algorithms for Markov Decision Processes. Springer, 2013.

[CMBZ18] Michele Compare, Paolo Marelli, Piero Baraldi, and Enrico Zio. A markov decision process framework for optimal operation of monitored multi-state systems. Proceedings of the Institution of Mechanical Engineers, Part O : Journal of Risk and Reliability, 232 :1748006X1875707, 02 2018.

[CP91] D. I. Cho and M. Parlar. A survey of maintenance models for multi-unit systems. European Journal of Operational Research, 51 :1–23, 1991.

[CT97] Christiane Cocozza-Thivent. Processus stochastiques et fiabilité des systèmes, volume 28. Springer Science & Business Media, 1997.

[CTKL13] C. Ponzoni Carvalho Chanel, F. Teichteil-Königsbuch, and C. Lesire. Multi-target detection and recognition by uavs using online pomdps. In AAAI, 2013.

[DC87] P. J. Dejax and T. G. Crainic. Survey paper—a review of empty flows and fleet management models in freight transportation. Transportation Science, 21(4) :227–248, November 1987.

[Den70] E. V. Denardo. On linear programming in a markov decision problem. Management Science, 16 :281–288, 1970.

[Den03] E. V. Denardo. Dynamic programming : Models and applications. The Computer Journal, 2003.

[DG04] L. Doyen and O. Gaudoin. Classes of imperfect repair models based on reduction of failure intensity or virtual age. Reliability Engineering and System Safety, 84 :45–56, 2004.

[DML+14] J. Demgne, S. Mercier, W. Lair, J. Lonchampt, and M. Baudin. Méthodes de quasi monte-carlo pour l'évaluation de stratégies d'investissements. In Congrès Lambda Mu 19 de Maîtrise des Risques et Sûreté de Fonctionnement, Dijon, France, October 2014. IMdR, Institut pour la Maîtrise des risques.

[DMLL17] J. Demgne, S. Mercier, W. Lair, and J. Lonchampt. Modelling and numerical assessment of a maintenance strategy with stock through piecewise deterministic markov processes and quasi monte carlo methods. Proceedings of the Institution of Mechanical Engineers, Part O : Journal of Risk and Reliability, 231 :429 – 445, 2017.

[dSDZE12] B. de Saporta, F. Dufour, H. Zhang, and C. Elegbede. Optimal stopping for the predictive maintenance of a structure subject to corrosion. Proceedings of the Institution of Mechanical Engineers, Part O : Journal of Risk and Reliability, 226(2) :169–181, 2012.

[FK18] D. J. Fagnant and K. Kockelman. Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in austin, texas. Transportation, 45 :143–158, 2018.

[GBD02] A. Grall, C. Bérenguer, and L. Dieulle. A condition-based maintenance policy for stochastically deteriorating systems. Reliability Engineering and System Safety, 76 :167–180, 2002.

[Gee17] Alizee Geeraert. Contrôle optimal stochastique des processus de Markov déterministes par morceaux et application à l'optimisation de maintenance. Theses, Université de Bordeaux, 2017.

[GG92] A. Gersho and R. Gray. Vector quantization and signal compression. Kluwer Academic Press, 1992.

[GL00] S. Graf and H. Luschgy. Foundations of quantization for probability distributions. SERBIULA (sistema Librum 2.0), 1730, 2000.

[GMZ07] R. Gouvriveau, K. Medjaher, and N. Zerhouni. Du concept de PHM à la maintenance prédictive 1 : surveillance et pronostic. ISTE Editions, 2007.

[GSP19] M. Geist, B. Scherrer, and O. Pietquin. A theory of regularized markov decision processes. In ICML 2019 - Thirty-sixth International Conference on Machine Learning, Long Island, United States, June 2019. ICML 2019.

[HHC12] J. Hu, P. Hu, and H. Chang. A stochastic approximation framework for a class of randomized optimization algorithms. IEEE Transactions on Automatic Control, 57(1) :165–178, 2012.

[HLL96] O. Hernández-Lerma and J. Lasserre. Discrete-Time Markov Control Processes : Basic Optimality Criteria. volume 30, Applications of Mathematics. New-York : Springer-Verlaga, 1996.

[HLL99] O. Hernandez-Lerma and J. Lasserre. Further Topics on Discrete-Time Markov Control Processes. Springer Science & Business Media, 1999.

[How60] R. Howard. Dynamic programming and markov processes. MIT Press, Cambridge, 1960.

[HPvB+10] J. Hoey, P. Poupart, A. von Bertoldi, T. Craig, C. Boutilier, and A. Mihailidis. Automated handwashing assistance for persons with dementia using video and a partially observable markov decision process. Comput. Vis. Image Underst., 114 :503–519, 2010.

[Huy11] Khac Tuan Huynh. Quantification de l'apport de l'information de surveillance dans la prise de décision en maintenance. Theses, Université de Technologie de Troyes, France., 2011.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. Science, 220 :671 – 680, 1983.

[Kij89] M. Kijima. Some results for repairable systems with general repair. Journal of Applied Probability, 26(1) :89–102, 1989.

[KRL16] S. Keneally, M. Robbins, and B. Lunday. A markov decision process model for the optimal dispatch of military medical evacuation assets. Health Care Management Science, 19 :111–129, 2016.

[KSO+21] M. Kordestani, M. Saif, M. Orchard, R. Razavi-Far, and K. Khorasani. Failure prognosis and applications—a survey of recent literature. IEEE Transactions on Reliability, 70 :728–748, 2021.

[LBB+15] R. Lesobre, K. Bouvard, C. Bérenguer, V. Cocquempot, and A. Barros. Politique de maintenance dynamique pour un système multi-composant intégrant les informations de surveillance. In Congrès Lambda Mu 19, 2015.

[Lia16] Gwo-Liang Liao. Production and maintenance policies for an epq model with perfect repair, rework, free-repair warranty, and preventive maintenance. IEEE Transactions on Systems, Man, and Cybernetics : Systems, 46 :1129–1139, 2016.

[LND+06] J. Lee, J. Ni, D. Djurdjanovic, Hai Qiu, and H. Liao. Intelligent prognostics tools and e-maintenance. Comput. Ind., 57 :476–489, 2006.

[LPE98] E. Levin, R. Pieraccini, and W. Eckert. Using markov decision process for learning dialogue strategies. Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181), 1 :201–204 vol.1, 1998.

[LPS15] S. Lam, A. Pitrou, and S. Seibert. Numba : A llvm-based python jit compiler. In Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, pages 1–6, 2015.

[LZXZ18] K. Lin, R. Zhao, Z. Xu, and J. Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018.

[MC19] S. Mercier and I. Castro. Stochastic comparisons of imperfect maintenance models for a gamma deteriorating system. European Journal of Operational Research, 273(1) :237–248, 2019.

[ND04] M. Newby and R. Dagg. Optimal inspection and perfect repair. Ima Journal of Management Mathematics, 15 :175–192, 2004.

[NFD07] R. Nicolai, J. Frenk, and R. Dekker. Modelling and optimizing imperfect maintenance of coatings on steel structures. Environmental Economics, 2007.

[OCO17] B. Oliveira, M. A. Carravilla, and J. Oliveira. Fleet and revenue management in car rental companies : A literature review and an integrated conceptual framework. Omega-international Journal of Management Science, 71 :11–26, 2017.

[oD04] Ministry of Defence. Health and usage monitoring capability for land platforms (hums), standard 25-24. United Kingdom, 2004.

[Pag98] G. Pagès. A space quantization method for numerical integration. Journal of Computational and Applied Mathematics, 89(1) :1–38, 1998.

[Pal13] G. Palem. Condition-based maintenance using sensor arrays and telematics. International Journal of Mobile Network Communications & Telematics, abs/1309.1921, 2013.

[PP03] G. Pagès and J. Printems. Optimal quadratic quantization for numerics : the gaussian case. Monte Carlo Methods and Applications, 2003.

[PPP04] G. Pagès, H. Pham, and J. Printems. Optimal Quantization Methods and Applications to Numerical Problems in Finance, pages 253–297. Birkhäuser Boston, Boston, MA, 2004.

[Put94] M. Puterman. Markov Decision Processes : Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[Rob19] Elodie Robert. Health management of industrial vehicles and fleet maintenance optimization : taking into account operation constraints and mission planning. Theses, Université Grenoble Alpes, December 2019.

[SBJ11] N. Safaei, D. Banjevic, and A. Jardine. Workforce-constrained maintenance scheduling for military aircraft fleet : a case study. Annals of Operations Research, 186 :295–316, 2011.

[SBSR04] A. Schaefer, M. Bailey, S. Shechter, and M. Roberts. Modeling medical treatment using markov decision processes. In Operations Research and Health Care : A Handbook of Methods and Applications, pages 593–612, Boston, MA, 2004. Springer US.

[Sel05] A. Sellami. Méthodes de quantification optimale pour le filtrage et applications à la finance. Theses, Université Paris Dauphine, 2005.

[Sha86] R. Shachter. Evaluating influence diagrams. Oper. Res., 34 :871–882, 1986.

[THDV+17] Phuong Khanh Nguyen Thi, Khac Tuan Huynh, Phuc Do Van, Christophe Bérenguer, and Antoine Grall. A pomdp model for the joint optimization of condition monitoring quality and replacement decisions. In 10th International Conference on Mathematical Methods in Reliability (MMR 2017), 2017.

[TPA+13] P. Tixier, N. Peyrard, J. Aubertot, S. Gaba, J. Radoszycki, G. Caron-Lormier, F. Vinatier, G. Mollot, and R. Sabbadin. Modelling interaction networks for enhanced ecosystem services in agroecosystems. Advances in Ecological Research, 49 :437–480, 2013.

[Wan02] Hongzhou Wang. A survey of maintenance policies of deteriorating systems. European Journal of Operational Research, 139 :469–489, 2002.

[WHW05] P. Wu, J. C. Hartman, and G. R. Wilson. An integrated model and solution approach for fleet sizing with heterogeneous assets. Transportation Science, 39(1) :87–103, February 2005.

[WPC+18] M. Weinechter, R. Parouty, A. Cabarbaye, F. Farago, E. Remy, V. Bordeau, N. Matahri, L. Marle, J. Cardon, A. Adjadj, B. Colin, B. Iung, J. Léger, and J. Barbet. Hums -health & usage monitoring system -état de l'art et opportunités hums -health & usage monitoring system -state of art and opportunities. In Congrès Lambda Mu 21 " Maîtrise des risques et transformation numérique : opportunités et menaces ", Reims, France, 2018.

[Wu11] S. Wu. Optimal inspection policy for three-state systems monitored by variable sample size control charts. The International Journal of Advanced Manufacturing Technology, 55 :689–697, 2011.

[WW89] C. White and D. White. Markov decision processes, volume 39. Elsevier, 1989.

[WXL+17] Zeng Wei, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. Reinforcement learning to rank with markov decision process. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 945–948, 2017.

[YGTW13] S. Young, M. Gasic, B. Thomson, and J. Williams. Pomdp-based statistical spoken dialog systems : A review. Proceedings of the IEEE, 101 :1160–1179, 2013.

[YWC+17] Y. Yin, Y. Wang, T.C.E. Cheng, W. Liu, and J. Li. Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. Omega, 69 :17–28, 2017.

[ZCdRR+20] T. Zonta, C. Costa, R. da Rosa Righi, M. J. Lima, E. Silveira da Trindade, and G.P Li. Predictive maintenance in the industry 4.0 : A systematic literature review. Comput. Ind. Eng., 150 :106889, 2020.

[ZFB17] N. Zhang, M. Fouladirad, and A. Barros. Maintenance analysis of a two-component load-sharing system. Reliability Engineering & System Safety, 167 :67 – 74, 2017.

[ZFM10] E. Zhou, M. Fu, and S. Marcus. Solving continuous-state pomdps via density projection. IEEE Transactions on Automatic Control, 55 :1101–1116, 2010.

[ZYW19] W. Zhang, D. Yang, and H. Wang. Data-driven methods for predictive maintenance of industrial equipment : A survey. IEEE Systems Journal, 13 :2213–2227, 2019.