Conjurer la malédiction de la dimension dans le calcul du noyau de viabilité à l'aide de parallélisation sur carte graphique et de la théorie de la fiabilité : application à des dynamiques environnementales

Antoine Brias

N° d'ordre : D.U. : 2778
N° EDSPIC : 784

UNIVERSITÉ BLAISE PASCAL
U.F.R. Sciences et Technologies

ÉCOLE DOCTORALE DES SCIENCES POUR L'INGÉNIEUR

THÈSE

présentée pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

Spécialité : Informatique

par **Antoine BRIAS**

Conjurer la malédiction de la dimension dans le calcul du noyau de viabilité à l'aide de parallélisation sur carte graphique et de la théorie de la fiabilité : application à des dynamiques environnementales.

Soutenue publiquement le 15 décembre 2016 devant le jury.

Rapporteurs :	**Vincent MARTINET** Directeur de recherche, INRA, Thiverval-Grignon
	Jean-Christophe SOULIÉ Cadre de recherche, CIRAD, Montpellier
Examinateurs :	**Alaa CHATEAUNEUF** Professeur des Universités, Université Blaise Pascal, Aubière
	Luc DOYEN Directeur de Recherche, CNRS, Paris
Directeur de thèse :	**Jean-Denis MATHIAS** Chargé de Recherche, IRSTEA, Clermont-Ferrand
Co-directeur de thèse :	**Guillaume DEFFUANT** Directeur de Recherche, IRSTEA, Clermont-Ferrand

Avant-propos

Cette thèse a été effectuée au centre d'Aubière de l'Institut national de Recherche en Sciences et Technologies pour l'Environnement et l'Agriculture (IRSTEA) au sein du Laboratoire d'Ingénierie pour les Systèmes Complexes (LISC).

Je remercie grandement IRSTEA ainsi que la région Auvergne qui par leur soutien financier ont permis la réalisation de cette thèse.

Le document qui suit présente une thèse dont la majeure partie est rédigée en anglais, sous forme d'articles scientifiques soumis dans des revues internationales.

Résumé

La théorie de la viabilité propose des outils permettant de contrôler un système dynamique afin de le maintenir dans un domaine de contraintes. Le concept central de cette théorie est le noyau de viabilité, qui est l'ensemble des états initiaux à partir desquels il existe au moins une trajectoire contrôlée restant dans le domaine de contraintes. Cependant, le temps et l'espace nécessaires au calcul du noyau de viabilité augmentent exponentiellement avec le nombre de dimensions du problème considéré. C'est la malédiction de la dimension. Elle est d'autant plus présente dans le cas de systèmes incorporant des incertitudes. Dans ce cas-là, le noyau de viabilité devient l'ensemble des états pour lesquels il existe une stratégie de contrôle permettant de rester dans le domaine de contraintes avec au moins une certaine probabilité jusqu'à l'horizon de temps donné. L'objectif de cette thèse est d'étudier et de développer des approches afin de combattre cette malédiction de la dimension. Pour ce faire, nous avons proposé deux axes de recherche : la parallélisation des calculs et l'utilisation de la théorie de la fiabilité. Les résultats sont illustrés par plusieurs applications. Le premier axe explore l'utilisation de calcul parallèle sur carte graphique. La version du programme utilisant la carte graphique est jusqu'à 20 fois plus rapide que la version séquentielle, traitant des problèmes jusqu'en dimension 7. Outre ces gains en temps de calcul, nos travaux montrent que la majeure partie des ressources est utilisée pour le calcul des probabilités de transition du système. Cette observation fait le lien avec le deuxième axe de recherche qui propose un algorithme calculant une approximation de noyaux de viabilité stochastiques utilisant des méthodes fiabilistes calculant les probabilités de transition. L'espace-mémoire requis par cet algorithme est une fonction linéaire du nombre d'états de la grille utilisée, contrairement à l'espace-mémoire requis par l'algorithme de programmation dynamique classique qui dépend quadratiquement du nombre d'états. Ces approches permettent d'envisager l'application de la théorie de la viabilité à des systèmes de plus grande dimension. Ainsi nous l'avons appliquée à un modèle de dynamique du phosphore dans le cadre de la gestion de l'eutrophisation des lacs, préalablement calibré sur les données du lac du Bourget. De plus, les liens entre fiabilité et viabilité sont mis en valeur avec une application du calcul de noyau de viabilité stochastique, autrement appelé noyau de fiabilité, en conception fiable dans le cas d'une poutre corrodée.

Mots clés : Théorie de la viabilité, Malédiction de la dimension, Parallélisation, Théorie de la fiabilité, Programmation dynamique, Systèmes environnementaux.

Abstract

Dispel the dimensionality curse in viability kernel computation with the help of GPGPU and reliability theory: application to environmental dynamics

Viability theory provides tools to maintain a dynamical system in a constraint domain. The main concept of this theory is the viability kernel, which is the set of initial states from which there is at least one controlled trajectory remaining in the constraint domain. However, the time and space needed to calculate the viability kernel increases exponentially with the number of dimensions of the problem. This issue is known as "the curse of dimensionality". This curse is even more present when applying the viability theory to uncertain systems. In this case, the viability kernel is the set of states for which there is at least a control strategy to stay in the constraint domain with some probability until the time horizon. The objective of this thesis is to study and develop approaches to beat back the curse of dimensionality. We propose two lines of research: the parallel computing and the use of reliability theory tools. The results are illustrated by several applications. The first line explores the use of parallel computing on graphics card. The version of the program using the graphics card is up to 20 times faster than the sequential version, dealing with problems until dimension 7. In addition to the gains in calculation time, our work shows that the majority of the resources is used to the calculation of transition probabilities. This observation makes the link with the second line of research which proposes an algorithm calculating a stochastic approximation of viability kernels by using reliability methods in order to compute the transition probabilities. The memory space required by this algorithm is a linear function of the number of states of the grid, unlike the memory space required by conventional dynamic programming algorithm which quadratically depends on the number of states. These approaches may enable the use of the viability theory in the case of high-dimension systems. So we applied it to a phosphorus dynamics for the management of Lake Bourget eutrophication, previously calibrated from experimental data. In addition the relationship between reliability and viability is highlighted with an application of stochastic viability kernel computation, otherwise known as reliability kernel, in reliable design in the

case of a corroded beam.

Keywords: Viability theory, Curse of dimensionality, Parallel computing, Reliability theory, Dynamic programming, Environmental systems.

Remerciements

Ce travail de thèse est le fruit de trois années passées au LISC. Beaucoup de personnes y ont contribué, de près ou de loin, que ce soit par des apports scientifiques ou non. En tout premier, j'aimerai remercier mon directeur de thèse, Jean-Denis Mathias, pour m'avoir supervisé. Merci pour ton soutien inconditionnel, ton enthousiasme et tes conseils tout le long de notre collaboration. Je remercie également mon co-directeur de thèse, Guillaume Deffuant pour son aide précieuse et dont les remarques pertinentes ont permis de faire avancer ma réflexion. Merci à tous les deux d'avoir constitué le cadre de travail idéal à l'élaboration de cette thèse.

Je tiens ensuite à adresser mes remerciements au jury de cette thèse. Vincent Martinet et Jean-Christophe Soulié, vous avez ma gratitude pour avoir accepté de rapporter cette thèse. Je remercie les examinateurs, Luc Doyen et Alaa Chateauneuf, qui ont aussi fait partie des deux comités de thèse avec la participation de Bruno Bonté et David Hill. Ces comités ont complété l'encadrement de ma recherche, et j'en remercie vivement les membres.

Le LISC a été mon foyer d'accueil durant trois ans ; mes collègues ont pris part à la réussite de cette expérience, grâce à l'ambiance intellectuelle et chaleureuse, grâce aux pauses-café et leurs discussions parfois sérieuses, souvent décontractées. Une mention spéciale à mes compagnons de thèse, les autres doctorants du LISC, qui vivent leur aventure parallèlement. Je remercie toutes les personnes de l'Irstea avec qui j'ai noué des liens par le travail ou en dehors.

La vie ne se limitant pas au laboratoire, je souhaite remercier mes amis de tout horizon qui m'ont soutenu ; mes amis clermontois, l'équipe moustachue de handball, ceux de classe préparatoire et d'école d'ingénieur, et bien entendu ceux que j'ai connu avant tout cela, et qui se reconnaîtront.

Je souhaite remercier ma môme d'avoir croisé mon chemin, et de le partager maintenant. Merci Maryon.

Enfin, je remercie ma famille, mes frères et ma sœur d'être présents, malgré la distance. Ce travail est dédié à mes parents et mes grand-parents. Merci pour tout.

Table des matières

Chapitre 1

Introduction

Préambule

L'objectif de cette thèse est l'étude de solutions visant à repousser la malédiction de la dimension dans le cas du calcul de noyaux de viabilité. Deux axes de recherche sont explorés dans cette thèse : la parallélisation des calculs et l'utilisation de techniques issues de la théorie de la fiabilité. Ce chapitre d'introduction rappelle le cadre de la théorie de la viabilité dans lequel nous travaillons et définit plus prééminent la malédiction de la dimension.

1.1 Cadre de la thèse

1.1.1 La théorie de la viabilité

Établie dans le début des années 1990 par Jean-Pierre Aubin et ses collaborateurs [Aubin 90], la théorie de la viabilité fournit des concepts théoriques ainsi que des outils pratiques pour maintenir un système dynamique dans un ensemble donné de contraintes. Ce problème est courant en écologie et en environnement, où la gestion d'écosystèmes nécessite de mettre en place des politiques d'actions afin de conserver des propriétés d'intérêt du système [Martin 04, Mullon 04, Chapel 07, De Lara 09, Deffuant 11, Krawczyk 13b, Mathias 15]. Les concepts viabilistes s'appliquent aussi à d'autres domaines comme l'économie [Aubin 01], la robotique [Spiteri 00], l'aéronautique [Seube 00] ou même la classification des émotions [Bonneuil 14].

Le système dynamique étudié est composé de deux types de variables. Les variables d'état décrivent le système, et les variables de contrôle permettent d'agir dessus. Partant d'un état initial, le système évolue de différentes façons, suivant les contrôles choisis au cours du temps. L'ensemble de contraintes est un sous-ensemble de l'espace d'états au sein duquel on souhaite maintenir notre système. On peut supposer par exemple qu'au sein de cet ensemble de contraintes, certaines propriétés intéressantes du système sont conservées. Mais si le système est amené

à évoluer en dehors de cet ensemble, il sera alors considéré comme dégradé ou détérioré. Les contraintes peuvent être de plusieurs natures, économiques, écologiques, et peuvent évoluer dans le temps. En revanche, il n'y a pas de hiérarchie dans ces contraintes comme ça peut être le cas dans la théorie du contrôle optimal [Bernard 11]. Le système n'est pas considéré comme viable si aucun des choix de contrôle ne permet de le maintenir dans le domaine de contraintes. Le choix des contrôles se fait à tout instant afin de s'adapter aux modifications de l'état du système. En général, dans le cas déterministe, plusieurs politiques de contrôle différentes sont possibles pour maintenir le système dans son ensemble de contraintes. C'est une différence avec le contrôle optimal qui propose souvent une solution optimale unique. Dans le cas stochastique, le problème de viabilité est posé comme un problème de contrôle optimal.

1.1.2 Cadre des systèmes dynamiques contrôlés

Le système dynamique contrôlé déterministe que nous étudions s'écrit de manière générale, en temps discret :

$$\forall t \geq 0,\, x\left(t+dt\right) = x\left(t\right) + f\left(x\left(t\right), u\left(t, x\left(t\right)\right)\right) dt \tag{1.1}$$

où 0 représente la date initiale. $x\left(t\right) \in \mathbb{X}$ est l'état du système à l'instant t. L'espace d'état $\mathbb{X}$ est un sous-ensemble de $\mathbb{R}^n$ où n est le nombre de dimensions du problème. La dynamique f du système dépend à chaque pas de temps du contrôle $u\left(t, x(t)\right) \in U\left(t, x\right)$. Ce contrôle pris à l'instant t en fonction de l'état dans lequel le système se trouve, influence la dynamique au pas de temps suivant. L'espace des contrôles $U\left(t, x\right)$ est généralement discrétisé ; c'est alors un sous-ensemble de $\mathbb{R}^q$ où q est le nombre de valeurs discrétisées du contrôle disponible.

On définit alors l'ensemble de contraintes $K \subset \mathbb{R}^n$ dans lequel on souhaite maintenir le système. On dira que l'évolution du système est *viable* si :

$$\forall t \geq 0,\, x\left(t\right) \in K \tag{1.2}$$

1.1.3 Le noyau de viabilité déterministe

La théorie de la viabilité permet de déterminer comment choisir les actions à chaque instant afin de satisfaire les contraintes de manière durable. Le concept majeur de cette théorie est le *noyau de viabilité*. C'est l'ensemble des états initiaux du système étudié pour lesquels il existe au moins une suite de contrôles maintenant le système dans le domaine de contraintes, jusqu'à un horizon de temps donné. Le noyau de viabilité s'écrit :

$$Viab\left(K\right) = \left\{x\left(0\right) \in \mathbb{X} | \exists u\left(\cdot\right),\, \forall t \geq 0,\, x\left(t\right) \in K\right\} \tag{1.3}$$

Ainsi, si l'état initial $x(0)$ du système n'est pas dans ce noyau de viabilité, sa sortie du domaine de contraintes est inéluctable. *A contrario*, si son état initial fait partie du noyau de viabilité alors il existe une possibilité de maintenir le système dans le domaine de contraintes. Quand l'état du système s'approche de la frontière, une carte des *contrôles viables* est utilisée afin de rester dans le noyau de viabilité. Cette notion de distance à la frontière et les politiques de contrôles associées sont développées dans [Alvarez 11].

Le noyau de viabilité donne des informations importantes sur le système étudié. A titre d'exemple, si le noyau occupe la totalité de l'espace de contraintes, peu importe l'état initial du système, on aura des solutions pour maintenir ses propriétés. A l'opposé, si le noyau est vide, cela implique que le système actuel n'est pas viable comme c'est le cas dans [Domenech 11]. Il faut alors étudier d'autres possibilités de gestion. C'est le calcul de ce noyau qui sera au cœur des préoccupations de cette thèse, dans le cas de dynamiques pouvant être déterministes mais aussi stochastiques.

1.1.4 Extension aux dynamiques stochastiques

Originellement, les dynamiques étudiées par la théorie de la viabilité sont déterministes. Mais dans beaucoup de problèmes, certaines informations sont manquantes, ou la dynamique est mal connue, ou encore elle est bruitée par des phénomènes extérieurs. Les travaux de [Doyen 10] ont étendu les concepts viabilistes déterministes aux dynamiques stochastiques. Dans ce cas-là, pour un contrôle donné, la trajectoire du système n'est plus unique mais plusieurs trajectoires différentes sont possibles dues aux aléas de la dynamique. Le système en temps discret s'écrit alors :

$$\forall t \in [0,T],\ x(t+1) = x(t) + f(x(t), u(t, x(t)), w(t))\, dt \tag{1.4}$$

où T est l'horizon de temps considéré, fini ou infini. $w(t) \in \mathbb{W} \subset \mathbb{R}^s$ est le tirage d'une variable ou d'un vecteur aléatoire $\mathbf{W}(t)$ au pas de temps t, sur s dimensions de la dynamique. Celle-ci étant incertaine, il n'y a plus forcément de garantie totale du maintien du système dans le domaine de contraintes.

En considérant un scénario d'incertitude qui est une réalisation de la suite $(\mathbf{W}(0), \mathbf{W}(1), \ldots, \mathbf{W}(T-1))$, on définit une trajectoire $S(x(0), u, t) = \{x(0), x(1), \ldots, x(t))\}$ partant d'un état initial $x(0)$. La fonction $u : t \to U$ représente le contrôle donné à chaque pas de temps. Le *noyau de viabilité stochastique* est l'ensemble des états initiaux du système tels qu'il existe au moins une suite de contrôle pour laquelle la probabilité de maintenir sans interruption le système pendant au moins T pas de temps dans le domaine de contraintes est supérieure à un seuil β donné :

$$Viab\left(K\right)=\left\{x\left(0\right)\in\mathbb{X}|\exists u\left(\cdot\right),\mathbb{P}\left(S\left(x\left(0\right),u\left(\cdot\right),T\right)\in K\right)\geq\beta\right\} \tag{1.5}$$

1.1.5 Calcul du noyau de viabilité et malédiction de la dimension

1.1.5.1 Le calcul du noyau de viabilité ...

Le calcul du noyau de viabilité a fait l'objet de nombreuses études ces deux dernières décennies. S'il est possible de calculer théoriquement ce noyau pour certains cas [Aubin 90, Bernard 11], ce calcul devient vite infaisable au vu de la complexité des modèles mathématiques étudiés. Partant des travaux de [Saint-Pierre 94], de nombreux algorithmes ont vu le jour afin de traiter les spécificités propres aux différents problèmes [Krawczyk 13a].

L'algorithme originel de Patrick Saint-Pierre est basé sur la discrétisation de l'espace d'états en répartissant les points selon une grille régulière, et calcule une approximation numérique du noyau de viabilité qui est un sous-ensemble des points de cette grille. Si aucune évolution viable n'existe partant d'un point de la grille, cet état est supprimé de l'ensemble des états viables. Ce processus est itéré jusqu'à ce que le nombre d'état viable ne diminue plus. L'ensemble obtenu est une approximation du noyau de viabilité par l'extérieur. D'autres algorithmes utilisant des méthodes d'ensembles de niveaux [Mitchell 05] ou des méthodes de classification supervisée existent [Deffuant 07, Wei 12]. Pour des problèmes en deux dimensions, le schéma Ultra-Bee [Bokanowski 06] limite l'effet diffusif des algorithmes identifié par [Cardaliaguet 00]. Dans ce cas là, une fonction valeur est utilisée pour résoudre le problème de viabilité, vu comme un problème de contrôle optimal.

Des algorithmes ont été créés afin de résoudre des problèmes spécifiques. Ainsi, Maidens et ses collaborateurs [Maidens 09] ont développé un algorithme utilisant des méthodes lagrangiennes adaptées à des dynamiques linéaires et un ensemble de contraintes convexe et compact. Il perd en efficacité pour les autres types de dynamiques. L'algorithme de [Mullon 04] est conçu pour résoudre des problèmes de viabilité où le noyau de viabilité est convexe, ce qui n'est pas connu *a priori*.

Concernant le calcul de noyau de viabilité stochastique, le principal algorithme, développé par [Doyen 10], s'appuie sur la programmation dynamique stochastique, calculant par récurrence à rebours une fonction valeur pour une grille d'états afin de déterminer les états appartenant au noyau de viabilité. Pour faire ce calcul, il est nécessaire de calculer les probabilités de transition entre chaque état de la grille pour chaque valeur discrétisée du contrôle. Le nombre total de ces transitions dépend quadratiquement du nombre d'états de la grille et proportionnellement du nombre de valeurs discrétisées du contrôle. Cela limite le calcul de noyaux de viabilité quand la taille de la grille et le nombre de dimensions augmentent.

1.1.5.2 ... hanté par la malédiction de la dimension

La dimension du problème a un impact fort sur le calcul du noyau de viabilité. Quand le nombre de dimensions augmente, le nombre de données à stocker et les temps de calcul augmentent car le nombre de points nécessaire pour discrétiser l'espace d'état croît exceptionnellement avec la dimension du problème. Par conséquent, le nombre total de transitions contrôlées à considérer augmente lui aussi. Cette malédiction de la dimension, originellement identifiée par [Bellman 59] concerne de nombreux domaines, comme l'apprentissage statistique ou la fouille de donnée, et affecte le calcul de noyau de viabilité.

Ce problème est exacerbé en viabilité stochastique. Dans ce cas, l'algorithme de programmation dynamique stochastique [Doyen 10] nécessite le calcul des probabilités de transition, ce qui devient vite rédhibitoire quand le nombre de dimension augmente (et donc le nombre de points de la grille utilisée). Dans ce dernier cas, il faut alors avoir les capacités de stocker les transitions entre chaque couple de points de la grille, pour chaque contrôle. De plus, ces transitions doivent être recalculées à chaque étape si elles sont dépendantes du temps entraînant à la fois des problèmes de mémoire et des problèmes de temps de calcul.

1.1.5.3 Pourquoi lutter contre cette malédiction ?

La malédiction de la dimension empêche actuellement le calcul de noyaux de viabilité pour des systèmes dynamiques de grande dimension. Réduire les effets de cette malédiction est un enjeu majeur notamment pour la gestion de systèmes environnementaux. Actuellement, le nombre de dimensions de l'espace d'états des problèmes traités ne dépasse pas six [Chapel 08], et beaucoup de modèles étudiés sont des modèles-jouets ou très simplifiés. Cela empêche l'analyse viabiliste de systèmes plus complexes en terme de nombre de variables d'état. A titre d'exemple, la version en deux dimensions du modèle de Carpenter [Carpenter 99] d'eutrophisation des lacs est très étudiée en viabilité [Martin 04, Rougé 13]. Il est en revanche difficile d'obtenir le noyau de viabilité stochastique d'un modèle d'eutrophisation en cinq dimensions comme celui proposé par DeAngelis [DeAngelis 89] qui prend en compte plusieurs réseaux trophiques.

1.2 Objectifs de la thèse

1.2.1 Comment lutter contre cette malédiction ?

Afin de lutter contre cette malédiction, on distingue deux types d'approches (Fig.1.2.1) dans ce travail : i) les approches technologiques, telle que l'optimisation de l'espace mémoire et des temps de calcul nécessaires aux algorithmes viabilistes

par le biais de nouvelles technologies ; ii) les approches méthodologiques, telle que la conception d'algorithmes nécessitant moins de ressources en mémoire et en temps. Ces deux approches ne sont pas contradictoires et peuvent être liées afin de concevoir des algorithmes rapides, sans grand besoin d'espace-mémoire, utilisant des technologies efficaces.

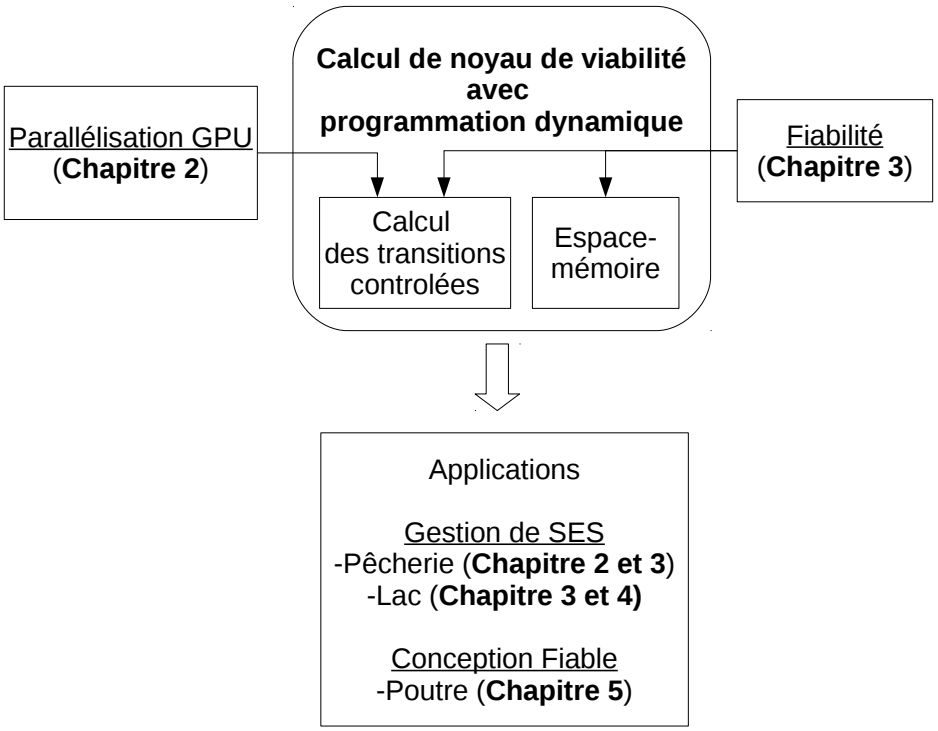

FIGURE 1.2.1 – Schéma d'organisation de la thèse.

1.2.2 Paralléliser les calculs

La parallélisation est efficace pour accélérer de nombreux types de calculs, en dynamique des fluides, en traitement d'image ou encore en décryptage. Il faut pour cela un algorithme adapté, dans lequel les parties parallélisées font l'objet d'un traitement particulier. De nombreuses techniques ont été développées afin

de paralléliser des programmes informatiques à différents niveaux. L'idée d'avoir plusieurs processeurs dans un même ordinateur datent des années 1960. Depuis d'autres solutions sont apparues, comme le multicœur dû à la miniaturisation des processeurs, ou la mise en parallèle de plusieurs machines. L'utilisation de coprocesseurs fournit de nouvelles fonctionnalités aux processeurs, permettant de réaliser de nombreux calculs simultanément. Ainsi, les cartes graphiques ont vu leur usage détourné afin d'assister le processeur central à l'aide, par exemple, de la bibliothèque CUDA.

Les algorithmes basés sur une grille demandent généralement de nombreux traitements pouvant être effectués simultanément au lieu d'être effectués séquentiellement. L'implémentation de VIKAASA [Krawczyk 13a] utilise les fonctions de parallélisation de MATLAB® afin de calculer des noyaux de viabilité dans le cadre de la gestion de pêcherie. Quant aux algorithmes de programmation dynamique, des solutions de parallélisation existent [Jacob 10].

Nous présentons dans le **chapitre 2** la parallélisation sur carte graphique d'un algorithme de programmation dynamique. Nous appliquons ce type de parallélisation au calcul de noyaux de viabilité déterministes, mais il est aussi applicable au calcul de noyaux de viabilité stochastiques.

1.2.3 Utiliser les techniques de la fiabilité

Plusieurs pistes ont été explorées afin de repousser la malédiction de la dimension. Des algorithmes ont été élaborés mais souvent dans des cas particuliers. [Bonneuil 06] approche le noyau de viabilité à horizon de temps fini à l'aide de recuit simulé. Cette méta-heuristique permet de déterminer si un état du domaine est viable, si l'horizon de temps et le nombre de contrôles disponibles ne sont pas très grands. [Deffuant 07] a introduit des fonctions de classifications dans l'algorithme de Saint-Pierre afin de réduire le nombre d'appels au modèle et de calculer la frontière du noyau de viabilité. Ainsi, l'utilisation de machines à vecteurs de supports permet de limiter la malédiction de la dimension sur l'espace des contrôles, en utilisant une méthode d'optimisation comme une descente de gradient afin de trouver les contrôles viables. L'utilisation des méthodes lagrangiennes limite les effets de la malédiction de la dimension [Maidens 09] mais les avantages de cet algorithme ne s'étendent pas aux dynamiques non-linéaires. Des approches s'affranchissant de la grille d'états ont été mises en place, en utilisant à la place des états de l'espace sélectionnés aléatoirement [Djeridane 08a, Djeridane 08b]. Un réseau de neurones permet alors de classer ces états comme viables ou non-viables. Néanmoins, cet algorithme ne garantit pas l'obtention de la solution exacte. Dans le cas des noyaux de viabilité stochastiques, l'algorithme de Luc Doyen utilisant la programmation dynamique, est très impactée par la malédiction de la dimension. En effet, le calcul de la fonction valeur fait intervenir la probabilité de transition

d'un état vers chacun des autres du système. Une manière très répandue de calculer ces probabilités est d'utiliser des méthodes de Monte-Carlo. Mais ces méthodes deviennent vite caduques quand la dimension du problème augmente.

De nombreux travaux existent en programmation dynamique approchée afin de réduire les effets de cette malédiction. Dans ce domaine, les deux approches principales sont l'utilisation de méthodes d'apprentissage et de simulation, et l'approximation de la fonction valeur utilisée [Powell 09]. C'est cette dernière piste que nous avons choisie d'approfondir. Au niveau du calcul des probabilités de transition, nous avons choisi d'étudier les outils d'un domaine proche de la théorie de la viabilité, la théorie de la fiabilité, afin de s'affranchir des méthodes de Monte-Carlo. C'est un champ de recherche dont l'objectif est de calculer la probabilité de défaillance d'un système par l'approximation de surface de réponse. Ce champ de recherche a de nombreuses applications en ingénierie des structures [Rackwitz 01], en science des matériaux [Mathias 13], en maintenance industrielle [Rausand 98] et en écologie [Naeem 98]. La probabilité de défaillance est la probabilité qu'a un système de ne plus respecter des contraintes données, et donc de perdre ses propriétés. L'une des principales manières de calculer cette probabilité utilise des méthodes de Monte-Carlo donnant une estimation statistique de cette probabilité. Mais depuis les années 70, des méthodes numériques plus efficaces ont été développées afin de traiter des problèmes de grande dimension. L'échantillonnage directionnel est une extension des méthodes de Monte-Carlo se focalisant sur une zone d'intérêt [Ditlevsen 07]. D'autres méthodes, plus utilisées, comme la méthode FORM [Maier 01], fournissent une approximation du domaine de contraintes permettant le calcul de la probabilité de défaillance à l'aide de techniques numériques. Les liens avec la théorie de la viabilité ont été établies par [Rougé 14], introduisant la notion de *noyau de fiabilité*, le penchant fiabiliste du noyau de viabilité stochastique.

En utilisant ces méthodes fiabilistes à un niveau local, on obtient une approximation de la fonction valeur demandée par la programmation dynamique. Ce couplage est présenté dans le **chapitre 3**. Nous présentons une application de la viabilité en théorie de la fiabilité, notamment en maintenance et en conception fiable dans le **chapitre 5** de cette thèse.

1.2.4 Applications

1.2.4.1 Gestion de systèmes environnementaux

Si les applications de cette théorie sont multiples, elle est notamment utilisée en développement durable afin de fournir aux gestionnaires de socio-écosystèmes (SES) des méthodes de régulation [Deffuant 11, Bernard 13]. Ce travail est principalement illustré par deux cas d'application en gestion de SES. Le premier concerne

la gestion de pêcherie [Krawczyk 13a] où l'enjeu est la régulation d'une ressource halieutique afin de satisfaire : i) des contraintes écologiques, comme le renouvellement de la ressource ; ii) des contraintes économiques, comme la pérennité des activités du pêcheur. En connaissant la dynamique de la biomasse de poissons et en ayant une possibilité d'agir sur l'effort de pêche, on obtient une gestion durable de l'écosystème. La gestion n'est pas nécessairement optimale à court-terme, en terme de profit pour le pêcheur, mais permet au système d'être pérenne sur le long terme.

Le second exemple d'application est la régulation de l'eutrophisation des lacs [Rougé 13, Martin 04]. Un lac est un écosystème subissant de fortes pressions anthropiques. Par exemple, le rejet de produits d'épandage agricole dans les affluents du lac a pour conséquence un sur-enrichissement de l'eau en nutriments. Cela nuit à la qualité de l'eau du lac, entraînant potentiellement une prolifération d'algues et une anoxie des populations de poissons. Cela impacte dès lors le tourisme et les activités de loisir aux alentours. Il y a donc des enjeux sociétaux à réguler la quantité de nutriment dans le lac.

Ces deux exemples illustrent bien les composants principaux du problème de viabilité :

- le *système dynamique contrôlé*, qui va évoluer dans le temps et sur lequel le décideur a une possibilité d'action, en l'occurrence la biomasse de poissons ou la quantité de nutriments dans le lac.

- le *domaine de contraintes de viabilité*, qui doivent être respectées afin d'avoir un système *viable* dans le temps. Ces contraintes sont d'ordre économique telles que les activités de la pêcherie ou des agriculteurs. Mais elles sont aussi être d'ordre écologique : la biomasse des poissons ne doit pas passer sous un certain seuil sous peine de voir la population péricliter et le taux de nutriments dans le lac ne doit pas être au-dessus d'une certaine limite afin de ne pas avoir une eau de mauvaise qualité.

Les méthodes étudiées dans cette thèse ont, entre autres, pour objectif d'être appliquées à la gestion de SES. Actuellement, il est difficile d'appliquer ces méthodes à un système si le nombre de dimensions est trop important et les principales applications se basent sur des systèmes de faible dimension. Dans le cas de l'eutrophisation des lacs, on considère principalement deux variables : la masse de nutriments dans le lac, et la masse de nutriments apportée au lac. On peut dès lors complexifier la dynamique en rajoutant d'autres variables comme la masse de nutriments se retrouvant au fond du lac dans les sédiments par exemple. Néanmoins les données nécessaires afin de calibrer le modèle doivent être disponibles. Plus on a de variables, plus le nombre de données à avoir est grand. Avant de pouvoir calculer un noyau de viabilité, il faut alors passer par l'étape de calibration

et de validation du modèle. L'aléa doit aussi être calibré dans le cas stochastique, ainsi que les contrôles disponibles. Dans le **chapitre 4**, nous proposons la calibration d'un modèle d'eutrophisation pour le cas du lac du Bourget. Le modèle utilisé est basé sur la dynamique du phosphore, entrant et restant dans le lac. C'est une première approche, travaillant sur un modèle simplifié de l'eutrophisation des lac, en vue d'inclure d'autres mécanismes en jeu, comme les interactions entre les niveaux trophiques lacustres.

1.2.4.2 Conception fiable

Dans le **chapitre 3**, une méthode d'approximation de noyaux de viabilité stochastiques est présentée, basée sur des techniques fiabilistes. Or, si la théorie de la fiabilité est utilisable en viabilité, l'inverse est aussi vrai, comme le montre les travaux existants sur les noyaux de fiabilité [Rougé 14]. Nous présentons dans le **chapitre 5** une application de ces noyaux de fiabilité en conception fiable. L'objectif est de déterminer les propriétés initiales d'un système, d'un matériau par exemple, afin de résister dans le temps aux contraintes qui lui sont appliquées. Nous exploitons le concept de noyau de fiabilité dans ce cadre afin de fournir des normes de conception pour des systèmes dynamiques.

1.2.5 Organisation du document

La parallélisation étudiée est développée dans le **chapitre 2**. On y détaille la parallélisation d'un algorithme de calcul de noyaux de viabilité dans le cas déterministe. Le calcul d'un noyau de viabilité d'un modèle de pêcherie illustre ce chapitre. Le **chapitre 3** est consacré à l'approche méthodologique. Notre algorithme couplant programmation dynamique et méthodes fiabilistes y est présenté avec des applications en gestion de pêcherie et d'eutrophisation. Le **chapitre 4** présente la calibration d'un modèle d'eutrophisation de lac à partir de données réelles. Des premiers résultats y sont exploités. Le **chapitre 5** illustre l'utilisation de la théorie de la viabilité en fiabilité des systèmes, avec l'étude de conception fiable d'une poutre soumise à la corrosion. Enfin, le **chapitre 6** conclut cette thèse et présente des perspectives à ce travail.

Chapitre 2

Parallélisation sur carte graphique du calcul des transitions

2.1 La parallélisation pour accélérer le calcul de noyau de viabilité

Le nombre de calculs nécessaires afin d'identifier l'ensemble des états viables ralentit le calcul de noyau de viabilité. De manière générale, le fait d'utiliser des processeurs plus puissants et un langage de programmation adapté permet de réduire ces problèmes. Mais ici, les temps de calculs évoluent de manière exponentielle avec la dimension et ces solutions ne suffisent plus. Afin de monter en dimension et de repousser la malédiction de la dimensionnalité, on peut optimiser le traitement des informations. Le principe de la parallélisation consiste à utiliser une architecture adaptée afin de faire des calculs parallèles, au lieu de les faire séquentiellement.

Le but de cet article est d'identifier les possibilités de parallélisation du calcul de noyau de viabilité déterministe au moyen d'un algorithme de programmation dynamique. L'algorithme est découpé en deux étapes : i) la matrice de transition d'un état à l'autre de la grille est calculée, ii) une récurrence est effectuée afin de déterminer les états appartenant au noyau de viabilité. La méthode de parallélisation à utiliser dépend de l'algorithme étudié. En l'occurrence, l'algorithme de programmation dynamique autorise le calcul parallèle d'un nombre important de tâches. Dans ce cas, la parallélisation sur carte graphique est adaptée. Cette technique utilise le processeur graphique afin de bénéficier de ses capacités de traitement massivement parallèle.

2.2 Présentation et contributions de l'article

Cet article a été publié dans la revue *Computational Management Science*. Les principales contributions sont les suivantes :

- nous analysons l'algorithme de programmation dynamique afin d'identifier les étapes pouvant être parallélisées. Ainsi, si l'ordre des tâches à effectuer n'importe pas, nous les effectuons simultanément ;
- nous codons l'algorithme en C++ en utilisant la technologie CUDA, afin de paralléliser les calculs sur carte graphique ;
- nous montrons l'importance de la parallélisation de la matrice de transition par rapport à l'actualisation de la fonction valeur. La technologie CUDA est pertinente pour ce type de problème. En effet, elle permet de paralléliser un nombre très important de tâches ne demandant pas beaucoup de ressources. C'est ici le cas pour le calcul des transitions par la dynamique contrôlée ;
- nous utilisons ce programme sur un problème de gestion de pêcherie en faisant varier le nombre de dimensions du problème, *i.e.* en faisant varier les nombres d'espèces en présence.

2.3 Conclusions

La parallélisation est un moyen efficace de diminuer les temps de calcul quand le problème s'y prête. C'est le cas de l'algorithme de programmation dynamique utilisé pour le calcul de noyau de viabilité déterministe. Dans cet article, on note qu'il est possible de paralléliser les deux étapes de l'implémentation de l'algorithme, amenant des gains en temps différents. La parallélisation du calcul de la matrice de transition se révèle être la plus intéressante divisant jusqu'à 30 fois le temps de calcul nécessaire à cette étape pour des problèmes à 6 dimensions. La parallélisation CUDA permet de repousser les effets de la malédiction de la dimension sur le temps de calcul.

Cet article amène des perspectives intéressantes pour le calcul de noyau de viabilité stochastique où l'algorithme de programmation dynamique nécessite aussi le calcul d'une matrice de transition. Une des limitations de cette approche concerne l'espace-mémoire utilisé dont la taille reste inchangée.

2.4 Texte de l'article

Comput Manag Sci
DOI 10.1007/s10287-015-0246-x

ORIGINAL PAPER

Accelerating viability kernel computation with CUDA architecture: application to bycatch fishery management

Antoine Brias[1] · **Jean-Denis Mathias**[1] · **Guillaume Deffuant**[1]

Received: 4 December 2014 / Accepted: 8 December 2015

Abstract Computing a viability kernel consumes time and memory resources which increase exponentially with the dimension of the problem. This curse of dimensionality strongly limits the applicability of this approach, otherwise promising. We report here an attempt to tackle this problem with Graphics Processing Units (GPU). We design and implement a version of the viability kernel algorithm suitable for General Purpose GPU (GPGPU) computing using Nvidia's architecture, CUDA (Computing Unified Device Architecture). Different parts of the algorithm are parallelized on the GPU device and we test the algorithm on a dynamical system of theoretical population growth. We study computing time gains as a function of the number of dimensions and the accuracy of the grid covering the state space. The speed factor reaches up to 20 with the GPU version compared to the Central Processing Unit (CPU) version, making the approach more applicable to problems in 4 to 7 dimensions. We use the GPU version of the algorithm to compute viability kernel of bycatch fishery management problems up to 6 dimensions.

Keywords Viability kernel · Dynamic programming · CUDA · GPU · Fishery management

Mathematics Subject Classification 90B50-management decision making, including multiple objectives · 90C39-dynamic programming

✉ Antoine Brias
antoinebrias@gmail.com; antoine.brias@irstea.fr

[1] Irstea, UR LISC Laboratoire d'Ingénierie des Systèmes Complexes, 24 Avenue des Landais, BP 20085, 63172 Aubière, France

Published online: 02 January 2016

1 Introduction

Viability theory provides mathematical and numerical tools and concepts for maintaining a dynamical system within a set of states (called the constraint set). This theory has numerous potential applications in food processing, finance, economics, environment (Sicard et al. 2012; Béné et al. 2001; Doyen et al. 2012; Bernard and Martin 2013; Andrés-Domenech et al. 2014; Chapel et al. 2008, 2010; Mathias et al. 2015). A essential step in this approach lies in the computation of the viability kernel. This is the set of states for which there exists a control policy that keeps the system in the constraint set for some (finite or infinite) time. Since the 1990's, several algorithms have been developed to compute viability kernels in different application fields (Saint-Pierre 1990; Bokanowski et al. 2006). Some of them are adapted to finite horizon problems (Djeridane and Lygeros 2008), linear dynamics (Kaynama and Oishi 2013) or particular kinds of problems such as single output nonlinear control systems affine in the control (Turriff and Broucke 2009; Mattioli and Artiouchine 2003). The concepts have been extended to the stochastic case (Doyen and De Lara 2010; Rougé et al. 2013), in which dynamics includes random variables accounting for uncertainties. However, most of them are limited by their computational complexity. When the number of dimensions increases, the required computation time and memory increase exponentially because they are mainly based on a grid covering the state space.

This curse of dimensionality is a moot point in a number of domains, like data mining, numerical analysis (Donoho 2000) and multiple approaches aim at mitigating it and at speeding up viability kernel computing. For instance, Bonneuil (2006) attempts to handle large dimensional state space with computing control policies on a given time horizon with simulated annealing. Deffuant et al. (2007) use Support Vector Machines for approximating viability kernels. Maidens et al. (2009) try to overcome this curse by using Lagrangian methods, which do not call for a grid. It offers particularly interesting performances for linear dynamics. Designing new algorithms is a crucial way to reduce the curse of dimensionality, but existing technological solutions can also lead to important gains of time. Some programs like Vikaasa, used in by-cash fishery management, embed a multi-core implementation of viability kernel computation (Krawczyk et al. 2013). However, in recent years, another technical improvement arose which seems well adapted for the viability kernel problematic: the GPU parallel computing technology. Historically, GPUs are electronic components used in computer graphics hardware. They are getting more widely used to accelerate computations in many fields (Mametjanov et al. 2012; Cekmez et al. 2013; Sabo et al. 2014). Due to its architecture, the GPUs solution is advantageous for problems for which a parallel execution is possible (Goldsworthy 2014). It often exceeds the capacity of computing clusters in terms of performance and for a lower cost. Thus, NVIDA has created the CUDA platform to offer tools and to allow programmers to use the parallel architecture of graphic cards. The basic point is to separate the program in two parts: the serial code executed by the host (the Central Processing Unit (CPU)), and the parallel code, executed in the GPU threads across multiple processing elements.

In this paper, we use the standard viability algorithm (Saint-Pierre 1990) which can be considered as a particular case of dynamic programming. To illustrate our work, the implemented algorithm is tested on a basic multidimensional model of population

dynamics. In particular, we compare the computation time between the parallelized version and the sequential one. Finally, the algorithm is used to compute the viability kernel in a case of bycatch fishery management.

2 Viability theory and dynamic programming

2.1 Viability kernel

We consider a discrete dynamics in which h is the function mapping the state and the control at time t with the state at the next time step $t+1$:

$$x(t+1) = h(x(t), u(t)) \tag{1}$$

with $x(t) \in \mathcal{X}$ the set of states, $u(t) \in \mathcal{U}(x(t))$ which is a finite set of possible controls allowing the regulation of the dynamics and $h(.,.) : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, associating a state $x(t) \in \mathcal{X}$ and a control $u(t) \in \mathcal{U}(x(t))$ with a successor $x(t+1) \in \mathcal{X}$. For example, it associates the closest grid point of the successor obtained with an Eulerian scheme defined with an appropriate local parameter. We want to assess the viability kernel $Viab_T(K)$; the set of initial states for which a control strategy exists to maintain the system inside K, the set of desirable states (the viability set), until T (a finite horizon). This is rewritten in Eq. 2:

$$Viab_T(K) = \{x_0 \in K | \exists (u(0), u(1), \ldots, u(T-1)),\ \forall t \in \{0, 1, \ldots, T\},\ x(t) \in K\} \tag{2}$$

with $x(t)$ the state of the system at date t and $u(t)$ the chosen control value at t, which maintains the system in the viability set K.

Different algorithms are available for computing this set. Because of the approximation of the dynamical system on a grid, these algorithms are submitted to the dimensionality curse: the necessary time and memory increase exponentially with the number of dimensions of the problem. We choose to focus on a dynamic programming algorithm which is very standard and well suited for a parallel approach.

2.2 Dynamic programming

We assume that the dynamics of the system fulfils the condition ensuring the convergence of the approximation to the actual kernel, when the resolution of the discretization tends to 0 (Saint-Pierre 1990). The dynamic programming algorithm is a backward recursive algorithm and determines a value function: $V(t, x)$ at date t. The seed value at T is:

$$V(T, x) = \begin{cases} 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases} \tag{3}$$

In this case, the value function directly determines the viability kernel. At T, being viable is the property of belonging to the viability kernel. The linear recurrence relation is given by:

$$\forall t \in \{0, T-1\}, \quad V(t,x) = \begin{cases} \max\limits_{u \in \mathcal{U}(x)} V(t+1, h(x,u)) & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases} \tag{4}$$

Thus, at $t = 0$:

$$V(0,x) = \begin{cases} 1 & \text{if } x \in Viab_T(K) \\ 0 & \text{if } x \notin Viab_T(K) \end{cases} \tag{5}$$

The principles of dynamic programming are respected because, at each step time:

- each point of the grid is a subproblem (V has to be computed for each point);
- a value function has to be optimized independently for these points;

The value function V takes only two values: 0 and 1. We are looking for initial states, for which the value function at time horizon T is 1. Equation 4 is equivalent to say that the value function $V(t,x)$ is equal to 1 if a control $u \in \mathcal{U}(x)$ exists, maintaining the system in the constraint set K. Otherwise, the value function $V(t,x)$ equals to 0. The backward dynamic programming equation defines the value function $V(x,t)$. By writing a max instead of a sup, we implicitly assume the existence of an optimal solution for each time t and state x. The algorithm involves two main steps:

- For each point of the grid, we store all the successors (the point of the grid at the next time step) obtained for each choice of control (the possible set of actions at each point is supposed finite).
- The second part performs iterations of kernel approximations by progressively excluding points of the grid until reaching a fixed point or a predefined number of time steps.

Thus we have the algorithm Algorithm 1 as follows:

Algorithm 1 Detailed Dynamic Programming Viability Kernel Algorithm

Storing successors:
$$\begin{cases} i = 0 \\ K^0 = K \\ \text{store coordinates, } \forall x \in K \\ \text{store } h(x,u), \forall x \in K, \ \forall \mathbf{u} \in \mathcal{U}(x) \end{cases}$$

Iterating kernel approximation:
$$\begin{cases} \text{repeat} \\ \quad K^{i+1} = \left\{ x \in K^i \,|\, V(i+1,x) = 1 \right\} \\ \quad i = i+1 \\ \text{until } K^{i+1} = K^i \text{ OR } K^{i+1} = \emptyset \text{ OR } i = T \end{cases}$$

$$Viab_T(K) = K^i$$

2.3 Parallelizing dynamic programing algorithm

Dynamic programming is well suited to parallelization in the two main parts. During the storing of successors, all successors $x(t+1)$ of each state and each value of u, defined in Eq. 1, can be computed separately. In the second part of the algorithm, it is possible to parallelize the update of the value function $V(t,x)$ applied to each state (Eq. 4). In both cases, computations are completely independent for one step of time. The parallelizing of all these calculations thus improves the performance in terms of computation time. Since GPUs have hundreds of processor cores running in parallel, they can be more efficient than using traditional CPU architectures.

3 Using GPU for computing viability kernels

3.1 GPU architecture

As they are widely used in many scientific computing fields, graphics hardware programming for general purpose computing is well documented, this includes ways to write and debug the code (Langdon 2011). Providing highly effective data parallelism, GPU cores contain multiple threads running at the same time which can accelerate viability algorithms. The CUDA programming framework enables the use of all the cores belonging to the GPU for general purpose computing. Programs written in this paper use the C++ CUDA framework. A GPU computing view consists in splitting the input data between the different threads, and performing operations on these pieces of data, independently and thus achieving parallelization. This kind of computing encourages to rethink programs to make this parallelization possible. Figure 1 illustrates the architecture of a GPU device which is seen as a grid housing blocks. Each block involves a number of threads performing the computations. The GPU device owns a global memory, each block has a memory shared by all of its threads, and each thread owns a smaller local memory. Here, we use the global memory to store big arrays of data with which all the threads can interact. In a block, a thread is indexed over 3 dimensions (a block can be shown as a cube of threads), thus the indexation has to be carefully done. Each thread and each block are indexed at their level. In our implementation, we want to get the ID $threadID$ of each one-dimensional thread in a grid of 3D blocks. According to the CUDA framework writing convention, this ID is given by:

$$\begin{aligned} threadID = {} & threadIdx.i + blockDim.i * blockIdx.i \\ & + gridDim.i * blockIdx.j * blockDim.i \\ & + gridDim.i * gridDim.j * blockIdx.k * blockDim.i \end{aligned} \quad (6)$$

Here, $threadIdx.i$ means the coordinates i of $threadIdx$. $gridDim$ contains the dimensions of the grid (the grid owns $gridDim.i \times gridDim.j \times gridDim.k$ blocks), $blockIdx$ contains the block indexes within the grid (in 3 dimensions: $blockIdx.i$, $blockIdx.j$ and $blockIdx.k$), $blockDim$ contains the dimensions of the block (a block contains $blockDim.i \times blockDim.j \times blockDim.k$ threads in 3 dimensions) and $threadIdx$ contains the thread indexes within the block (in 3 dimensions too, but

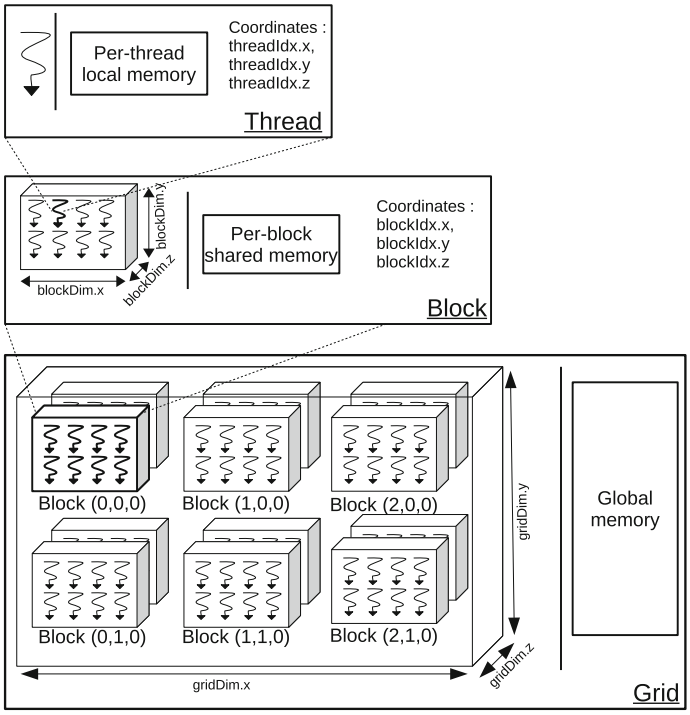

Fig. 1 GPU device architecture. The device is composed of a grid of threads blocks. Each level possesses its own memory. The parallelized tasks, like computation of controlled evolutions and computations of the value function, are made separately on the threads. A grid of $3 \times 2 \times 2$ blocks of $4 \times 2 \times 1$ threads is shown

for simplicity, only one is used here: $threadIdx.i$). These variables are mainly used in assigning the work per block and thread. The maximum values in each dimension in both $blockDim$ and $gridDim$ are GPU-dependent. At least, the i component needs to be declared. By default, the value for the j and k components is 1.

3.2 Overall strategy

The organization of the algorithm is summed up in Fig. 2. The figure shows the storing of trajectories and the iterating kernel approximation steps of Algorithm 1. The steps are:

- the initialization and setting of the GPU device (a);
- the initialization of the model parameters (b). The user inputs these parameters;
- the initialization of CUDA parameters (number of blocks, number of threads by blocks in each dimension) (c);
- the creation of empty arrays, which will contain controlled evolutions and the value function of each point using the CUDA instruction $cudaMalloc$ (d);
- the copy of parameters from host to the GPU device (e) using the CUDA instruction $cudaMemcpy$ with the flag $cudaMemcpyHostToDevice$;
- the computation of successors of each point i.e. the storing of next point of the trajectory for each possible value of the control Algorithm 1. This operation is

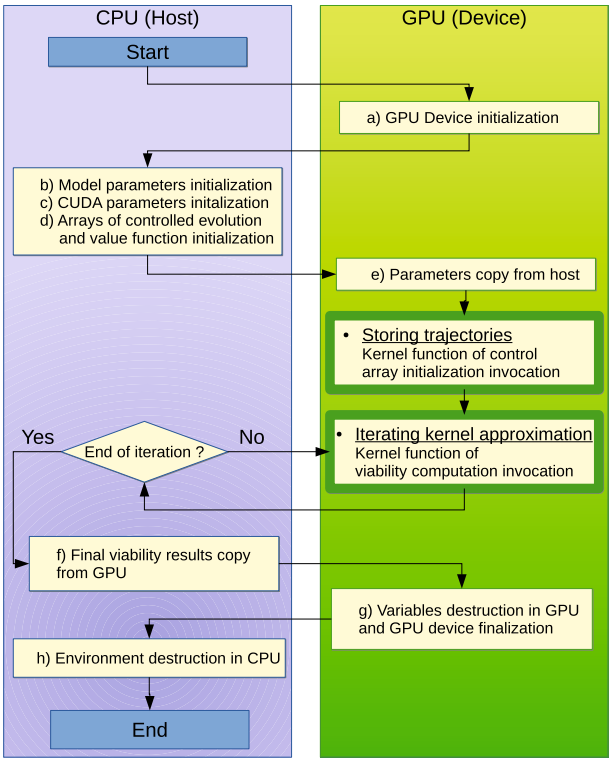

Fig. 2 Organization of the algorithm. Steps a–h are explained in Sect. 3.2. Kernel functions are detailed in Sects. 3.3 and 3.4

parallelized on the GPU and it is done in the first kernel function. Section 3.3 detailed this part.

– the iterating kernel approximation step of Algorithm 1. The value function of each point is calculated and ran until the stop condition is reached. It is parallelized in the second kernel function on the GPU. Section 3.4 detailed this part.
– the copy of results from the GPU memory to the CPU (f) using the CUDA instruction $cudaMemcpy$ with the flag $cudaMemcpyDeviceToHost$;
– The clear of the memory of the GPU and the finalization of the GPU device (g), using the CUDA instruction $cudaFree$;
– the disallocation of the CPU memory (h).

3.3 Parallelizing the computing and the storing of successors

The input of the algorithm is the number of points by dimension and each dimension bound. The algorithm computes the coordinates of all the points and their index. It can be parallelized since each point is independent from the other. Thus, one thread on the GPU will compute the coordinates of one point.

Now, we have to compute the all the successors of each point of the grid, when applying the different values of the control (if the control can take continuous values then it is discretized). We can thus suppose that the control takes $nbControls$ values

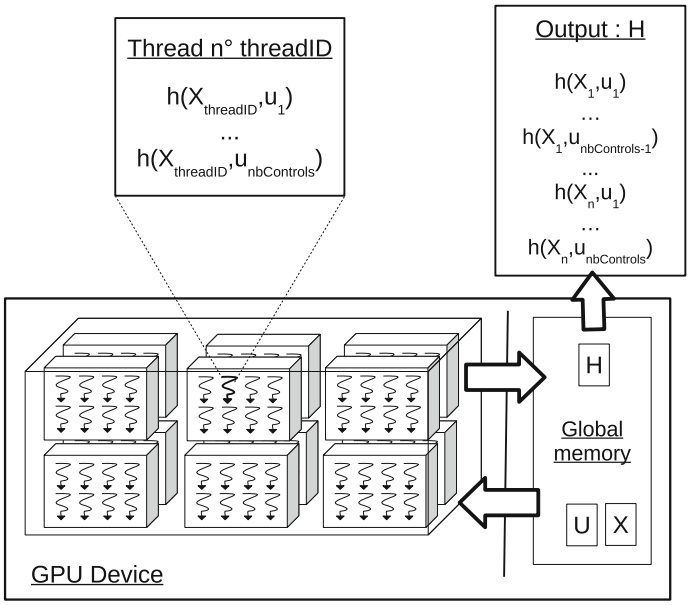

Fig. 3 Storing of trajectories: the parallelized controlled evolution computation from coordinates. The different steps of the parallelization are shown. The computation of all the controlled evolution for one point is done on one thread

point of the grid. Then, on each thread, the coordinates of one state X_i are sent, and all the $nbControls$ possible evolutions from this state are returned. Again, the number of threads is equal to the number of points of the grid. This treatment is summed up on Fig. 3. There are four steps in the GPU computation of the trajectories:

- the inputs are the values of the discretized control, $U = (u_1, \ldots, u_{nbControls})$ and the coordinates of all the points $X = (X_1, \ldots, X_{nbPoints})$. These coordinates were determined beforehand and remain in the GPU global memory;
- this data is sent on threads, each thread holding the computation of all the controlled evolutions of only one point;
- each thread then sends back the result in the array H. This array contains the index of the controlled evolutions. Moreover, it contains -1 if the evolution doesn't belong to K. Thus, we store the index of i-th controlled evolution of X in $H(X, i)$. As the number of threads is the same for each block, some threads can be superfluous on the last block;
- once we get the controlled evolutions, we just have to copy them from the GPU to the CPU and we apply the second part of the dynamic programming algorithm.

3.4 Parallelizing the iterating kernel approximation

The second part of the algorithm computes the value function V for each point until the stopping condition is reached. As shown on Fig. 4, we parallelize this task. The global memory contains the value function array: v in which the value function V of each point is stored; the successors computed at the first step: H; and a boolean value:

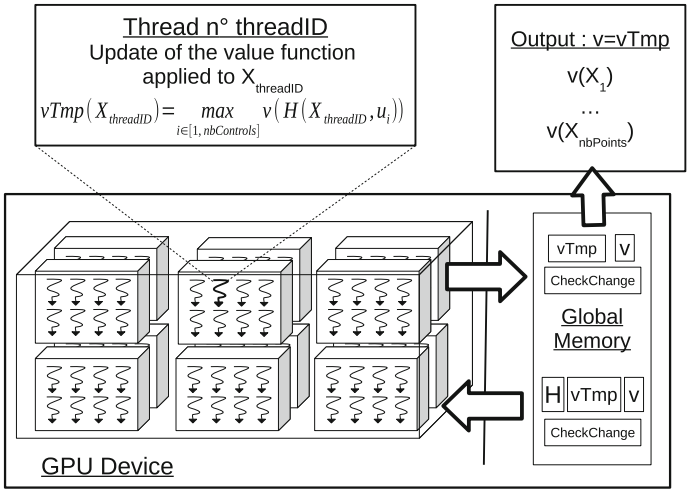

Fig. 4 Iterating kernel approximation: the parallelized viability computation. The different steps of the parallelization are shown

$CheckChange$ which states if there is no change during the step. In this latter case, the algorithm stops. The current value function V is stored in a temporary array $vTmp$. The treatment below (Algorithm 2) is applied to each thread (one point X by thread):

Algorithm 2 Update of the value function of X in the iterating kernel approximation.

If $v(X) = 1$ (We check if the point is viable, i.e., if the value function applied to it is equal to 1)
  If $\forall i \in [1, nbControls - 1]$,
  $(H(X, i) = -1)$ OR $(H(X, i) \geq 0$ AND $v(H(X, i)) = 0)$
    $vTmp(X) = 0$
    $CheckChange = CheckChange + 1$
  End If
End If

If the point is viable and if a viable successor exists, nothing is done, the point is still viable. But if none of its successors is viable, i.e. if none of its successors belongs to K (i.e., $H(X, i) = -1$) or if the value function V of the successors is equal to 0, then the point is considered non-viable: 0 is put in the appropriate cell of the array $vTmp$. This treatment represents the value function updating in dynamic programming. In this case, a change is made, then $CheckChange$ is incremented. At the end of the step, the value function is transferred of $vTmp$ to v and we check $CheckChange$. If it is equal to 0, the algorithm can stop. It happens after a sufficient number of iterations and we then get the approximation of the viability kernel.

4 Results

First, we test the GPU programming on a simple example of population dynamics, in order to compare the performance with the sequential algorithm. Then, we use

our parallel algorithm to treat a problem of bycatch fishery management up to 6 dimensions.

4.1 A multi-dimensional population model

One toy model is used here to evaluate the benefits of GPU programming. We consider the population model studied by Malthus and Verhulst. Malthusian ideas are the basis of the classical theory of population and growth (Ehrlich and Lui 1997). This model is a simple dynamical system of population growth on a bounded space without any predator. The system state includes two variables: the size of the population and its evolution rate. Aubin and Saint-Pierre (2007) introduced a control on this evolution rate and adapted it to the management of renewable resources.

4.1.1 The model

As we want to analyze the performances of the parallelized algorithm on multi-dimensional problems, we artificially increase dimensions of the system in adding some independent variables in order to study the behaviour of the program in different dimensions. The new model is as follows: we considered n states x_i representing the size of n populations, which all have the same evolution rate $y(t) \in [d, e]$. The size of the populations must remain in $K = [a, b]$. The control is applied on the derivative of the evolution rate at each time step with the inertia bound c. The equations ruling the system in discrete time are:

$$x_i(t+1) = x_i(t) + x_i(t)y(t), \quad i \in \{1, \ldots, n\} \tag{7}$$

$$y(t+1) = y(t) + u(t) \tag{8}$$

with $-c \leq u(t) \leq +c$. The set $K = [a, b]^n \times [d, e]$ is the viability constraint set. For the rest of this work, bounds for K are set as $a = 0.2$, $b = 3$, $c = 0.5$, $d = -2$, $e = 2$. By default, the control is discretized in 5 values. Since the parallelization simply breaks down processes that are already independent in the usual algorithm, the discretization does not impact the accuracy of the results.

The program is implemented in C++, using CUDA 5. The computer CPU is an Intel® Xeon(R) CPU E5-2687W 0 @ 3.10 GHz × 16 and the GPU card is a Tesla K20C with a computing capability of 3.5 and 2496 CUDA cores.

4.1.2 Speed-up

Here we show a comparison between the two versions of the program applied to this model. The speed-up S is a measure evaluating the performance of the parallelization:

$$S = \frac{T_{seq}}{T_{par}} \tag{9}$$

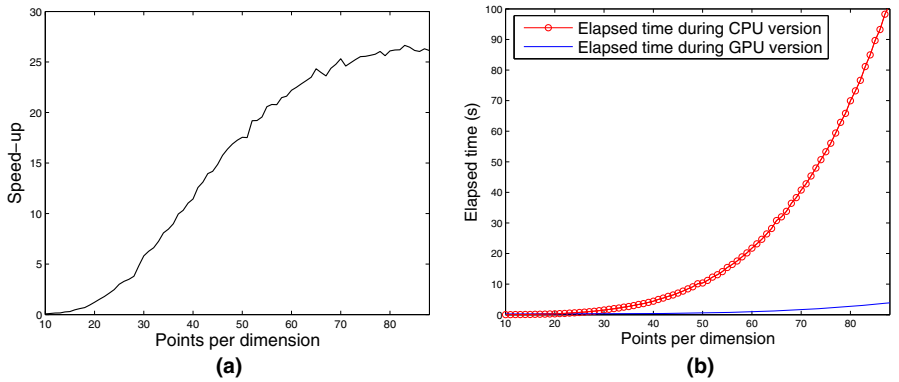

Fig. 5 Effect of grid refinement on the speed-up in a 4-dimensions problem in **a**. The elapsed time during the two versions of the algorithm are shown in **b**

where T_{seq} is the sequential algorithm execution time and T_{par} the execution time of the parallel one. The first one uses the usual algorithm; the second one uses the adapted algorithm to GPU computing.

We figure the evolution of the speed-up, in function of the size of the grid. In Fig. 5, we study the influence of grid refinement in the case of a 4-dimension problem.

First, according to Fig. 5a, we can see that the speed-up exceeds 5 in most cases of 4-dimension problems. It rises, using the whole GPU capacity but, then, the curve stagnates. The speed-up reaches a threshold when the GPU is fully loaded and where no more time gain is possible. On Fig. 5b, a comparison of elapsed times during the CPU and GPU version of the algorithm is shown. We notice that when the refinement of the grid is high, the computation time increases significantly in the CPU version.

Figure 6, shows speed-up isolines in function of the number of dimensions and the number of points by dimension. We observe that when we have a 2- or 3-dimension problem, the speed-up is between 0 and 4. This is not significantly time-saving, and the initialization of the GPU can be slower than the classic array filling. However, when the number of dimensions grows, the speed-up increases up until 20 in 7 dimensions or more in 5 or 6 dimensions with a finer discretization. The speed-up isolevels follow the total number of points of the grid for high-dimension problems (4 dimensions or more). In these problems, the main part of the time is spent in the trajectories storing part (the iterating kernel approximation ends quickly because of the low accuracy of the grid).

4.1.3 Parallelized parts

Here, we compare the speed-up during the storing of successors and during the computation one between the CPU and the GPU versions of the algorithm. In the following figure (Fig. 7), we study two different sizes of problems (2 and 6 dimensions) and we return speed-ups in function of the number of points by dimension. Elapsed times during the two parallelized parts are also returned.

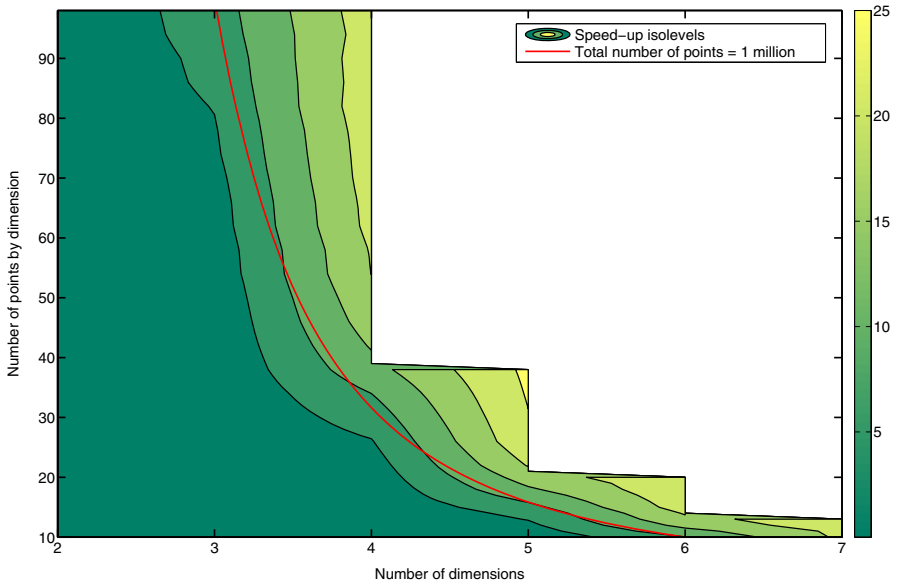

Fig. 6 Speed-up isolevels. The *red line* shows problems with same amount of points on the grid (1 million) (color figure online)

On Fig. 7a, we see that the elapsed time during the iterating kernel approximation is now higher than the one for storing the successors for 2-dimensions problems (for problems with more than 3000 points per dimension for the CPU version, or 1600 points per dimension for the GPU version). For problems in 3 dimensions or more (like 6 dimensions in Fig. 7c), storing successors time is still higher than iterating kernel approximation time. The time for storing successors exponentially increases with the number of dimensions and parallelizing this part of the algorithm is very efficient.

The speed-ups of the storing of trajectories and iterating kernel approximation are calculated separately here. As shown on Fig. 7b and d the speed-up increases up to 27 in 2 dimensions and 34 in 6 dimensions for the storing of trajectories, and up to 15 in 2 dimensions and 3.7 in 6 dimensions for the iterating kernel approximation.

For a small problem with a low discretization, the time taken by the two algorithms is negligible, but we observe that the speed-up is below one (up to 400 points by dimension in the 2D-problem, and 11 points by dimension in the 6D-problem). The mandatory initialization of the CUDA device explains the less successful results of the GPU version in smaller problems.

Moreover, we notice that the parallelization of the storing of successors is more efficient whatever the number of dimensions. Actually, memory exchanges (reading the value function array in the global memory from each thread at each step) in the second part *alter* the performance of the parallelization. Thus, the speed-up of the iterating kernel approximation falls down to 3.7 in 6 dimensions. Since the time elapsed during this step is small compared to the time elapsed during the storing of trajectories, its impact is not significant on the global speed-up.

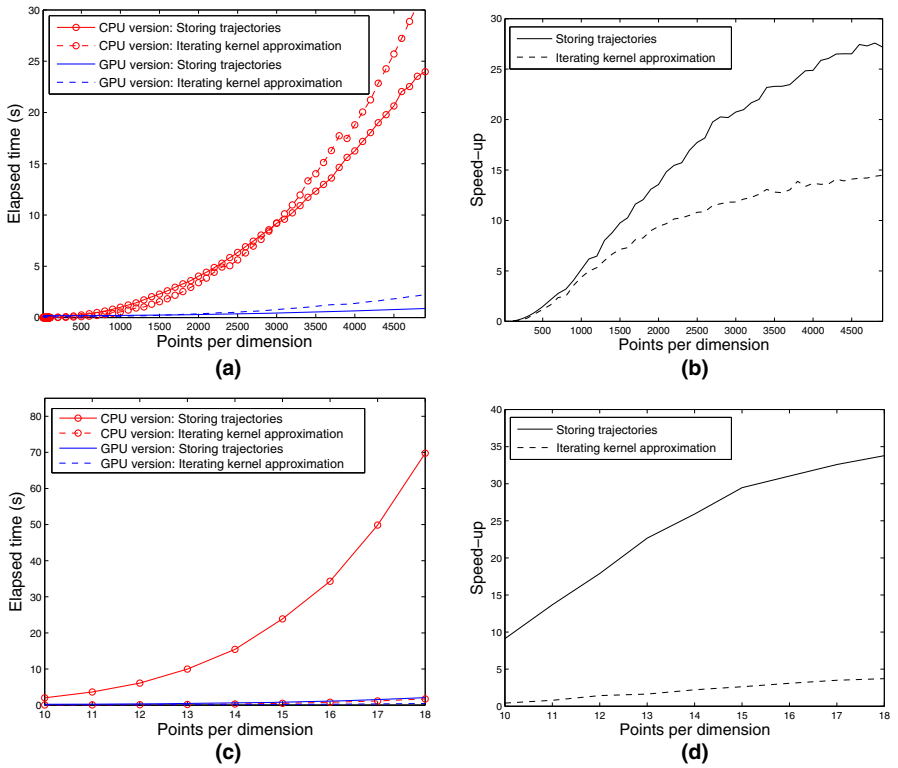

Fig. 7 Comparison of the storing of trajectories and the iterating kernel approximation elapsed times and speed-up between the GPU and CPU versions of the algorithm. Two sizes of problems are shown: 2-dimension problem (**a**, **b**), and 6-dimension problem (**c**, **d**). **a** Comparison of elapsed times (2D). **b** Speed-up of the storing of trajectories and iterating kernel approximation (2D). **c** Comparison of elapsed times (6D). **d** Speed-up of the storing of trajectories and iterating kernel approximation (6D)

4.2 Application to a bycatch fishery

We illustrate our results on a multidimensional bycatch fishery model. This model is an extension of the one studied by Krawczyk et al. (2013) from 3 dimensions to 6 dimensions. We study the influence of the number of bycatch species on the fishery management.

4.2.1 The two-species dynamics

The model of Krawczyk is reminded here. There are two ecologically independent populations of fish x and y, harvested by a single fleet. The fishery focuses on harvesting the target stock x. We define a catchability coefficient q_x determining the quantity of biomass that each unit of effort extracts, relative to the total size of the biomass at the time. Thus, the harvest rate at time t is:

$$h_x(t) = q_x e(t) x(t) \tag{10}$$

The product of q_x and $e(t)$ is generally termed the level of fishing mortality and it is expressed as a proportion of biomass.

A linear bycatch production function is used:

$$h_y(t) = \alpha h_x(t) \tag{11}$$

where the constant α, $0 < \alpha < 1$ is the bycatch to target harvest ratio.

The respective population dynamics are then described by:

$$x(t+1) = x(t) + r_x x(t)\left(1 - \frac{x(t)}{L_x}\right) - q_x e(t) x(t) \tag{12}$$

$$y(t+1) = y(t) + r_y y(t)\left(1 - \frac{y(t)}{L_y}\right) - \alpha q_x e(t) x(t) \tag{13}$$

where r_x, r_y, L_x and L_y are all positive constants. By convention, r_x and r_y represent the intrinsic growth rates, and L_x and L_y are the environment carrying capacities, of x and y, respectively. According to Krawczyk, the numerical values for the bycatch species are chosen to have a bycatch stock less productive and less valuable than the target stock.

We suppose that the effort e can be controlled: $e(t+1) = e(t) + u(t)$, $u(t) \in [u_{min}, u_{max}]$.

The speed at which the regulator can change fishing intensity is bounded by u_{min} and u_{max}. The fishing fleet's profit is given by:

$$\pi_{xy}(t) = p_x h_x(t) + p_y h_y(t) - ce(t) - C \tag{14}$$

where p_y represents the unit prices of y, p_x the unit price of x The marginal cost of the fishing effort is c and C is a fixed cost. We define the viability constraints set K:

$$K = \left\{(x, y, e, u) : x(t) \geq \frac{L_x}{10},\ y(t) \geq \frac{L_y}{10},\ \pi_{xy}(t) \geq 0,\ e(t) \in [e_{min}, e_{max}]\right\} \tag{15}$$

The safe minimum biomass levels are set to one tenth of each fish population's unexploited level, as commonly implemented in fisheries worldwide (Krawczyk et al. 2013).

4.2.2 The n-species dynamics

We propose to extend this model to treat n-species dynamics in order to illustrate the benefits of the GPU computation of the viability kernel. The dynamics of the target species remains unchanged (Eq. 2). We suppose here that we have n bycatch species y_i. Moreover, we assume that these species are at the same trophic level, competitively sharing the same food resource (Ekerhovd and Steinshamm 2014; Rice et al. 2013).

The bycatch species growth is limited by the common carrying capacity L_y. The discrete dynamics becomes:

$$y_i(t+1) = y_i(t) + r_{y_i} y_i(t)\left(1 - \frac{\sum_{k=1}^{n-1} y_k(t)}{L_y}\right) - \alpha_i q_x e(t) x(t), \quad \forall i \in [1, n-1] \tag{16}$$

r_{y_i} is the intrinsic growth rate. α_i denotes the bycatch to target harvest ratio, a measure of how highly coupled the production relationship is. Each bycatch stock y_i is a by-product of the production process. A linear bycatch production function is used:

$$h_{y_i}(t) = \alpha_i h_x(t) \tag{17}$$

We assume here that the sum of all bycatch harvest ratios α_i is equal to α defined in the previous section:

$$\sum_{i=1}^{n-1} \alpha_i(t) = \alpha \tag{18}$$

For reasons of simplicity, we suppose that all the α_i are equal:

$$\alpha_i = \frac{\alpha}{n} \tag{19}$$

As before, economic sustainability requires that the activity remains profitable. The fishing fleet's profit is given by:

$$\pi_{xy}(t) = p_x h_x(t) + \sum_{i=1}^{n-1} p_{y_i} h_{y_i}(t) - ce(t) - C \tag{20}$$

Then, we have the viability constraint set K:

$$K = \left\{ (x, y_1, \ldots, y_{n-1}, e, u) : x(t) \geq \frac{L_x}{10},\ \forall i \in [1, n-1],\right.$$
$$\left. y_i(t) \geq \frac{L_y}{10n},\ \pi_{xy}(t) \geq 0,\ e(t) \in [e_{min}, e_{max}] \right\} \tag{21}$$

In the following, we study dynamics with 1, 2, 3 or 4 bycatch species, which means dynamics with 3, 4, 5 or 6 dimensions respectively (including the fishing effort and the target species biomass).

4.2.3 Results

We use the parameters calibrated by Krawczyk et al. (2013) and summed up in the Table 1.

Table 1 Parameters of the bycatch fishery

Name	Variable	Value
Target carrying capacity	L_x	600
Target growth rate	r_x	0.4
Target unit price	p_x	4
Target catchability coefficient	q_x	0.5
Bycatch common carrying capacity	L_y	300
Bycatch growth rate	r_{y_i}	0.2
Bycatch unit price	p_{y_i}	1.9
Bycatch harvest ratio	α_{y_i}	0.2
Marginal cost	c	10
Fixed cost	C	150
Maximum effort	e_{max}	1
Minimum effort	e_{min}	0.1
Maximum effort variation	u_{max}	0.01
Minimum effort variation	u_{min}	-0.01

The parameters for all the bycatch species i are the same

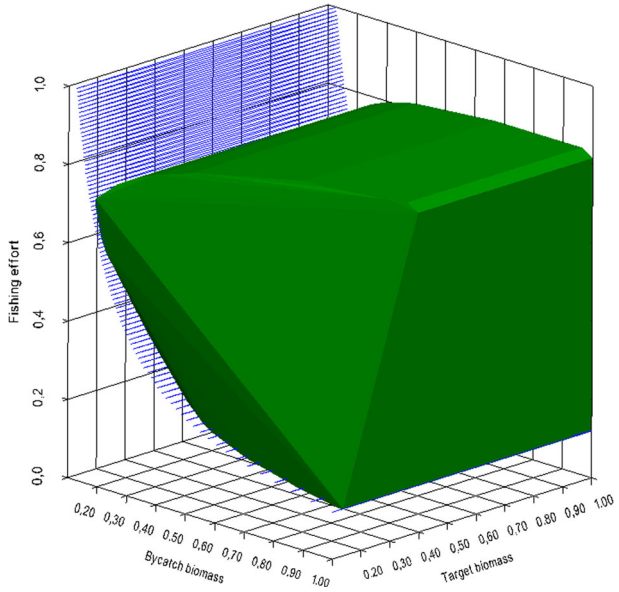

Fig. 8 Viability kernel for the 3D fishery model (with one bycatch species).Viability kernel for the 3D fishery model (with one bycatch species). *The blue surface* symbolizes the zero-profit surface. Having a positive profit (being *above the blue surface*) is a property of the viability constraint set K (color figure online)

With only one bycatch species, a similar shape of the viability kernel is obtained than the one in the previous study (Krawczyk et al. 2013). This viability kernel is presented in Fig. 8. For the following figures, the biomasses are scaled between 0.1 and 1.

The bounds of the figure are the constraint domain K. From each state contained inside the viability kernel, it exists at least a control strategy keeping the fishery in K. The states outside of the viability kernel are not viable, there is no control strategy satisfying the bio-economic constraints. For these states, the fishery is facing a "crisis" situation.

Now, bycatch species are added to the environment. For one bycatch species, 41.9 % of states are inside the viability kernel. It decreases until less than 7 % for four bycatch species.

The Fig. 9 shows 3D slices of the 4D kernel for different fishing effort values. It represents the biomass of the target species and the biomass of the bycatch species required to be viable to satisfy the ecological constraints while maintaining the profitability of the fishery. A small fishing effort needs big available stock of the target biomass in order to conserve the system profitability. A mean fishing effort leads to a large slice of the viability kernel according to the target biomass. Finally, keeping a big fishing effort requires a mean target species stock to not deplete the bycatch populations. The results are displayed for 4D problems, but they can be extended to higher dimension problems.

Finally, we show on Fig. 10 the number of states belonging to the viability kernel for a 6D problem with a grid of 20 points by dimension. The biggest viability kernel is obtained when the starting population of bycatch species are equal to $\frac{L_y}{n}$ (the offset is set by the grid step which is nearly 15.3 here). The environment is then completely filled with balanced increasing the possibility of sustainability. For larger initial populations, the environment is overloaded, causing the depletion of bycatch species. When the initial populations are smaller, the ecological sustainability requirement restricts the number of viable states.

5 Discussion and conclusion

The GPU parallelization provides significantly faster viability kernel computations and tackles problems with a higher number of dimensions or more precise discretization than a sequential algorithm.

However, a huge storage space is required to save all the coordinates and all the successors for high-dimension dynamical problems. For example, in C++, storing one coordinate in double-precision floating-point format required 8 bytes (64 bits). Then, storing the coordinates of 100 millions points in a 8-dimension problem (10 points per dimension) requires 6.4 Gb. Moreover the successors multiplied this value by the number of possible values of the control. Because of this size, it is impossible to send all the data at once. Instead, we cut the problem in smaller parts: computing the coordinates for some grid points and then computing the successors $g(x, u)$ for these points, the result is then saved. The process then continues with other grid points until all the whole grid has been processed. This reduces the need for memory.

This parallelized version may be improved with other tools (like STXXL in C++ to use large arrays) and using the mapped memory, which is based on some pointers to the RAM. Further improvements could be achieved by associating multi-CPUs and

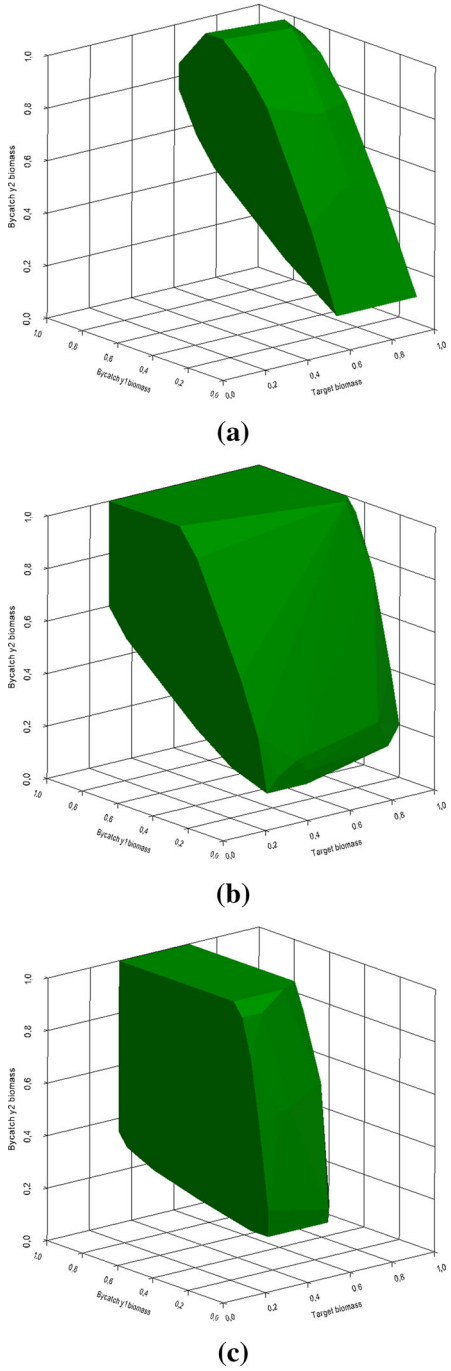

Fig. 9 3D slices of the 4D kernel for different initial values of the fishing effort e. **a** Slice through $e = 0.2$, **b** slice through $e = 0.5$, **c** slice through $e = 0.8$

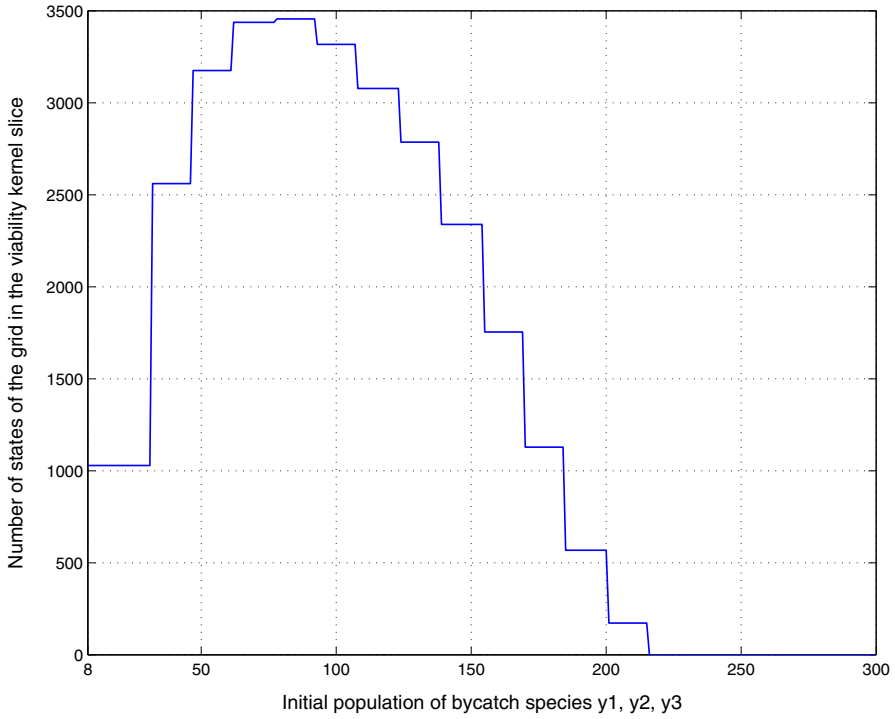

Fig. 10 Number of states of the grid belonging to the viability kernel for a 6D problem. The starting populations of three bycatch species are equal

multi-GPUs architectures, the first ones cutting the problem in small parts and the second ones solving the subproblems.

The parallelization approach proposed here opens up perspectives in terms of parallelization for other viability problems. Then, revisiting other viability approaches such as an extension to stochastic dynamics (Doyen and De Lara 2010) or algorithms such as support vector machine algorithm (Deffuant et al. 2007) could also lead to significant improvements of efficiency. Finally, the parallelization of the viability kernel algorithm provides a tool for the management of socio-ecosystems as illustrated with the management of the multi-species bycatch fishery.

Acknowledgments This work was supported by grants from Irstea and Region Auvergne. This support is gratefully acknowledged.

References

Andrés-Domenech P, Saint-Pierre P, Smala Fanokoa P, Zaccour G (2014) Sustainability of the dry forest in Androy: a viability analysis. Ecol Econ 104:33–49. doi:10.1016/j.ecolecon.2014.04.016

Aubin JP, Saint-Pierre P (2007) An introduction to viability theory and management of renewable resources. In: Advanced methods for decision making and risk management in sustainability science. Nova Science Publishers Inc, Hauppauge, pp 43–93

Béné C, Doyen L, Gabay D (2001) A viability analysis for a bio-economic model. Ecol Econ 36(3):385–396. doi:10.1016/S0921-8009(00)00261-5

Bernard C, Martin S (2013) Comparing the sustainability of different action policy possibilities: application to the issue of both household survival and forest preservation in the corridor of Fianarantsoa. Math Biosci 245(2):322–30. doi:10.1016/j.mbs.2013.08.002. http://www.ncbi.nlm.nih.gov/pubmed/23954403

Bokanowski O, Martin S, Munos R, Zidani H (2006) An anti-diffusive scheme for viability problems. Appl Numer Math 56(9):1147–1162. doi:10.1016/j.apnum.2006.03.004

Bonneuil N (2006) Computing the viability kernel in large state dimension. J Math Anal Appl 323(2):1444–1454. doi:10.1016/j.jmaa.2005.11.076

Cekmez U, Ozsiginan M, Technologies S, Air T, Academy F, Sahingoz OK (2013) Adapting the GA Approach to solve traveling salesman problems on CUDA architecture. In: 14th IEEE international symposium on computational intelligence and informatics, pp 423–428

Chapel L, Deffuant G, Martin S, Mullon C (2008) Defining yield policies in a viability approach. Ecol Model 212(1–2):10–15. doi:10.1016/j.ecolmodel.2007.10.007

Chapel L, Castelló X, Bernard C, Deffuant G, Eguíluz VM, Martin S, San Miguel M (2010) Viability and resilience of languages in competition. PLoS One 5(1):1–11. doi:10.1371/journal.pone.0008681

Deffuant G, Chapel L, Martin S (2007) Approximating viability kernels with support vector machines. IEEE Trans Autom Control 52(5):933–937. doi:10.1109/TAC.2007.895881. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4200855

Djeridane B, Lygeros J (2008) Approximate viability using quasi-random samples and a neural network classifier. Int Fed Autom Control 2:14342–14347

Donoho DL (2000) High-dimensional data analysis: the curses and blessings of dimensionality. In: Math challenges of the 21st century, pp 1–32

Doyen L, De Lara M (2010) Stochastic viability and dynamic programming. Syst Control Lett 59(10):629–634. doi:10.1016/j.sysconle.2010.07.008

Doyen L, Thébaud O, Béné C, Martinet V, Gourguet S, Bertignac M, Fifas S, Blanchard F (2012) A stochastic viability approach to ecosystem-based fisheries management. Ecol Econ 75:32–42. doi:10.1016/j.ecolecon.2012.01.005

Ehrlich I, Lui F (1997) The problem of population and growth: a review of the literature from Malthus to contemporary models of endogenous population and endogenous growth. J Econ Dyn Control 21(1):205–242. http://www.ncbi.nlm.nih.gov/pubmed/12292267

Ekerhovd NA, Steinshamm SI (2014) Optimization in the 'pelagic complex': a multi-species competition model of north east Atlantic fisheries. ICES J Mar Sci

Goldsworthy MJ (2014) A GPU-CUDA based direct simulation Monte Carlo algorithm for real gas flows. Comput Fluids 94:58–68. doi:10.1016/j.compfluid.2014.01.033

Kaynama S, Oishi M (2013) A modified Riccati transformation for decentralized computation of the viability kernel under LTI dynamics. IEEE Trans Autom Control 58(11):2878–2892. arXiv:1302.5990v1

Krawczyk JB, Pharo A, Serea OS, Sinclair S (2013) Computation of viability kernels: a case study of by-catch fisheries. Comput Manag Sci 10(4):365–396. doi:10.1007/s10287-013-0189-z

Langdon WB (2011) Debugging CUDA. In: Proceedings of the 13th annual conference companion on Genetic and evolutionary computation—GECCO '11, p 415. doi:10.1145/2001858.2002028. http://portal.acm.org/citation.cfm?doid=2001858.2002028

Maidens JN, Kaynama S, Mitchell IM, Oishi MMK, Dumont GA (2009) Lagrangian methods for approximating the viability kernel in high-dimensional systems. Automatica (2013) 49(7):2017–2029

Mametjanov A, Lowell D, Ma CC, Norris B (2012) Autotuning stencil-based computations on GPUs. In: 2012 IEEE international conference on cluster computing, pp 266–274. doi:10.1109/CLUSTER.2012.46. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6337788

Mathias JD, Bonté B, Cordonnier T, de Morogues F (2015) Using the viability theory to assess the flexibility of forest managers under ecological intensification. Environ Manag (June). doi:10.1007/s00267-015-0555-4

Mattioli J, Artiouchine K (2003) Noyau de viabilité: une contrainte globale pour la modélisation de systèmes dynamiques. Technique et Science Informatiques, pp 19–32

Rice J, Daan N, Gislason H, Pope J (2013) Does functional redundancy stabilized fish communities. ICES J Mar Sci 70(4):734–742

Rougé C, Mathias JD, Deffuant G (2013) Extending the viability theory framework of resilience to uncertain dynamics, and application to lake eutrophication. Ecol Indic 29:420–433. doi:10.1016/j.ecolind.2012.12.032

Sabo D, Barzelay O, Weiss S, Furst M (2014) Fast evaluation of a time-domain non-linear cochlear model on GPUs. J Comput Phys 265:97–112. doi:10.1016/j.jcp.2014.01.044

Saint-Pierre P (1990) Approximation of the viability kernel. Appl Math Optim 29(2):1–22

Sicard M, Perrot N, Reuillon R, Mesmoudi S, Alvarez I, Martin S (2012) A viability approach to control food processes: application to a Camembert cheese ripening process. Food Control 23(2):312–319. doi:10.1016/j.foodcont.2011.07.007

Turriff J, Broucke ME (2009) A method to construct viability kernels for nonlinear control systems. 2009 Am Control Conf 1(1):3983–3988. doi:10.1109/ACC.2009.5160409. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5160409

Chapitre 3

Utilisation d'outils fiabilistes dans la programmation dynamique

3.1 Une approximation fiabiliste de la fonction valeur utilisant moins de ressources

Le **chapitre 2** a mis en lumière le goulot d'étranglement qu'est le calcul de la matrice de transition dans l'algorithme de programmation dynamique. Cela mobilise à la fois la majeure partie de l'espace-mémoire et du temps de calcul. L'objectif de ce chapitre est d'arriver à passer outre le calcul de cette matrice dans sa totalité, spécialement dans le cas stochastique, pour lequel ce problème est exacerbé. L'algorithme de programmation dynamique pour le calcul de noyau de viabilité stochastique est présenté dans l'article de ce chapitre. Pour un état donné, sa fonction valeur calculée à chaque pas de temps nécessite le calcul de la probabilité de transition de cet état vers tous les autres de la grille. Ce calcul peut se faire avec une méthode de Monte-Carlo. Néanmoins, le nombre de dimensions augmentant, cette méthode se révèle vite inopérante. En effet, la précision du résultat dépend du nombre de trajectoires nécessaires, qui augmente exponentiellement avec la dimension. De plus, le stockage de toutes les probabilités de transition pour chaque valeur discrétisée du contrôle est un second frein à l'application directe de la programmation dynamique. Cela est d'autant plus vrai quand les probabilités de transition évoluent au cours du temps.

Cet article propose une approximation se passant du calcul de cette matrice dans sa totalité. Pour se faire, on regroupe les états de la grille par seuils de fonction valeur et on calcule la probabilité pour la dynamique d'aller d'un état de la grille vers chacun de ces ensembles. Nous effectuons ce calcul dans l'article de ce chapitre à l'aide de techniques issues de la théorie de la fiabilité comme la méthode FORM (First Order Reliability Method). L'intégration de ces méthodes dans les algorithmes de calcul de noyaux de viabilité stochastiques est un pas en avant vers la conception de programmes traitant des systèmes de haute dimension.

3.2 Présentation et contributions de l'article

Cet article est en cours de développement. On y propose un algorithme de programmation dynamique. Les principales contributions de cet article sont les suivantes :

- nous proposons un algorithme couplant programmation dynamique et méthodes fiabilistes afin de gagner en temps d'exécution et en espace mémoire. Les méthodes fiabilistes sont utilisées pour calculer les probabilités de transition ;
- nous comparons les résultats au calcul de noyau de viabilité par programmation dynamique classique. La carte des contrôles calculée par notre approximation permet de reconstruire la fonction valeur ;
- nous utilisons cet algorithme pour calculer des noyaux de viabilité stochastiques approchés dans le cadre de le gestion de systèmes environnementaux. Ainsi, nous ajoutons de l'incertitude au modèle de pêcherie déterministe présenté dans le **chapitre 2**, afin d'étudier l'impact des aléas sur la taille du noyau de viabilité. De plus, un modèle prenant en compte plusieurs niveaux trophiques dans le cadre de la gestion de l'eutrophisation illustre l'article.

3.3 Conclusions

Le couplage de la programmation dynamique et des techniques fiabilistes permet d'obtenir une approximation de la carte des contrôles optimaux ainsi qu'une approximation de la fonction valeur. L'espace-mémoire utilisé est beaucoup plus faible qu'avec l'algorithme classique de programmation dynamique, passant d'une fonction quadratique du nombre d'états de la grille à une fonction linéaire. Dans le cas présenté dans l'article (dynamique de population avec une matrice de transition évoluant dans le temps), le temps de calcul d'une borne inférieure de la fonction valeur est plus de 20 fois plus petit que le temps de calcul de l'algorithme de programmation dynamique. Ce temps de calcul dépend linéairement du nombre d'ensembles approximant la fonction valeur lors de l'exécution du programme. La borne inférieure calculée de la fonction valeur donne une sous-approximation du noyau de viabilité stochastique qui se dégrade rapidement quand l'horizon de temps augmente. En revanche, la précision de la carte des contrôles obtenue est assez bonne pour que celle-ci soit utilisée afin de reconstruire le noyau de viabilité stochastique. Pour éviter cette étape, l'utilisation d'une moyenne au lieu de la borne inférieure donne une meilleure approximation directe de la fonction valeur.

Dans cet article, nous avons utilisé la méthode FORM afin de calculer les probabilités de transition vers les différents domaines rassemblant les états de la grille

par paliers de fonction valeur. Selon le problème et la forme de la fonction valeur, d'autres méthodes plus précises d'approximation de surfaces de réponse sont utilisables, comme la méthode SORM (Second Order Reliability Method) ou la méthode RGMR (Riemanian Geometric Method for Reliability) [Lemaire 10].

3.4 Texte de l'article

Using reliability techniques in dynamic programming for solving stochastic viability problems: application to environmental dynamics

Antoine Brias [a], Jean-Denis Mathias [a], Guillaume Deffuant [a]

[a] *Irstea, UR LISC Laboratoire d'ingénierie des systèmes complexes, 9 Avenue Blaise Pascal, 63170 Aubière, France*

Abstract

This paper focuses on the problem of maintaining a controlled stochastic dynamical system in a given set of states (called the constraint set). Typically, solving this problem with dynamic programming requires discretising the state space and calculating all the probabilities of transition between each couple of discrete states for each control value. When the state space dimensionality increases, the storage space needed for these probabilities increases exponentially. Moreover, the required time for estimating these probabilities by Monte-Carlo (MC) simulations also increases exponentially with the state space dimensionality. To address this problem, this paper proposes an algorithm coupling dynamic programming and reliability methods. At each step, the algorithm approximates the value function by a given number of level sets. It uses reliability techniques that are efficient for computing the probability of transition from one state to a given level set. In this algorithm, the required memory is linear with the number of states, instead of being quadratic in standard dynamic programming. Moreover, we show that the control map derived from the level set value function is very close to the one obtained with the complete dynamic programming. This control map may be used to construct the stochastic viability kernel. We test the approach on three models, one of them in 5 dimensional state space.

Key words: Viability; Stochastic dynamics; Dynamic programming; Level-set methods; First-order reliability methods; High-dimensional systems

Email addresses: antoine.brias@irstea.fr (Antoine Brias), jean-denis.mathias@irstea.fr (Jean-Denis Mathias), guillaume.deffuant@irstea.fr (Guillaume Deffuant).

1 Introduction

Viability theory has been developed in 90s to address the problem of maintaining dynamical systems inside a set of admissible states. It has been applied to renewable resources management such as fisheries [16,4,15,10,18,24,27] or forest conservation [5,25] and others [11]. Viability theory looks for *viable states* from which there exists at least a control function that keeps the system in the constraint domain. The set of all viable states is called the *viability kernel*. Several algorithms compute numerical approximations of viability kernels in the case of deterministic dynamics [32,6,21].

This concept was extended to address stochastic dynamics [17], paving the way for other applications in uncertain dynamics [30,16]. The stochastic analogue of the viability kernel, called the stochastic viability kernel, contains all the initial states of the system, for which a control strategy exists, keeping the system in the constraint set, until a given time horizon, with a given probability.

We focus here in the resolution of the stochastic problem, looking for this control strategy maximizing the probability to keep the system inside the constraint set, for each initial state of this set. This problem may be solved by using dynamic programming [17]. With several backward iterations, it computes a value for each state of a grid (the value function) which is then used for choosing the control at each step. This method is efficient but costly, mainly because it requires computing the transition matrices from each state to each other is required. These transitions are straightforward in deterministic dynamics but their computation already consumes most of the of the time and space resources [8]. This problem is exacerbated with stochastic dynamics because the evolution is not unique anymore and depends on the uncertainties. The transition matrices are classically estimated with Monte-Carlo (MC) methods. This approach requires to store an amount of data exponentially growing with the state space dimensionality and the time needed by MC methods also dramatically increases with the dimensionality.

Time-variant reliability and stochastic viability theory have recently been connected [31], opening the cross-fertilization between these fields. In this line, this paper proposes an algorithm incorporating the tools developed in reliability to improve or replace the MC methods by less intensive methods [26], within dynamic programming.

Our approach proceeds with an approximate value iteration [28]. The approximation is made with a given number of level sets, attributing the same value to sets of states. As a consequence, the algorithm computes state-to-set transitions, instead of state-to-state transitions by using reliability techniques, such

as FORM/SORM methods ([9,13]). Then the value function is again approximated by level sets, for the next iteration. Finally, the method provides an approximate value function from which a control map can be derived, as well as an approximate stochastic viability kernel.

This article is organized as follows: in the next section, we recall the details of the dimensionality curse in dynamic programming. From there, we develop a new algorithm yielding lower and upper bounds of the value function used in the dynamic programming algorithm, and using reliability method. In section 3, we present the results on the error and the complexity of our approximation. Section 4 shows some applications of this work on fishery and lake management. Section 5 gives some discussions about our approximation.

2 Stochastic viability : the curse of dimensionality

2.1 Dynamic programming

We consider a bounded discrete state space $X \subset \mathbb{Z}^d$ (where $\mathbb{Z}$ is the set of all integers, d is the dimensionality of the space) and a dynamical system of state $x(t)$ evolving in X. The evolution of the system can be modified at each time step by choosing the control $u(t, x(t))$ in discrete set U according to the current state $x(t)$. More precisely:

$$\begin{cases} x(t+1) = f\left(x(t), u(t, x(t)), w(t)\right) \\ x\left(0\right) = x_0 \end{cases} \tag{1}$$

where $w(t)$ is drawn from a random variable $W(t)$ and introduces random modifications of the system's trajectory; it models the uncertainty. In general, this discrete dynamical is derived from a continuous dynamics with an adequate discretization scheme.

A viability problem is finding a control function that maximizes the probability to maintain the system inside a constraint set $K \subset X$. This problem can typically be solved using dynamic programming. For a given finite horizon T, dynamics programming computes a value function $V(t, x)$ for each state $x \in K$ and for each $t \in \{0, 1, .., T\}$. Then, starting from any state x_0 of K, choosing the control $u(t, x(t))$ such that the expectation of $V\left(t+1, f(x(t), u(t, x(t)), w(t)\right)$ is the highest. This control strategy maximizes the probability to maintain the system within K longer than T. The value function is computed iteratively as follows:

$$\begin{cases} V(T,x) = \begin{cases} 1 & if\ x \in K \\ 0 & if\ x \notin K \end{cases} \\ \forall t \in \{0, T-1\}, \forall x \in K,\ V(t,x) = \max\limits_{u \in U(t)} \sum\limits_{y \in K} \mathbb{P}\left(f\left(x, u(t,x), w\left(t\right)\right) = y\right) V\left(t+1, y\right) \end{cases} \tag{2}$$

Therefore, computing the value function requires computing $\mathbb{P}\left(f\left(x, u(t,x), w\left(t\right)\right) = y\right)$ the transition probabilities between two states, for each choice of control u.

2.2 Curse of dimensionnality

Let matrix $\mathcal{M}_{x,y,u}\left(t\right)$ be the probabilities of transition between two states x and y of K for each discretized control u:

$$\mathcal{M}_{x,y,u}\left(t\right) = \mathbb{P}\left(f\left(x, u\left(t,x\right), w\left(t\right)\right) = y\right) \tag{3}$$

The computing cost of this matrix exponentially increases with the number of dimensions of the problem. Indeed, if the discrete set $\hat{K}$ contains N^d points (i.e. N points by dimension) and if the control is discretized in N_u values, dynamic programming requires computing and storing $N^d \times N^d \times N_u$ values for the transition matrix $\mathcal{M}_{x,y,u}\left(t\right)$ at each step of time, plus N^d values for the value function $V\left(t,x\right)$. Even in deterministic problems, the most important part in computation time is the calculation of the transition matrix ([8]). In the stochastic case, the transition probabilities are usually estimated with MC simulations, i.e. by drawing a large number of trajectories starting from each state and then computing the frequency of reaching each other state as estimation of the transition probability. If we note T_{MC} the time required for estimating the transitions starting from one state with MC simulations, the total time T_{DP} for dynamic programming is approximated by :

$$T_{DP} \approx T \times N^d \times N_u \times T_{MC} \tag{4}$$

Therefore, in practice, dynamic programming can only solve problems in low dimensionality state spaces. This limitation is often called the "curse of dimensionality", and can be found in other domains of computational sciences such as machine learning or data mining [3,14,23]. We propose in this paper to overcome this limitation by using a level set approximation of the value function and by using an approximation for the computation of transition probabilities.

3 Computing transition probabilities with reliability method

The field of reliability developed efficient alternatives to MC simulations [19] for estimating the probability that an uncertain system lies in a failure domain. We propose to integrate these methods into dynamic programming.

3.1 Level set approximate value function

The first step for using reliability methods is to compute the transition probability to larger domains than the transition probability to a single state. Hence we define n sets K_i^t gathering states x such that their value function $V(t,x)$, lies in a given interval $[\alpha_i^t, \alpha_{i+1}^t]$, where $0 = \alpha_0^t < \alpha_1^t < \ldots < \alpha_{n-1}^t < \alpha_n^t = 1$ define n intervals covering $[0,1]$:

$$\forall i \in \{0,..,n-1\}, \forall t \in \{0,..,T\}, \; x \in K_i^t \Longleftrightarrow \alpha_i^t \leq V(t,x) < \alpha_{i+1}^t \qquad (5)$$

Three approximate value functions can be based on these sets, the upper, lower and middle approximations, respectively associating to each x state in K_i^t:

- the upper bound of the interval: $V_h(t,x) = v_h(t,i) = \alpha_{i+1}^t$;
- the lower bound of the interval: $V_l(t,x) = v_l(t,i) = \alpha_i^t$;
- the middle of the interval: $V_m(t,x) = v_m(t,i) = (\alpha_i^t + \alpha_{i+1}^t)/2$;

By denoting V_a one of the approximate functions V_h, V_l or V_m, for each set K_i^t, the value associated to each state x is noted v_a, only depending on the time step and the index of the set K_i^t. The computing process of the approximate value function is similar to the standard procedure of dynamic programming, except that the value function $V(t+1,x)$ in Eq. 2 is replaced by its approximation $V_a(t+1,x)$, and the transitions are computed from one state to one set K_i^t at each time step, instead of being computed from one state x to one state y. The value function $\tilde{V}_a(t,x)$ is then approximated using n level sets K_i^t.

The level set function is then defined by a procedure denoted $A_a(\tilde{V})$ applied to the values $\tilde{V}_a(t,x)$ (this procedure determines the values of α_i as described in details further):

$$V_a(t,.) = A_a\left(\tilde{V}_a(t,x), x \in K\right). \qquad (6)$$

More precisely, for time T, the value function is initialized as in the standard dynamic programming:

$$V_a(T,x) = \begin{cases} 1 & if\ x \in K \\ 0 & if\ x \notin K \end{cases} \tag{7}$$

Then, there is a particular case for $T-1$, where the value $\tilde{V}_a(T-1,x)$ is actually also the same as $V(T-1,x)$ in standard dynamic programming:

$$\tilde{V}_a(T-1,x) = \max_{u \in U(t)} \mathbb{P}\left(f\left(x,u\left(T,x\right),w\left(T\right)\right) \in K\right), \forall x \in K, \tag{8}$$

Then this value function is approximated by a level set function:

$$V_a(T-1,.) = A_a\left(\tilde{V}_a(T-1,x), x \in K\right). \tag{9}$$

Then, for the time steps $t \in \{0,..,T-2\}$, the intermediate value function $\tilde{V}_a$ is based on the probabilities to reach the sets K_i^{t+1}:

$$\tilde{V}_a\left(t,x\right) = \max_{u \in U(t)} \sum_{i=0}^{n-1} \mathbb{P}\left(f\left(x,u\left(t,x\right),w\left(t\right)\right) \in K_i^{t+1}\right) v_a\left(t+1,i\right), \forall x \in K, \tag{10}$$

Again, at each iteration, the intermediate value function is approximated by a level set function:

$$V_a(t,.) = A_a\left(\tilde{V}_a(t,x), x \in K\right). \tag{11}$$

This algorithm provides an approximate value function $V_a(t,x)$, and calculates state-to-sets transitions $\mathbb{P}\left(f\left(x,u\left(t,x\right),w\left(t\right)\right) \in K_i^{t+1}\right)$ instead of state-to-state transitions $\mathbb{P}\left(f\left(x,u\left(t,x\right),w\left(t\right)\right) = y\right)$. These transitions are calculable by using Monte-Carlo methods. We make the choice of using reliability methods such as FORM (see section 3.2)

3.1.1 Choosing values of α_i^t

This section describes the procedure $A_a(\tilde{V})$ for choosing values of α_i^t. The intermediate value function $\tilde{V}_a\left(t,x\right)$ is approximated by the level set function $V_a\left(t,x\right)$, with a given number n of sets K_i^t, determined by the values α_i^t. The selection of α_i^t is made by minimizing the approximation error

$E(\alpha_0^t, \alpha_1^t, .., \alpha_{n-1}^t, \alpha_n^t)$:

$$\min_{0=\alpha_0^t<\alpha_1^t<...<\alpha_{n-1}^t<\alpha_n^t=1} E(\alpha_0^t, \alpha_1^t, .., \alpha_{n-1}^t, \alpha_n^t) \tag{12}$$

in which

$$E(\alpha_0^t, \alpha_1^t, .., \alpha_{n-1}^t, \alpha_n^t) = \sum_{x \in K} |V_a(t,x) - \tilde{V}_a(t,x)|. \tag{13}$$

This minimization is performed with standard optimization tools.

3.1.2 The lower and upper approximations bounding the value function

The value function $V(t,x)$ given by standard dynamic programming is always between $V_l(t,x)$ and $V_h(t,x)$:

$$\forall x \in K, \forall t \in \{0, .., T-1\}, V_l(t,x) \leq V(t,x) \leq V_h(t,x); \tag{14}$$

Indeed, by construction, for any x we have:

$$V_l(T-1,x) \leq V(T-1,x) \leq V_h(T-1,x) \tag{15}$$

Now, suppose that, for $t \in \{0, .., T-2\}$ for any $x \in K$:

$$V_l(t+1,x) \leq V(t+1,x) \leq V_h(t+1,x) \tag{16}$$

Thus, we have:

$$\tilde{V}_a(t,x) = \max_{u(t) \in U(t)} \sum_{i=0}^{n-1} \sum_{y \in K_i^{t+1}} V_a(t+1,y) \mathbb{P}\left(f\left(x, u\left(t,x\right), w\left(t\right)\right) = y\right) \tag{17}$$

Because we supposed $V_l(t+1,y) \leq V(t+1,y) \leq V_h(t+1,y)$, and considering the computation of $V(t,x)$ (eq 2) we get:

$$\tilde{V}_l(t,x) \leq V(t,x) \leq \tilde{V}_h(t,x) \tag{18}$$

which leads to:

$$V_l(t,x) \leq V(t,x) \leq V_h(t,x) \tag{19}$$

because by construction $V_l(t,x) \leq \tilde{V}_l(t,x)$ and $V_h(t,x) \geq \tilde{V}_h(t,x)$, for any $x \in K$.

3.2 Approximate probabilities of transition

3.2.1 Statement of the problem

There are two strategies for computing the probability for the system to belong to the set K_i^t in order to provide an approximation of the intermediate value function $\tilde{V}_a(t,x)$ by the procedure $A_a(\tilde{V})$ (Fig.1):

- Monte-Carlo methods generate samples to compute the probability to belong to the set K_i^t. However, these samples are not necessarily close to the zone of interest which is the zone with the highest probability of failure. It is quickly ineffective when the number of dimensions increases.
- approximate the response surface by using a metamodel. These methods, coming from the field of system reliability, are more efficient in high dimension than Monte-Carlo methods. Several meta-models may be used, more or less accurate. We make the choice here to use the first-order reliability method.

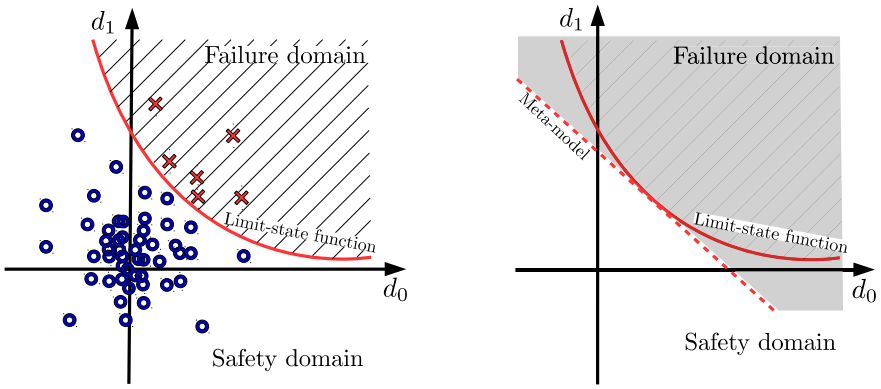

Figure 1. Illustration of Monte-Carlo methods (left) and reliability analysis methods using (right) for computing transition probability in the reduced centered space (d_0, d_1). Monte-Carlo methods generate samples according the probability distribution over the domain. The probability for the system to come into the failure is given by the proportion of samples over the limit-state function (red crosses). An approximation of this probability can also be obtained by using a meta-model of the limit-state function, computing the probability for the system to reach the gray domain (see subsection 3.2.2).

3.2.2 First-order reliability method

The first-order reliability method (FORM) focuses on the area where the probability density is the highest. This method approximates the limit-state surface by a hyperplane in the standard normal space to find the *design point*, *i.e.* the point of the limit-state function with the maximal failure probability density [29].

In our case, we use FORM to compute the probability for $f\left(x, u\left(t,x\right), w\left(t\right)\right)$ to belong to the set K_i^t for each i. The first stage of the method is the Rosenblatt and Nattaf transform [12]: we move into reduced centered space by transforming the coordinates to have density probability of mean 0 and variance 1. Therefore, we solve the problem of finding the design point, the closest point of a given set boundary to the origin by using an optimization solver. In our case, we look for the point belonging to the set K_i^t, whose the distance β to the origin is minimal. This distance β from the origin to the design point is called the *reliability index* and the probability of failure, the probability to belong to the given set is $\phi\left(-\beta\right)$ (with ϕ the normal cumulative distribution function) if the boundary is a hypersurface. Indeed, the probability of failure is equal to the integral of the density function. In the case of an hypersurface, the integral of the density function is equal to 1 in tangential directions and only depends on the radial direction. A demonstration of this result is shown in Appendix 1.

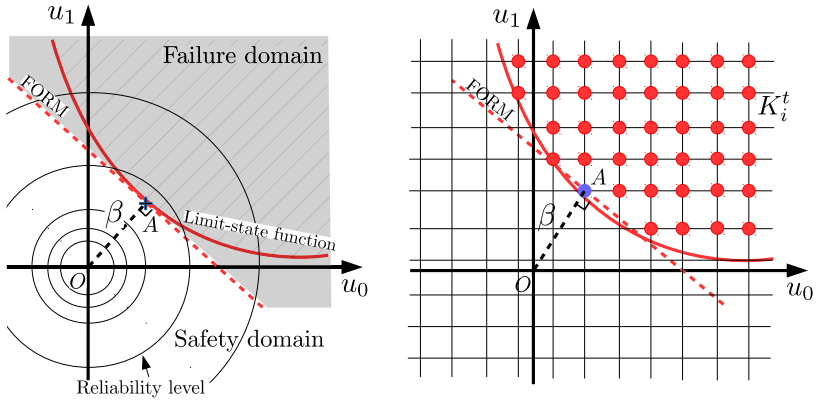

Figure 2. FORM illustration (left). In the reduced centered space, the probability distribution depends only on its norm (circular reliability levels). By replacing the limit-state function (and the hatched failure domain) by a tangent hyper-plane at the design point A (and the gray failure domain), FORM approximates the probability of failure. Our algorithm looks for the point A (blue point) of the grid belonging to K_i^t minimizing the distance to the origin in the reduced center space, among all the points of the grid belonging to K_i^t (red points).

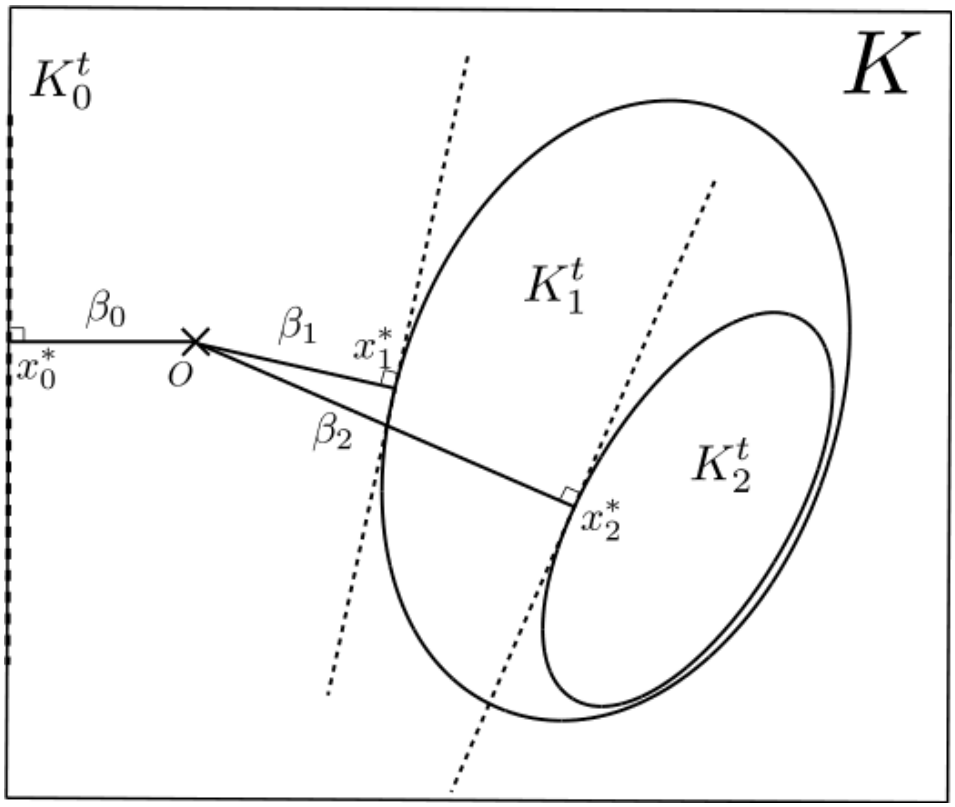

Figure 3. FORM method applied to the computation of stochastic viability kernel. The computation of the distance β from the origin O to the design point x_i^* in the reduced centered space leads to the probability to belong to the set K_i^t.

3.2.3 Multi-FORM

Therefore, by approximating locally K_i^t external boundary and internal boundary as hyperplanes, we get the probability for $f\left(x, u\left(t,x\right), w\left(t\right)\right)$ to belong to the set K_i^t by :

$$\mathbb{P}\left(f\left(x, u\left(t,x\right), w\left(t\right)\right) \in K_i^t\right) = \phi\left(-\beta_i\right) - \phi\left(-\beta_{i+1}\right) \tag{20}$$

in which β_{i+1} (resp. β_i) is the reliability index associated to the internal boundary of the set K_i^t (resp. the external boundary). Since the internal boundary of a set K_i^t is the external border of the set K_{i+1}^t, we obtain the probability to belong in all the set K_i^t at a time step t with n FORM computations for a state x of the grid. In the particular case of $i = 0$ and if the transition matrix is not time-dependent, we compute and store $\mathbb{P}\left(f\left(x, u\left(t,x\right), w\left(t\right)\right) \in K\right)$ at the beginning in order to compute $\mathbb{P}\left(f\left(x, u\left(t,x\right), w\left(t\right)\right) \in K_0^t\right)$ at each step of time.

3.3 Algorithm

Algorithm.1 provides the pseudo-code of the procedure computing the value function approximation by level sets, using FORM. The initialization step computes $V_a(T-1,x)$ which can be done by FORM calls evaluating the transition probability from x to K when applying control u:

$$\tilde{V}_a(T-1,x) = \max_u \text{FORM}(x,u,w,K) \tag{21}$$

If the constraint domain K is a convex polyhedron, this computation can be performed quickly and accurately by FORM. The procedure FORM (x,u,K_i^t) evaluates $\mathbb{P}(f(x,u(t,x),w(t)) \in K_i^t)$ by the method described in Section **??**. The procedure ALPHA$(\tilde{V}_a(t,.),n)$ determines the n α_i^t minimizing the error between $\tilde{V}_a(t,.)$ and its approximation $V_a(t,.)$ by level sets defined with α_i^t as seen in Section 3.1.1.

Algorithm 1 Detailed algorithm: approximation of value function by level sets using FORM

1: Choose n (Affects approximation precision)
2: **for** each $x \in \hat{K}$ **do**
3: $\quad \tilde{V}_a(T-1,x) \leftarrow \max_u \text{FORM}(x,u,K)$ (Initialization)
4: **end for**
5: $\left\{\alpha_0^{T-1}, \alpha_1^{T-1}, \ldots, \alpha_n^{T-1}\right\} \leftarrow \text{ALPHA}\left(\tilde{V}_a(T-1,.),n\right)$ (Optimization procedure for choosing α_i^t)
6: **for** $t = T-2 \rightarrow 1$ **do**
7: $\quad$ **for** each $x \in K$ **do**
8: $\quad\quad \tilde{V}_a(x,t) \leftarrow \max_u \sum_{i=1}^n v_i^{t+1}\left[\text{FORM}\left(x,u,K_i^{t+1}\right)\right]$ (Iterative kernel loop)
9: $\quad$ **end for**
10: $\quad \{\alpha_0^t, \alpha_1^t, \ldots, \alpha_n^t\} \leftarrow \text{ALPHA}\left(\tilde{V}_a(t,.),n\right)$
11: **end for**

3.4 Algorithmic analysis

3.4.1 Required memory

The main advantage of this approximation is that it decreases the required memory space. Particularly, instead of storing all the $\mathbb{P}(f(x,u(t,x),w(t)) = y)$ for each couple (x,y) of the grid and the current value function for each time step, we mainly store the current value function and the probability of belonging to the constraint domain K computed during the initialization (for time

independent transition matrices). The approximation requires the storing of $N^d \times N_u$ values for the probability of belonging to the constraint domain K instead of $N^d \times N^d \times N_u$ (see section 2.2), plus N^d values for the value function $V_a(t, x)$. Overall, the required memory space is divided by the number of points of the grid, passing from a quadratic to a linear function of the number of states.

3.4.2 Temporal complexity

The minimization procedure used by FORM, as well as the optimization of the choice of alpha and every time step slow down the procedure in the case where transitions do not change over time, when the number of dimension d is small. However, if state transitions depend on time or if the dimensionality increase, the algorithm becomes more effective. The temporal complexity T_a of the algorithm is :

$$T_a \approx T \times N^d \times N_u \times n \times T_{FORM} \tag{22}$$

where T_{FORM} is the time required by a FORM call. The temporal complexity depends on n, the number of sets K_i^t. Therefore, with a constant transition matrix, the approximation method is faster than the dynamic programming if $n \times T_{FORM}$ is lower than T_{MC}. As the performance of MC methods decreases with the number of dimensions, the approximation is expected to be faster in high dimensions. However, if the transition matrix evolves in time, the initialization step of the dynamic programming algorithm has to be repeated at each step of time.

4 Applications

We illustrate our work with three applications. The first one is a classical example used in viability theory. The second one is a model for which the computation of the stochastic viability kernel is feasible with dynamic programming. The last model is an example of a problem in 5 dimensions with uncertainties on each dimension.

4.1 Population model

4.1.1 Statement of the problem

In this part, we use a population dynamics to compare the results given by our approximations and the results given by the dynamic programming algorithm:

$$\begin{cases} x_j(t+1) & = x_j(t) + x_j(t)\, x_d(t) + \varepsilon_1(t)\,,\ j \in \{1, ..., d-1\} \\ x_d(t+1) & = x_d(t) + u(t,x) \end{cases} \tag{23}$$

The disturbance ε_1 are modeled by a normal distribution with mean 0 and standard deviation σ. The constraint domain is $K = [0.2, 3]^{d-1} \times [-2, 2]$. We use a grid $\hat{K}$ of 20×20 states. The time horizon is $T = 50$. For the remainder of this section, we will use 5 α_i^t by step of time ($0 \leq \alpha_i^t \leq 1$). We assume here that the control is bounded : $u_{min} \leq u(t,x) \leq u_{max}$. We use : $u_{min} = -0.5$ and $u_{max} = 0.5$ and by default, the control is discretized in 5 values.

We use the root-mean-square error (RMSE) defined as:

$$E_a = \sqrt{\frac{\sum\limits_{x \in \hat{K}} \left(u_{DP}(T,x) - u_a(T,x)\right)^2}{N^d}} \tag{24}$$

comparing the optimal control $u_{DP}(T,x)$ computed by the dynamic programming algorithm and the one computed by our approximation value function $u_a(T,x)$ for each point x of the grid $\hat{K}$ at the time horizon T.

4.1.2 Results

The accuracy is compared on the 2D population model with $\sigma = 0.1$. The control map obtained by the lower bound approximation is the closest one to the control map computed by dynamic programming according the RMSE (see Fig.4). The use of the upper bound or a middle approximation creates false positives points, for which some control strategies seem appropriate, whereas they are not.

An example of time benchmark is shown in Fig.5. It highlights that the approximation is well-suited for smaller time horizon if the transition matrix remains constant, for instance if the uncertainties are not time-dependent (with disturbance ε_1). However, if the transition matrix evolves in time, the initialization step of the dynamic programming algorithm has to be repeated at each step of time. We replace the disturbance ε_1 by the disturbance $\varepsilon_2(t)$ following a

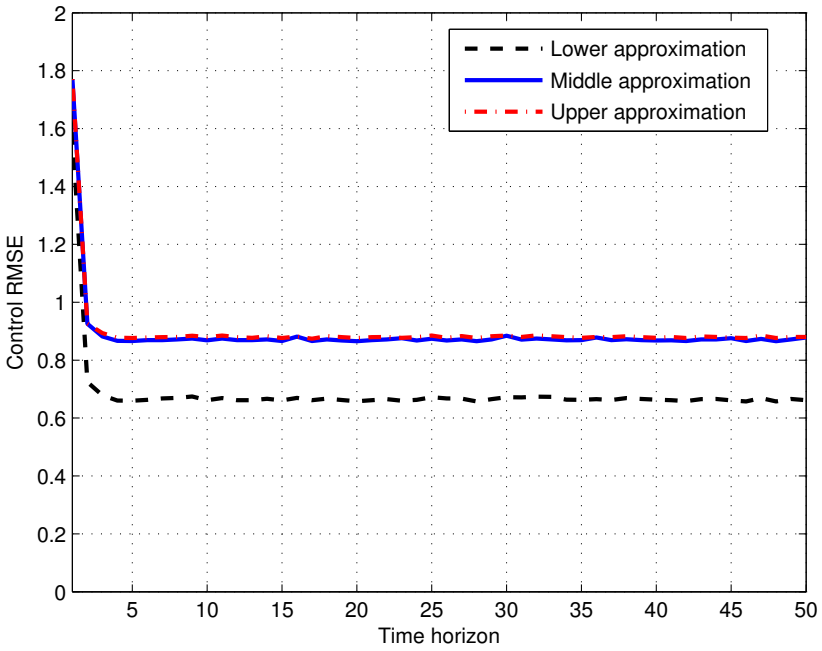

Figure 4. Root-mean-square error between the controls maximizing the probability of belonging to K until the time horizon T computed by dynamic programming and these controls computed by the approximations.

normal distribution with mean 0 and standard deviation $\sigma(t) = 0.1 + \frac{t}{100}$ to obtain the right figure of Fig.5.

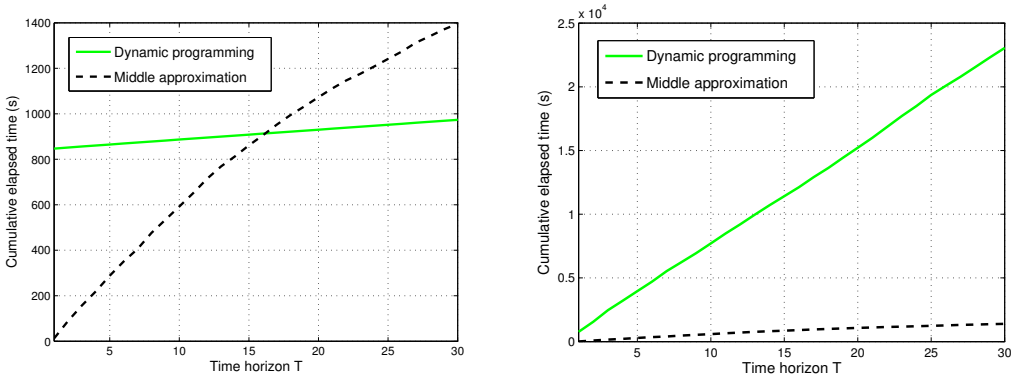

Figure 5. Cumulative elapsed time according time horizon T for a 3D population problem. On the left : the dynamic programming algorithm (green line) needs the computation of the whole transition matrix before starting the computation of the value function. Our approximation (dotted line) does not require this step but it needs FORM calls at each step of time. On the right : the dynamic programming algorithm (green line) needs the computation of the whole transition matrix at each step of time, since this one evolves in time.

In this case, our approximation is faster than the dynamic programming algorithm, regardless of the time horizon, since the FORM method for computing state-to-sets probability transitions requires less time than the MC method for computing the whole transition matrix.

Fig.6 shows that the impact of increasing the number of sets K_i^t is negligible. In this example, the uncertainties are small and increasing the number of sets K_i^t is unnecessary and does not provide enough additional information. Most of these sets are nearly empty.

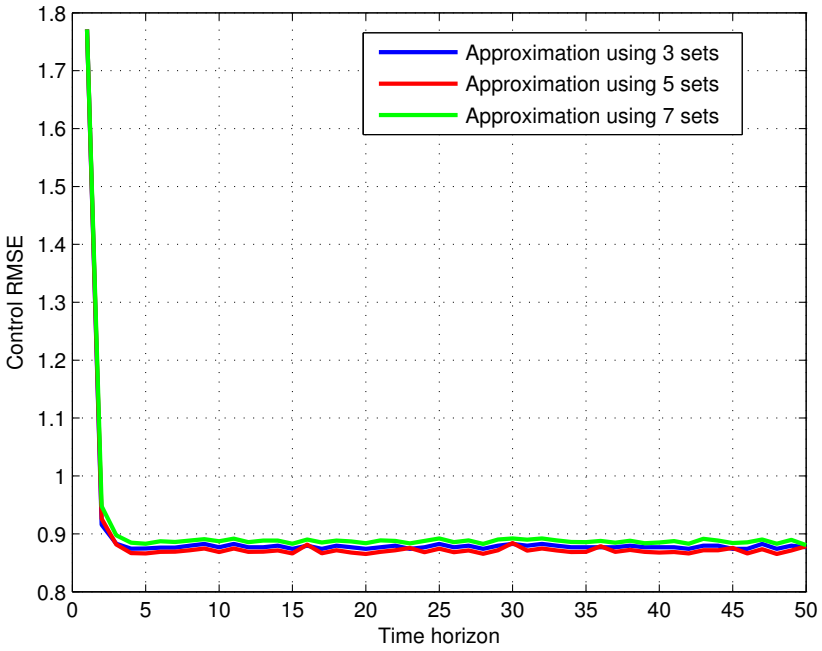

Figure 6. RMSE between the middle approximation and the dynamic programming in regard of the number of sets K_i^t. Increasing the number of sets used during the calculation does not improve the accuracy of the approximation, but it slows down the computation.

4.2 Application to bycatch fishery management

4.2.1 Bycatch fishery dynamics

We illustrate our approximation on a bycatch fishery model. In this section, we use the control map computed by our approximation in order to approximate the stochastic viability kernel. This set gathers all the states x_0, for which a control strategy exists to maintain the system in K with a probability higher than *beta* until a time horizon T:

$$Viab_K(\beta, T) = \{x_0 \in K, \exists u(\cdot) \,|\, \mathbb{P}(\forall t \in [0, T],\ f(x(t), u(t), w(t)) \in K) \geq \beta\} \tag{25}$$

According to [17], a state of the constraint domain belongs to this set if its value function at T is over β. Starting from an initial state of the grid, we

generate random trajectories following the control map and we count the ratio of these trajectories which are not leaving the constraint domain until the time horizon T. It leads to the probability of remaining in the constraint domain starting from an initial state and following the control map computed during our approximation. By selecting the states of the grid for which the probability is over *beta*, we get the stochastic viability kernel.

We incorporate stochasticity in the model studied by [22] and [8]. Two ecologically independent fish populations x and y coexist and they are harvested by a single fleet. The fishery focuses on harvesting the target stock x. The catchability coefficient q_x accounts for the quantity of biomass that each unit of effort extracts, relative to the biomass amount. The harvest rate at time t is:

$$h_x(t) = q_x e(t) x(t) \tag{26}$$

The level of fishing mortality is the product of q_x and $e(t)$ expressed as a proportion of biomass. A linear bycatch production function is used :

$$h_y(t) = \alpha_y h_x(t) \tag{27}$$

where the constant α_y, $0 < \alpha_y < 1$ is the bycatch to target harvest ratio. The population dynamics are described by :

$$x(t+1) = x(t) + r_x x(t) \left(1 - \frac{x(t)}{L_x}\right) - q_x e(t) x(t) + \varepsilon_x(t) \tag{28}$$

$$y(t+1) = y(t) + r_y y(t) \left(1 - \frac{y(t)}{L_y}\right) - \alpha_y q_x e(t) x(t) + \varepsilon_y(t) \tag{29}$$

where r_x , r_y, L_x and L_y are all positive constants. By convention, r_x and r_y represent the intrinsic growth rates, and L_x and L_y are the environment carrying capacities, of x and y, respectively. According to Krawczyk, the numerical values for the bycatch species are chosen to have a bycatch stock less productive and less valuable than the target stock. Uncertainties are added to these equations by $\varepsilon_x(t)$ and $\varepsilon_y(t)$. $\varepsilon_x(t)$ follows a normal distribution $\mathcal{N}(0, 20)$ and $\varepsilon_y(t)$ follows a normal distribution $\mathcal{N}(0, 10)$. We do not take into account cases where uncertainty would make biomasses negative or above the carrying capacities.

We suppose that the effort e can be controlled even if there is a potential variability on it:

$$e(t+1) = e(t) + u(t) + \varepsilon_e(t),\ u(t) \in [u_{min}, u_{max}] \tag{30}$$

The effort uncertainty $\varepsilon_e(t)$ follow a normal distribution $\mathcal{N}(0, 0.02)$. Here again, we do not take into account cases where uncertainty would make effort negative or above 1. We suppose that the regulator can change fishing intensity with a speed bounded between u_{min} and u_{max}. The fishing fleet's profit is given by :

$$\pi_{xy}(t) = p_x h_x(t) + p_y h_y(t) - ce(t) - C \tag{31}$$

where p_y represents the unit prices of y, p_x the unit price of x. The marginal cost of the fishing effort is c and C is a fixed cost. We define the viability constraint set K:

$$\begin{aligned} K = \Big\{ & (x, y, e) : \\ & x(t) \in \left[\frac{L_x}{10}, L_x\right],\ y(t) \in \left[\frac{L_y}{10}, L_y\right],\ \pi_{xy}(t) \geq 0,\ e(t) \in [e_{min}, e_{max}] \Big\} \end{aligned} \tag{32}$$

The safe minimum biomass levels are set to one tenth of each fish population's unexploited level, as commonly implemented in fisheries worldwide [22] . The parameters values are summed up in Appendix 2.

4.2.2 Results

We compute the stochastic viability kernel $Viab_K(0.8, 10)$ with our approximation. The result is shown on Fig.7, with the stochastic viability kernel computed by dynamic programming for comparison. The set computed by reconstruction is an under-estimation of the stochastic viability kernel computed by dynamic programming, since the control map slightly different of the optimal control map.

4.3 Application to lake eutrophication regulation

4.3.1 Statement of the problem

The last application concerns the management of lake eutrophication. The model has been used in many cases for lake management [20,1,2]. The applications show that the model is capable of behaving in a bistable manner, where regime shifts are able to be triggered by sufficient perturbation under intermediate nutrient loading. In addition, the threshold value of phosphorus loading

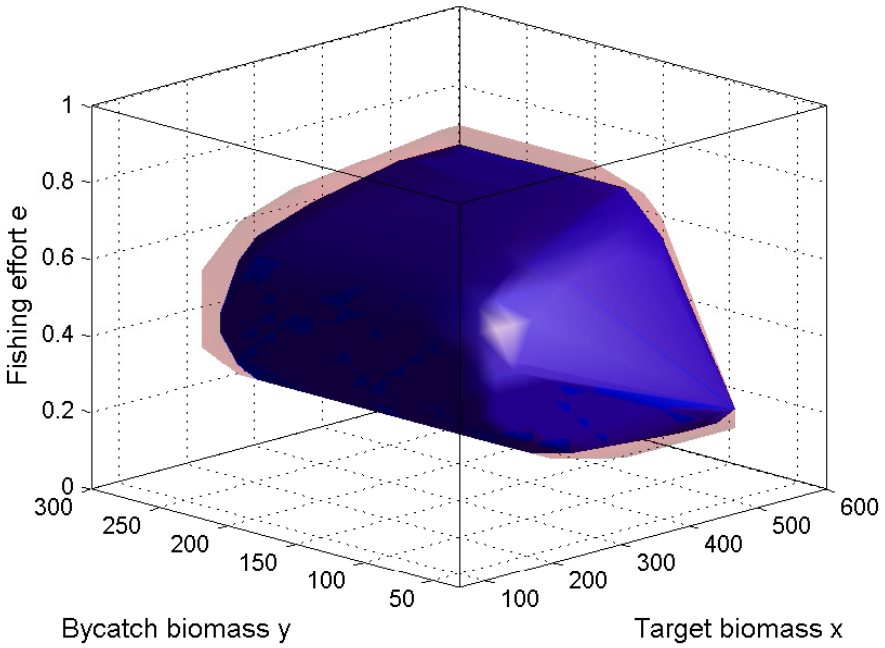

Figure 7. Bycatch fishery stochastic viability kernel. Rebuilt $Viab_K\,(0.8, 10)$ is shown in hard blue. The transparent red set is the stochastic viability kernel computed with dynamic programming.

consistent with field data. We adapt the model introduced by incorporating uncertainties. The system studies the nutrient dynamics and its influence on a trophic chain. We base our work on the following equations :

$$N\,(t+dt) = N\,(t) + dt\left(I_N - r_N N\,(t) - \frac{\gamma r_1 N(t) X(t)}{k_1 + N\,(t)} + \gamma d_4 D\,(t) + \varepsilon_N\,(t)\right) \quad (33)$$

$$X\,(t+dt) = X\,(t) + dt\left(\frac{r_1 N(t) X(t)}{k_1 + N\,(t)} - \frac{f_1 X^2\,(t)\,Y\,(t)}{k_2 + X^2\,(t)} - (d_1 + e_1)\,X\,(t) + \varepsilon_X\,(t)\right) \quad (34)$$

$$Y(t+dt) = Y(t) + dt\left(\frac{\eta f_1 X^2(t) Y(t)}{k_2 + X^2(t)} - \frac{f_2 Y^2(t) Z(t)}{k_3 + Y^2(t)} - (d_2 + e_2) Y(t) + \varepsilon_Y(t)\right) \quad (35)$$

$$Z(t+dt) = Z(t) + dt\left(\frac{\eta f_2 Y^2(t) Z(t)}{k_3 + Y^2(t)} - (d_3 + e_3) Z(t) + \varepsilon_Z(t)\right) \quad (36)$$

$$D(t+dt) = D(t) + dt\Big(\frac{(1-\eta) f_1 X^2(t) Y(t)}{k_2 + X^2(t)} + \frac{(1-\eta) f_2 Y^2(t) Z(t)}{k_3 + Y^2(t)} + d_1 X(t) + d_2 Y(t) + d_3 Z(t) - (d_4 + e_4) D(t) + \varepsilon_D(t)\Big) \quad (37)$$

where N is the amount of nutrient in the lake, X the biomass of phytoplankton, Y the biomass of zooplankton, Z the biomass of zooplanktivorous fish and D is the detrial biomass. We arbitrary define the constraint set:

$$K = \Big\{(N, X, Y, Z, D) : N(t) \in [0, 0.15],\ X(t) \in [0, 20],\ Y(t) \in [0, 0.8],\ Z(t) \in [0, 0.25],\ D(t) \in [0, 30]\Big\} \quad (38)$$

A normal noise is added on each variable, with parameters detailed in Appendix 3. The time horizon is $T = 10$. The available control is the limitation of nutrient input I_N from 0 to 0.0015 discretized in 3 values. We are looking for the stochastic viability kernel $Viab_K(0.8, 10)$. We use a grid of 10 points by dimension. In this case, the computation of all the transition probabilities makes the dynamic programming impossible to use in practice.

4.3.2 Results

The 5 dimensions of the model makes complex the visualization of the results. The computation of the control map shows that the optimal control to maintain the system in K is to reduce the input to 0 for almost 90% of the domain. The control map computed by our approximation can be used for instance to give the maximal probability to maintain the system in k by starting from a given initial state. For instance, for the initial state $x_0 = (0,\ 6.666,\ 0,\ 0.1389,\ 6.666)$, we find a probability over 0.98 to stay in the

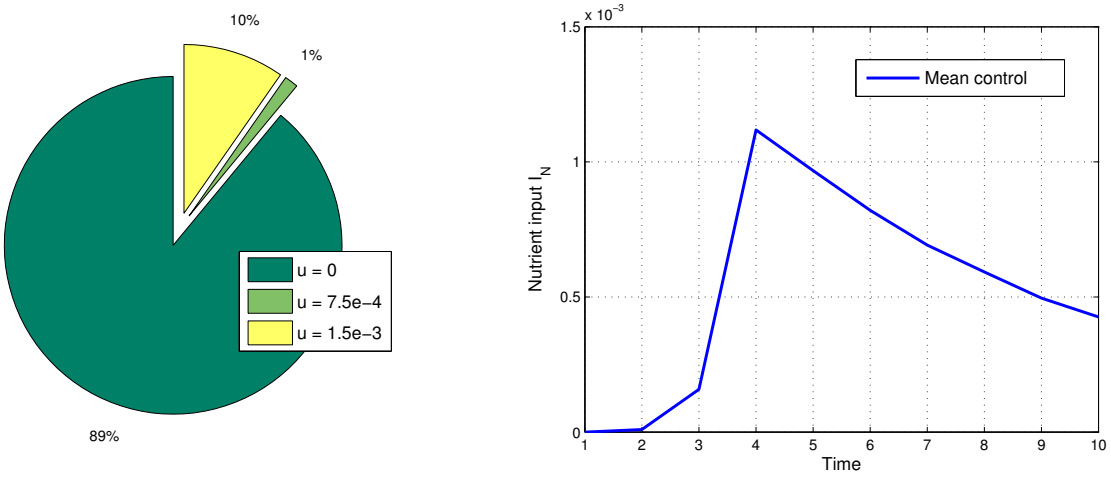

Figure 8. On the left: percentage of each control over the constraint domain at the time horizon T. On the right: mean control applied to the trajectories starting from $x_0 = (0,\ 6.666,\ 0,\ 0.1389,\ 6.666)$ in order to maximize the probability for the system to stay in the constraint domain K, according the control map.

constraint domain K until T (with 100000 trajectories starting from x_0, the mean control is shown in Fig.8 on the left).

5 Conclusion

In this paper, we presented a new method which reduces the memory requirement in dynamic programming when addressing a stochastic viability problem. Instead of computing the whole state-to-state transition matrix, we compute transitions from states to a limited number of level sets. This reduces significantly the memory space requirement. Moreover, reliability techniques are much more faster than Monte-Carlo simulations for computing these transitions. The algorithm is tested on dynamics defined in space from two to five dimensions, with uncertainties on each dimension. The use of reliability methods results in a significant gain in speed and memory space in large dimensions. We used in this paper the first-order reliability method to compute the probabilities of transition. However, second-order reliability method or 2-SMART method may be used [7] to improve the accuracy. The control map computed from the approximation can be used to build an approximate viability kernel. This approach particularly fits viability problems because the reward function is defined as a set (the constraint set). Nevertheless, it would be interesting to test it on other problems with a reward function of different types.

References

[1] Takashi Amemiya, Takatoshi Enomoto, Axel G. Rossberg, Noriko Takamura, and Kiminori Itoh. Lake restoration in terms of ecological resilience: A numerical study of biomanipulations under bistable conditions. *Ecology and Society*, 10(2), 2005.

[2] Takashi Amemiya, Takatoshi Enomoto, Axel G. Rossberg, Tetsuya Yamamoto, Yuhei Inamori, and Kiminori Itoh. Stability and dynamical behavior in a lake-model and implications for regime shifts in real lakes. *Ecological Modelling*, 206(1-2):54–62, 2007.

[3] R. Bellman and R. Kalaba. On adaptive control processes. *IEE Transactions on Automatic Control*, 4(2):1–9, 1959.

[4] C. Béné, L. Doyen, and D. Gabay. A viability analysis for a bio-economic model. *Ecological Economics*, 36(3):385–396, mar 2001.

[5] C Bernard and S Martin. Comparing the sustainability of different action policy possibilities: application to the issue of both household survival and forest preservation in the corridor of Fianarantsoa. *Mathematical biosciences*, 245(2):322–30, oct 2013.

[6] O. Bokanowski, S. Martin, R. Munos, and H. Zidani. An anti-diffusive scheme for viability problems. *Applied Numerical Mathematics*, 56(9):1147–1162, sep 2006.

[7] J. M. Bourinet, F Deheeger, and M Lemaire. Assessing small failure probabilities by combined subset simulation and Support Vector Machines. *Structural Safety*, 33(6):343–353, 2011.

[8] Antoine Brias, Jean-Denis Denis Mathias, and Guillaume Deffuant. Accelerating viability kernel computation with CUDA architecture: application to bycatch fishery management. *Computational Management Science*, pages 1–21, jan 2016.

[9] Michel Broniatowski and Kom Gildas Hermann. Méthodes Form et Sorm. 2014.

[10] Laetitia Chapel, Guillaume Deffuant, Sophie Martin, and Christian Mullon. Defining yield policies in a viability approach. *Ecological Modelling*, 212(1-2):10–15, mar 2008.

[11] G Deffuant and N Gilbert. *Viability and Resilience of Complex Systems: Concepts, Methods and Case Studies from Ecology and Society*. 2011.

[12] O Ditlevsen and H O Madsen. Structural Reliability Methods. (September):360, 2007.

[13] O Ditlevsen and HO Madsen. Structural reliability methods. (July), 1996.

[14] David L Donoho. High-Dimensional Data Analysis: The Curses and Blessings of Dimensionality. In *Math Challenges of the 21st Century*, pages 1–32, 2000.

[15] L. Doyen, M. De Lara, J. Ferraris, and D. Pelletier. Sustainability of exploited marine ecosystems through protected areas: A viability model and a coral reef case study. *Ecological Modelling*, 208(2-4):353–366, 2007.

[16] L. Doyen, O. Thébaud, C. Béné, V. Martinet, S. Gourguet, M. Bertignac, S. Fifas, and F. Blanchard. A stochastic viability approach to ecosystem-based fisheries management. *Ecological Economics*, 75:32–42, mar 2012.

[17] Luc Doyen and Michel De Lara. Stochastic viability and dynamic programming. *Systems & Control Letters*, 59(10):629–634, oct 2010.

[18] Klaus Eisenack, Juergen Scheffran, and Juergen Peter Kropp. Viability analysis of management frameworks for fisheries. *Environmental Modeling and Assessment*, 11(1):69–79, 2006.

[19] Henri P. Gavin and Siu Chung Yau. High-order limit state functions in the response surface method for structural reliability analysis. *Structural Safety*, 30(2):162–179, mar 2008.

[20] E. Jeppesen, M. Søndergaard, E. Mortensen, P. Kristensen, B. Riemann, H. J. Jensen, J. P. Müller, O. Sortkjær, J. P. Jensen, K. Christoffersen, S. Bosselmann, and E. Dall. Fish manipulation as a lake restoration tool in shallow, eutrophic temperate lakes 1: cross-analysis of three Danish case-studies. *Hydrobiologia*, 200-201(1):205–218, 1990.

[21] Shahab Kaynama, Meeko Oishi, Ian M. Mitchell, and Guy a. Dumont. The continual reachability set and its computation using maximal reachability techniques. *IEEE Conference on Decision and Control and European Control Conference*, 0(0):6110–6115, dec 2011.

[22] Jacek B. Krawczyk, Alastair Pharo, Oana S. Serea, and Stewart Sinclair. Computation of viability kernels: a case study of by-catch fisheries. *Computational Management Science*, 10(4):365–396, oct 2013.

[23] Jong Min Lee and Jay H Lee. Approximate Dynamic Programming Strategies and Their Applicability for Process Control: A Review and Future Directions. 2004.

[24] Vincent Martinet, Olivier Thébaud, and Luc Doyen. Defining viable recovery paths toward sustainable fisheries. *Ecological Economics*, 64(2):411–422, 2007.

[25] Jean-Denis Mathias, Bruno Bonté, Thomas Cordonnier, and Francis de Morogues. Using the Viability Theory to Assess the Flexibility of Forest Managers Under Ecological Intensification. *Environmental Management*, (JUNE), 2015.

[26] Jean-Denis Mathias and Maurice Lemaire. Reliability analysis of bonded joints with variations in adhesive thickness. *Journal of Adhesion Science and Technology*, 27(10):1069–1079, 2013.

[27] C Mullon, P Cury, L Shannon, Rogge Bay, Cape Town, and South Africa. Viability model of Trophic Interactions in Marine Ecosystem. *Natural Resource Modelling*, 17(1):27–58, 2004.

[28] Warren B. Powell. *Approximate Dynamic Programming*. Number 1. 2005.

[29] Rüdiger Rackwitz. Reliability analysis-a review and some perspective. *Structural Safety*, 23(4):365–395, 2001.

[30] Charles Rougé, Jean-Denis Mathias, and Guillaume Deffuant. Extending the viability theory framework of resilience to uncertain dynamics, and application to lake eutrophication. *Ecological Indicators*, 29:420–433, jun 2013.

[31] Charles Rougé, Jean-Denis Mathias, and Guillaume Deffuant. Relevance of control theory to design and maintenance problems in time-variant reliability: The case of stochastic viability. *Reliability Engineering & System Safety*, 132:250–260, 2014.

[32] Patrick Saint-Pierre. Approximation of the Viability Kernel. *Applied Mathematics & Optimization*, 29(2):1–22, 1990.

Appendix 1

Consider a problem with N independent normal random variables $X_1, X_2, \ldots, X_N$ in the reduced centered normal space $(O, u_1, u_2, \ldots, u_N)$. The limit-state surface is supposed to be a hyperplan in the reduced centered normal space.

Since the probability density $f = f_1 \times f_2 \times \ldots \times f_N$ has a central symmetry of center O, we suppose that the limit-state surface equation is $u_1 = -\beta$.

Therefore, we obtain the probability of failure P_f, the probability to be in the failure domain, over the limit-state surface:

$$P_f = \int_{u_1=-\infty}^{-\beta} \int_{u_2=-\infty}^{+\infty} \ldots \int_{u_N=-\infty}^{+\infty} f_1(u_1) \times f_2(u_2) \times \ldots \times f_N(u_N)\, du_1 du_2 \ldots du_N$$

Moreover, in the reduced centered normal space, the probability density function f_i on each dimension i is the probability function φ of the standard normal distribution. The probability of failure becomes:

$$P_f = \int_{u_1=-\infty}^{-\beta} \int_{u_2=-\infty}^{+\infty} \ldots \int_{u_N=-\infty}^{+\infty} \varphi(u_1) \times \varphi(u_2) \times \ldots \times \varphi(u_N)\, du_1 du_2 \ldots du_N$$

It leads to:

$$P_f = \int_{u_1=-\infty}^{-\beta} \varphi(u_1)\, du_1 \times \int_{u_2=-\infty}^{+\infty} \varphi(u_2)\, du_2 \times \ldots \times \int_{u_N=-\infty}^{+\infty} \varphi(u_N)\, du_N$$

Since $\int_{-\infty}^{+\infty} \varphi(u)\, du = 1$, we get:

$$P_f = \int_{u_1=-\infty}^{-\beta} \varphi(u_1)\, du_1 = \Phi(-\beta)$$

where Φ is the cumulative distribution function of the standard normal distribution.

Appendix 2

Name	Parameter	Value
Target carrying capacity	L_x	600
Target growth rate	r_x	0.4
Target unit price	p_x	4
Target catchability coefficient	q_x	0.5
Bycatch carrying capacity	L_y	300
Bycatch growth rate	r_y	0.2
Bycatch unit price	p_y	1.9
Bycatch harvest ratio	α_y	0.2
Marginal cost	c	10
Fixed cost	C	150
Maximum effort	e_{max}	1
Minimum effort	e_{min}	0.1
Maximum effort variation	u_{max}	0.01
Minimum effort variation	u_{min}	-0.01

Table 1
Bycatch fishery model parameters

Appendix 3

Name	Parameter	Value	Unit
Loss rate of limiting nutrient	r_n	0.005	day^{-1}
Maximum growth rate of phytoplankton	r_1	0.3	day^{-1}
Half-saturation constant of nutrient	k_1	0.005	gm^{-2}
Half-saturation constant of phytoplankton biomass	k_2	6	$\left[\left(\text{gm}^{-2}\right)^2\right]$
Half-saturation constant of zooplankton biomass	k_3	2	$\left[\left(\text{gm}^{-2}\right)^2\right]$
Feeding rate of zooplankton	f_1	2	day^{-1}
Feeding rate of zooplanktivorous fish	f_2	10	day^{-1}
Ratio of nutrient mass to biomass	γ	0.02	-
Assimilation efficiency	η	0.5	-
Death rate of phytoplankton	d_1	0.1	day^{-1}
Death rate of zooplankton	d_2	0.55	day^{-1}
Death rate of zooplanktivorous fish	d_3	0.5	day^{-1}
Decomposition rate of detritus	d_4	0.1	day^{-1}
Removal rate of phytoplankton from the system	e_1	0.001	day^{-1}
Removal rate of zooplankton from the system	e_2	0.001	day^{-1}
Removal rate of zooplanktivorous fish from the system	e_3	0.001	day^{-1}
Removal rate of detritus from the system D	e_4	0.001	day^{-1}

Table 2
Lake model parameters, from [2]

Parameter	Mean	Standard deviation
ε_N	0	0.003
ε_X	0	0.4
ε_Y	0	0.016
ε_Z	0	0.005
ε_D	0	0.6

Table 3
Stochasticity parameters

Chapitre 4

Application à la gestion d'un système environnemental : l'eutrophisation du lac du Bourget

4.1 Un cas d'étude pour éprouver les outils développés

Les méthodes mises en place dans les chapitres précédents ont été testées principalement sur des modèles jouets. Nous cherchons à travers ce chapitre à les éprouver sur un cas réel de système dynamique environnemental. Nous étudions l'eutrophisation des lacs déjà largement traitée dans la littérature [Martin 04, Chapel 07, Alvarez 10, Rougé 13]. L'eutrophisation est modélisée par la dynamique de phosphore à laquelle des dimensions supplémentaires peuvent être ajoutées. Le travail présenté dans ce chapitre se limite à deux dimensions et est centré essentiellement sur la calibration d'un modèle standard ainsi que celle des incertitudes. Ce travail en deux dimensions constitue la première étape vers la calibration d'un modèle d'eutrophisation de grande dimension. L'article étudie les conséquences potentielles d'un changement dans la variabilité des précipitations sur l'eutrophisation du lac du Bourget, comme indiqué dans l'un des scénarios du Groupe d'Experts Intergouvernemental sur l'Évolution du Climat (GIEC). Un modèle stochastique de dynamique du phosphore est calibré à l'aide des données du Comité Intersyndical pour l'Assainissement du Lac du Bourget (CISALB), afin d'étudier les probabilités de changement de régime.

4.2 Présentation et contributions de l'article

Cet article a été soumis à la revue *Ecology & Society*. C'est un premier pas vers l'analyse d'un système lacustre réel à l'aide d'outils viabilistes. Les principales contributions de ce papier sont les suivantes :

- nous corrélons les précipitations et le taux de phosphore apporté par les affluents du lac entre 2004 et 2014 ;
- nous calibrons un modèle de dynamique du phosphore à l'aide de données extraites des rapports du CISALB. L'étude à l'équilibre de ce modèle permet de déterminer des seuils, qualifiant l'état du lac comme oligotrophe, mésotrophe ou eutrophe ;
- nous étudions l'impact de la sécheresse des années 2000 sur la réoligotrophisation du lac. Les apports moyens en phosphore par les affluents étant réduits durant cette période, on observe une diminution de la quantité de phosphore présent dans le lac de 120 tonnes en 2004 à 40 tonnes en 2014 ;
- nous étudions l'impact de la variabilité des précipitations sur les probabilités de changement de régime du lac. Deux scénarios sont étudiés, l'un reprenant les conditions nominales de précipitation, c'est à dire celles avant la sécheresse du début des années 2000 ; et celui du GIEC considérant une augmentation de la variabilité des précipitations de 25% ;
- nous obtenons les probabilités d'atteindre des états mésotrophes et eutrophes en partant d'un état initial oligotrophe, à l'aide d'un algorithme de programmation dynamique stochastique.

4.3 Conclusions

Cet article est un premier pas dans l'application de la théorie de la viabilité à un cas pratique de gestion de l'eutrophisation. Les étapes de calibration du modèle et de l'aléa y sont développées, suivies d'une étude sur l'importance de la variabilité des précipitations sur la dynamique. Nous montrons qu'à l'horizon 2100, l'augmentation de la variabilité des précipitations selon le scénario du GIEC augmente jusqu'à 6.1% la probabilité d'eutrophisation. Le modèle utilisé est un modèle simplifié comprenant deux variables d'état. Selon les données disponibles, d'autres variables comme le phosphore dans les sédiments, ou la population de phytoplanctons peuvent être ajoutées au modèle.

La théorie de la viabilité a permis, dans cet article, de calculer la probabilité de maintien dans un état oligotrophe, ou la probabilité pour le lac de devenir eutrophe. Nous n'avons pas considéré la possibilité d'un contrôle sur la dynamique. Afin de calibrer les possibilités d'action sur le système, il convient d'effectuer une analyse économique, car les coûts des politiques d'action entrent en jeu dans leur sélection. D'autres phénomènes comme les interactions entre les réseaux trophiques

du lac sont à incorporer au modèle, mais nécessitent également des données supplémentaires pour leur calibration.

4.4 Texte de l'article

Rainfall variability may foster lake regime shifts: evidence from lake Bourget in France

Abstract

The objective of the paper is to model the effect of rainfall variability on lake eutrophication. IPCC (Intergovernmental Panel on Climate Change) reports show that increases in extreme rain events are expected during the 21th century, yielding more and more significant floods. It could result in nutrient over-enrichment, disrupting the equilibrium of lakes and causing eutrophication. We build and calibrate a model of annual phosphorus dynamics of the lake Bourget in France and simulate the evolution of phosphorus concentration for different rainfall scenarios. Results show that the drought of the 2000s has fostered the effect of the management measures in reducing phosphorus input, enabling the compliance of the objective recommended by OECD (Organisation for Economic Co-operation and Development) in 2020. Moreover, a return to previous rainfall variability increases the probability of eutrophication in the short and long terms. For instance, increasing the rainfall variability by 25% increases the probability of eutrophication from 0.5% to 6.1% between 2015 and 2100.

Keywords: Lake eutrophication, Phosphorus dynamics, Rainfall, Regime shift, Uncertainty

1 INTRODUCTION

Human activities are often disrupting the dynamical regime of ecosystems ([Fried et al., 2003, Leng, 2009, Shukla et al., 2008, Ansari and Gill, 2014]). In the particular case of lakes, discharges from agricultural spraying and industrial waste in tributary waters may lead the lake to shift from an oligotrophic state with clear water to a eutrophic state with turbid water due to algae blooms favoured by an excessive nutrient concentration. This is the lake eutrophication [Carpenter, 2005, Sharpley, 1993, Pote et al., 1996].

Lake eutrophication may yield significant economic costs [Bennett et al., 1999, Carpenter et al., 1998]. For instance, losses attributed to lake-front properties values and recreational use of U.S. freshwaters were approximately $2.2 billion annually [Dodds et al., 2009]. Moreover, reducing the nutrient intake is not always sufficient for recovering the oligotrophic state: in some cases, additional interventions must be performed on lake internal mechanisms [NRC (National Research Council), 1992]. These operations are

not always effective or feasible and may be difficult in practice as shown by the relapse of the biomanipulated Lake Zwemlust in the Netherlands six years after its cleaning [Van Donk and Gulati, 1995].

Climate change may also impact eutrophication [Scheffer and Carpenter, 2003, Jeppesen et al., 2007]. For instance, rising temperatures may foster the emergence of rainfall instead of snowfall during winter causing a greater soil leaching with implications for the magnitude of nutrient transport [Górniak and Piekarski, 2002, Woodbury and Shoemaker, 2013]. However, interactions between climate change and eutrophication are complex, and projections are somewhat contradictory because climate-influenced processes have interacting and often opposing effects (Fig.1). A temperature increase would intensify phosphorus (P) diffusion from deep sediments but would also increase P sedimentation and these two flows may have compensatory effects on the P concentration [Bryhn et al., 2010]. Moreover, changes in temperatures and wind may have compensatory effects and overall a modest impact on P concentration.

This paper studies the impact of rainfall variability on the water quality of the lake Bourget in France. Indeed, according to the Intergovernmental Panel on Climate Change (IPCC), the frequency of heavy rainfall is likely to increase, which could affect water quality by increasing pollutant loads flushed into rivers due to soil erosion [Ipcc, 2013]. We use a model of phosphorus concentration dynamics, calibrated on the data of this lake and simulate the evolution of phosphorus concentration for different scenarios of rainfall. We show that the relative drought that took place between 2004 and 2014, had positive effects on the lake's sewerage [Jacquet, 2015]. For the future, we analyze two scenarios: a return to the rainfall variability of before 2004 and an increase of rainfall variability as suggested by the IPCC [Nigel W. Arnell et al., 2014]. Results show that a return to previous rainfall variability affects in a limited way the probability of eutrophication in the short and long terms. However, increasing the variability of rainfalls by 25% increases the probability of eutrophication from 0.5% to 6.1% between 2015 and 2100.

2 LAKE BOURGET CASE STUDY

2.1 Introduction

Lake Bourget is the biggest lake located entirely within France. The lake is located at the southernmost end of the Jura Mountains in the department of Savoie. Its maximum length is 18 km and its surface area equals 44.5 km^2. Two main tributaries flow into the lake: the Leysse river and the Sierroz river. The three smaller rivers are Tillet, Belle-Eau and Chautagne canal. The primary outflows is Canal the Savières, flowing into the Rhône which also minimally contributes to the incoming flow [Jacquet, 2015].

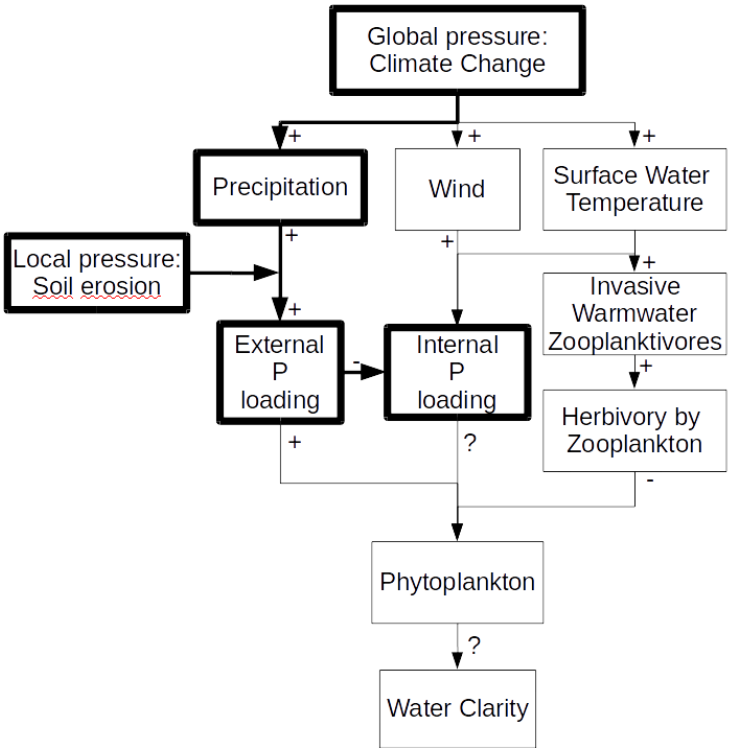

Figure 1: Diagram of interactions between climate change, watershed and lake processes, and water clarity of a lake adapted from IPCC [McCarthy and II., 2001]. A "+" means an increase and a "-" means a decrease in the process; a "?" means contentious expectations. Blooms depend on external and internal loading of phosphorus (P). Wetter climates or more episodic rains lead to more external loading.

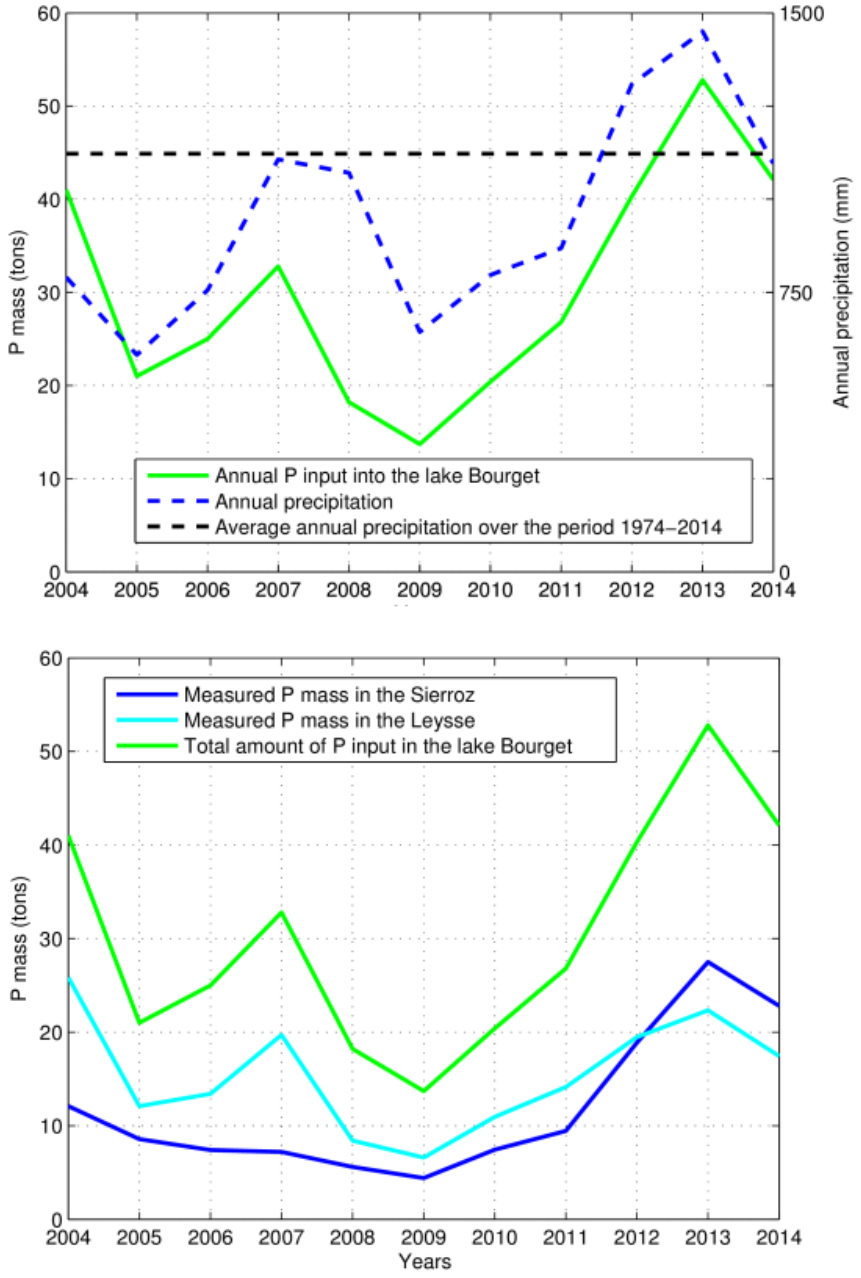

Figure 2: Annual rainfall between 2004 and 2014, measured in Voglans meteorologic station. The left axis measures the annual P input into the lake. The right axis measures the rainfall. On the bottom right: annual P input into the lake Bourget (in green). The blue lines show the contribution of the two main tributary rivers, the Leysse and the Sierroz.

2.2 Mitigating eutrophication

This lake suffered heavy nutrients discharges since the 50s, leading to the eutrophication of the lake. In 1974, the amount of incoming P in the lake reaches 300 tons annually. Then, several measures have been taken to reduce the eutrophication of lake Bourget (Table 1). The main objective was to reduce P input to 30 tons per year. The fixed boundary system given by OECD (Organisation for Economic Co-operation and Development) shows that the current state of the lake is oligo-mesotrophic. The P concentration in the lake should not exceed a threshold of $10\mu g.L^{-1}$ (i.e. 36 tons of P in the lake, if we consider the lake volume as a constant) in order to limit eutrophication. In particular, a 12 km gallery treating effluent stations was set up in 1980 to export treated water from wastewater treatment plants to an environment able to support the rejection. For its flow rate and proximity, the Rhône is preferred to the Isere. This measure lightens the mass of P ending up into the lake. The last major project to date is the construction of a polluted water retention basin for processing by a treatment plant.

The lake Bourget is monitored by the CISALB, the intersyndical committee to clean up the lake, and physico-chemical data are collected regularly. Measurements are made in different places of the lake. In addition, the stations upstream of the lake, on the two main tributaries, collect the phosphorus concentration (Fig. 2). The Sierroz and the Leysse represent the majority of P input. To complete incoming balances, the CISALB takes into account the phosphorus from the Tillet (about 5% of the total volume transited to the lake) and direct discharges to the lake, including the discharge of mixed water by the unit network of Aix les-Bains to the storm overflow of Biâtres. The flows of Belle-Eau and Chautagne canal remain unknown and are neglected.

2.3 Correlation between phosphorous concentration in the lake and rainfall variability

Currently, the ecological state of the lake is healthy, progressing towards an oligotrophic state. Measurement surveys give 150 tons of P in 1983 and 94 tons of P in 1994-1995. Lake Bourget is a successful example of oligotrophication, since the mass of phosphorus is lower than 60 tons from the early 2000s. A weather station provides pluviometry information about the lake situation (the stations are shown in Fig.3). Hydrological studies [Jacquet, 2015] reveal a drought period between 2004 and 2014 (except 2012 and 2013) compared to the mean of the last 40 years, as shown in Fig. 2. CISALB studies have shown that over 90% of P inputs come during rainy episodes. The major floods of the year lead to soil leaching explaining the unusually high concentration of nutrients in the water during these periods. The Pearson correlation between annual rainfall and P loading is0.7402 indicating a high degree of linear dependence as shown in Appendix 1. It suggests that most of the uncertainties in the loading are due to rainfall variability. Thereby, the recent period of low rainfall correlated to

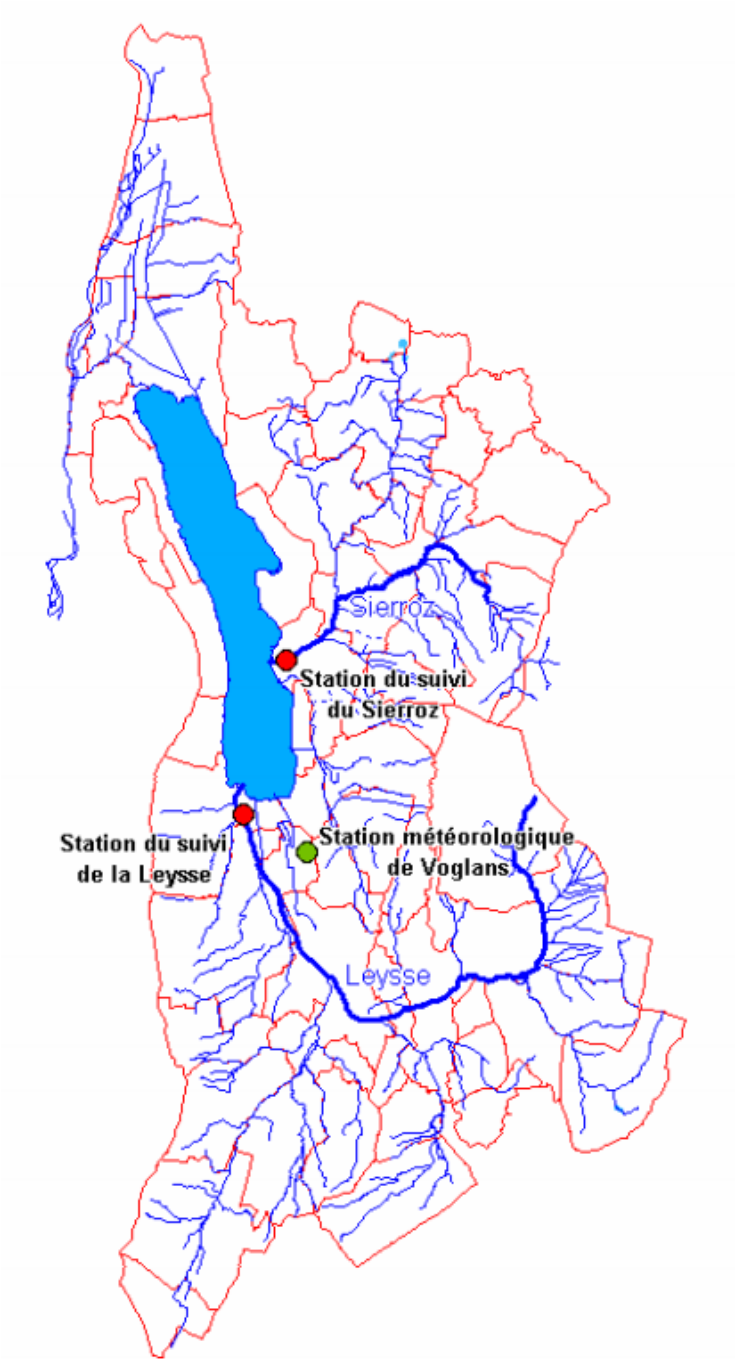

Figure 3: Map of the lac Bourget (from Jacquet et al. 2015). Measurement stations are shown in red. Meteorologic station is shown in green.

Years	Measures	Effects
1976	Expansion of Aix-les-Bains wastewater treatment plant	From 1974 to 1989 : 72% P input decrease, from 300 tons to 86 tons
1977	Expansion of Chambéry wastewater treatment plant	
1981	Creation of a deviation gallery of treated water	
1999	Expansion of Aix-les-Bains wastewater treatment plant	From 1990 to 2000 : 17% P input decrease, from 86 tons to 34 tons
2001	Expansion of Chambéry wastewater treatment plant	P input stabilization (between 30 and 50 tons)
Current project	Creation of storage/return basin on unit network of Chambéry and Aix-les-Bains	

Table 1: Non-exhaustive list of measures taken against the lake Bourget eutrophication.

the fact that nutrient intakes are smaller in case of dry weather may explain the decrease in P contributions in recent years. Therefore, we assume in this paper that the changes in P loading are mainly caused by changes in rainfall variability.

2.4 Mitigating phosphorous concentration: an environmental challenge

Despite the current good ecological state of the lake, some elements undermine the CISALB results. Frequent blooms of toxic cyanobacteria (*Plantothrix rubescens*) appear, and a Microcystis bloom was observed in 2014. It suggests that the current state of the lake is unstable [Jacquet, 2015]. Indeed, weather conditions may yield nutrient inputs to the lake exceeding the regulation threshold. In addition, the mechanics of lag release of phosphorus stored in sediments may also deteriorate the state of the lake. In order to take into account both the rain variability and the P stored in the sediments, we use an annual phosphorus dynamical model. This model, calibrated over the period 2004-2014, yields the amount of phosphorus in the lake each year, taking as input the phosphorus in the lake the previous year and phosphorus coming into the lake during the year. We use this model to forecast the probability to a comeback to a eutrophic state.

3 MODELING THE EUTROPHICATION DYNAMICS

3.1 Annual phosphorous dynamics

Many models study the P dynamics by separating the dynamics according to the epilimnion and the hypolimnion [Lung et al., 1976] and the different phases of the phosphorous [Malmaeus and Håkanson, 2004]. Some of them were built for a specific lake and hardly reusable without having the same set of parameters [Vigiak et al., 2012]. A monthly dynamic model was developed for the lake Bourget yielding concentrations and flows in three water layers [Bryhn et al., 2010]. We present here a model taking into account an annual P dynamics developed by Carpenter et al. [Carpenter et al., 1999] and used in Rougé et al. [Rougé et al., 2013]. Two state variables are used: the whole mass of P present in the lake, noted P and the phosphorus loading, L (Fig. 4). The model is built from the analogous equation for P :

$$\frac{dP}{dt} = L - (s+h)\,P + r\frac{P^q}{P^q + m^q} \tag{1}$$

$s$is the rate of sedimentation, h is the rate of out-flooding P, r is the maximal mass of phosphorus recycled by sediments, q is an exponent depending on the type of the lake, m is the P mass for which the recycling reaches the half of the maximal rate. For this problem, rates of sedimentation s and out-flooding h are not small (i.e, $s+h$ can be upper than 1) and therefore a linear model is not suitable [Ludwig et al., 2012]. Interactions exist between these processes and the difference equation must take into account for the possibility of several sedimentation and recycling events during a single year. By following Carpenter et al. (1999) and Ludwig et al. (2012), we derive the differential equation from Eq. 1:

$$P_{t+1} = e^{-s-h}P_t + \frac{1-e^{-s-h}}{s+h}\left(L_t + r\frac{P_t^q}{P_t^q + m^q}\right) \tag{2}$$

The main sources of P are the inflowing streams coming from the Leysse and the Sierroz rivers. This study neglects the diffuse soil P which ends up in the lake. Loadings are uncertain via unpredictable shocks attributable to stochastic drivers such as the weather or punctual pollutions. The estimation of loading uncertainties is presented in 3.4.

3.2 Calibration of ecological parameters

The annual reports from CISALB give annual P loading and annual P in the lake from 2004 to 2014. Although P loading data exist from the 80s, data are very sparse in time. The data used in this paper as well as the calibration method is shown in Appendix 1. Parameters s, h, r, q and m are calibrated by using a nonlinear curve-fitting problems solver in least-squares sense. We obtain the

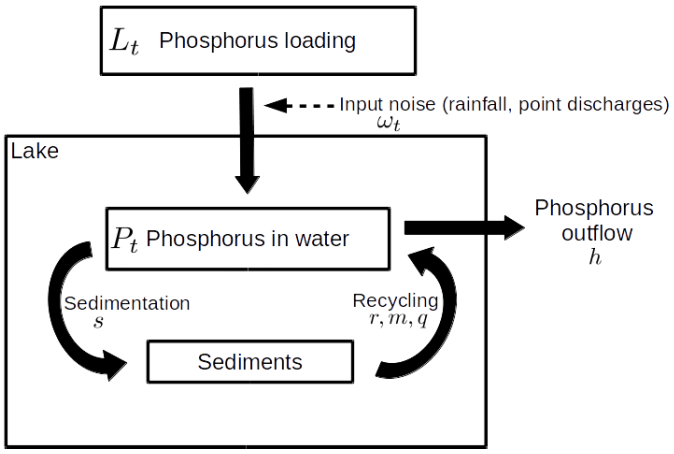

Figure 4: Phosphorus flows in the model (adapted from Carpenter and Brock (2006))

following parameters: $s = 2,3262\,yr^{-1}$, $h = 0.1217\,yr^{-1}$, $r = 244\,tons.yr^{-1}$, $q = 3.1709$ and $m = 66.3624\,tons$. The goodness-of-fit is 0.8225 using normalized root mean square error, which is suitable. The exponent q is in agreement with Carpenter, because for deep lakes, q is close to 2 while in a shallow lake, it is around 20. Concerning the input uncertainties calibration, the 11 values between 2004 and 2014 do not allow an accurate estimation of the distribution followed by the inputs. We assume that L_t follows a log-normal distribution since it cannot be negative. We use a maximum likelihood estimator giving the mean $\mu_L = 30.3727$ and standard deviation $\sigma_L = 12.3$, giving $\mu_{log} = 3.336$ and for $\sigma_{log} = 0.42$ the input log-normal distribution. The 95% confidence interval for μ_{log} is $[3.0534, 3.6181]$ and the one for σ_{log} is $[0.2936, 0.7375]$. With these parameters, the hypothesis of a log-normal distribution is not rejected by the Kolmogorov-Smirnov test (the p-value is equal to 0.8648). By using the model, we obtain Fig. 5 during the period 2004-2014. We predict the P mass in the lake, using the measured P input during the whole period 2004-2014 and the P mass in the lake in 2004. There is a continuing decrease in the level of phosphorus in the lake since the early 2000s. The approximation using the dynamics presented before fits the data.

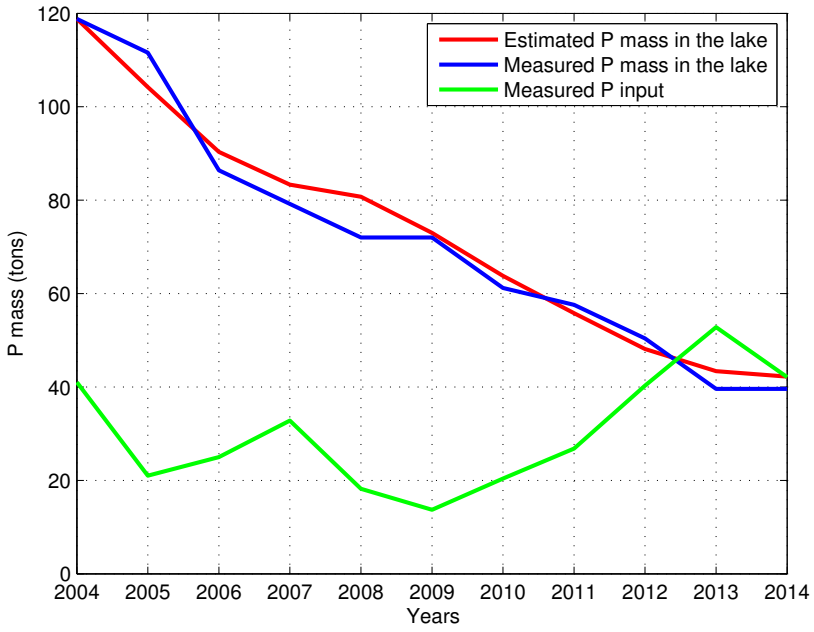

Figure 5: Evolution of phosphorus level of the lake Bourget.

3.3 OECD recommendations

We recall that OECD norms consider the state of the lake as oligotrophic if the level of P in the lake is below 36 tons. These phosphorus thresholds are calculated from the Vollenweider model [OECD, 1982] that can be easily used by managers and technical personnel with only limited limnological knowledge. Studying equilibrium states of our model provides information on those recommendations. The bifurcation diagram is shown in Fig. 6. The stability of the low equilibrium becomes hazardous as L approaches 57 tons because the domain of attraction moves back. When $L = 57$ tons the concentration of phosphorus jumps to high values. Once the mass of phosphorus in the lake has reached high values of equilibrium, phosphorus input has to be reduced below $L = 38$ tons to recover low equilibria. The recommendation of the OECD about the maximum P in the lake (36 tons) fits well with the model. This threshold corresponds to the boundary between oligotrophic states and mesotrophic states. In addition, we fix a second threshold (76 tons) above which the amount of P in the lake is in the range of the upper equilibria . It corresponds to the boundary between mesotrophic and eutrophic states. Note that, because the inputs are low, the state of the lake is in the attraction basin of low equilibria since 2005. Return time diagrams may be computed, giving additional informations about the system. They give an indication of the time required to return to oligotrophic

state in the low equilibrium domain of attraction for fixed P inputs (Fig. A2.1). Similarly, we estimate the time to bring the system to a eutrophic state with high P input.

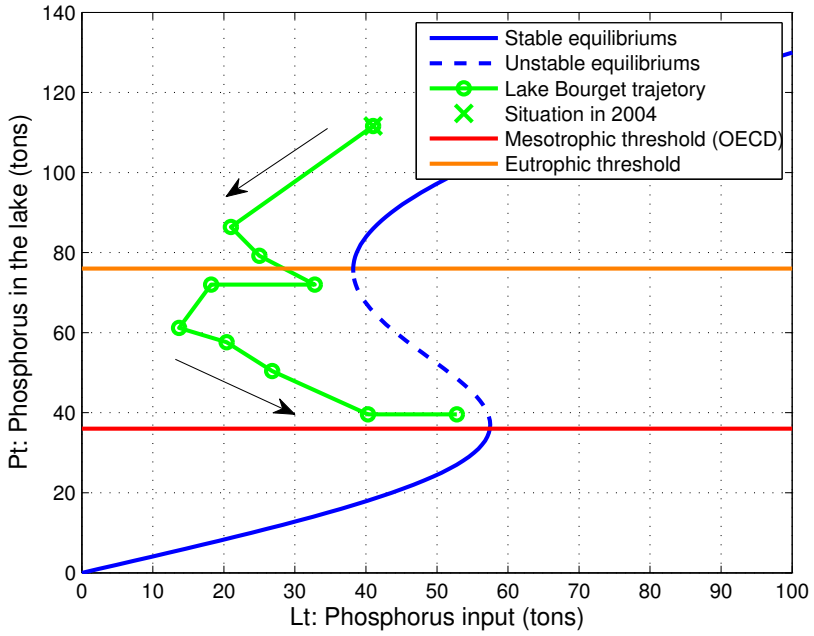

Figure 6: Bifurcation diagram for our model. Equilibriums are shown in blue, trajectory of the lake Bourget in green. The OECD recommendation separating mesotrophic and oligotrophic states is shown in red. For this lake, we fix a threshold in orange for which the state of the lake is defined as eutrophic.

4 ASSESSING SHORT- AND LONG-TERM PHOSPHOROUS EVOLUTION

4.1 Lake benefits from recent drought

In 2014, the level of P in the lake nearly reaches the OECD threshold. We propose to exhibit the role of recent drought on this result. For this purpose, we calculate all possible trajectories of the P in the lake according to the input hazards from 2004 initial conditions (see Appendix C for mathematical details). Results are reported in Fig. 7. The main result is that current P concentration is lower than the mean P concentration in 2014 according to the model. It suggests that the lake benefits from this recent drought in terms of water quality. The input P level and the P present in the lake are reaching the recommended

values of the CISALB qualifying the state of the lake as oligotrophic. The input P level and the P present in the lake are reaching the recommended values of the CISALB qualifying the state of the lake as oligotrophic.

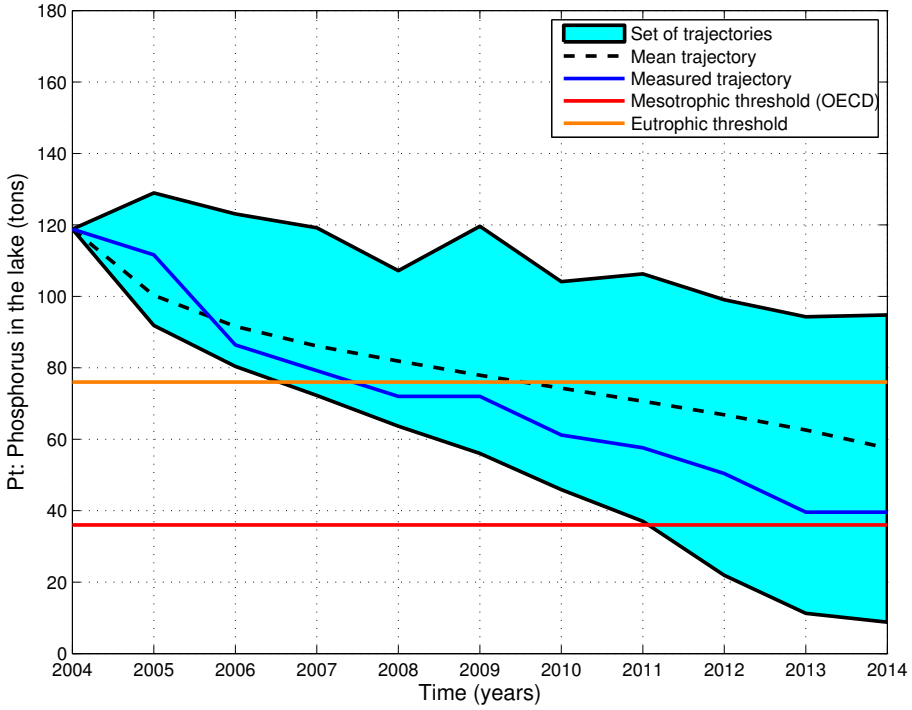

Figure 7: Running the model from 2004. A set of 10000 trajectories is computed and the range in which these trajectories evolve is shown in cyan. The dotted line shows the mean trajectory. The blue line shows the measured trajectory.

4.2 Short-term prediction: estimating the state of the lake in 2020

One objective of the CISALB is to reach lake management objectives in 2020. We estimate the probability to being under the thresholds in 2020. We considered that the rainfall variability has the same settings as during the interval 2004-2014, until 2020. It is an optimistic assumption, given the drought between 2004 and 2011 which has favored the reduction of nutrient inputs to the lake. As a first step, we compute possible trajectories starting from the level of P measured in 2004. Results are shown in Fig. 8. A main trend emerges in 2020. A decline began in the early 2010s to reach a mean P level in the lake in 2020 under 20 tons. The level of P has 73.2% chances to be under the threshold of 36 tons in 2020.

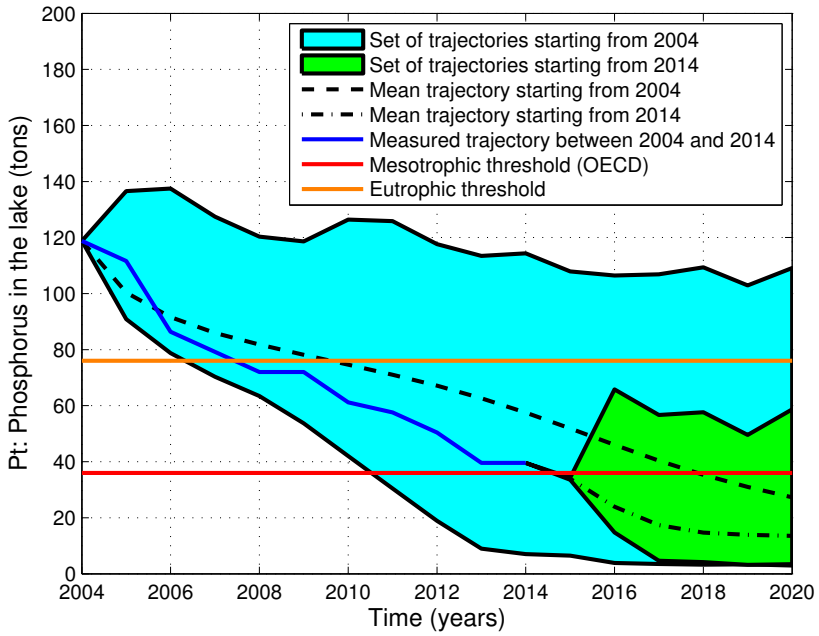

Figure 8: Predictions starting from 2004 and 2014 until 2020. A set of 10000 trajectories is computed and the range in which these trajectories evolve is shown in cyan (starting from 2004) and green (starting from 2014). The dotted lines show the mean trajectory for the two sets. The blue line shows the measured trajectories. The green set of trajectories is computed with stochastic parameters of 2004-2014. However using former parameters (nominal scenario) does not change significantly neither the mean nor the shape of the green set between 2015 and 2020.

However, the situation between 2004 and 2014 is known. The mean trajectory from these initial conditions shows a significant decrease of phosphorus in the lake. There is a difference of 13.9 tons in 2020 between the mean trajectory starting from 2004 and the mean trajectory starting from 2014, dividing by two the mean amount of P in the lake in 2020. According to the CISALB, the low pluviometry of last years could explain this decrease of P in the lake. We note that mean trajectory starting in 2004 drops below the OECD threshold in 2018, while the mean trajectory starting from the state in 2014 reaches this threshold in 2015. This objective – reaching OECD threshold - will be achieved in 2020 with a probability over 0.99 according the model. The difference in results between the prediction starting from 2004 and the one starting from 2014 shows that P inputs between 2004 and 2014 have probably fostered a return to the low equilibrium of P in the lake.

4.3 Long-term prediction: scenario of change in rainfall variability until 2100

As there is a correlation between rainfall and nutrient inputs, we study the potential impact of increased rainfall variability on the state of the lake. We assume that a change in the rainfall variability results in a change of standard deviation of P loading. We focus our work on two scenarios: a return to the rainfall variability of the period 2000 - 2003 used in Brynh et al. (2010); and an increase of 25% of the variability which is in the range of IPCC predictions. The parameters of these scenarios are summarized in Table 2.

Parameters	Nominal precipitation variability scenario (scenario 1)	High precipitation variability scenario (scenario 2)
Mean loadingμ_L	36	36
Loading standard deviation σ_L	16	20
Log-normal distribution parameters (μ_{log},σ_{log})	(3.48, 0.4246)	(3.45, 0.5186)

Table 2: Parameters of long-term rainfall scenarios.

After running the model, we observe that a return to P variability of period 2000-2003 (scenario 1) will lead to the lake oligotrophication in the long term except for a negligible set of trajectories that exceed the eutrophication threshold (Fig. 9). Cleaning operations conducted by the CISALB made the lake strong enough to withstand disturbances since the mean P in the lake is between 16.7 tons and 18.2 tons after 2020 according the scenarios. Besides, the mean trajectory is slightly affected by increasing rainfall variability (scenario 2), However, increase in variability leads to a higher number of trajectories exceeding the eutrophication threshold (Fig. 10), and the probability of regime shift is no longer negligible .

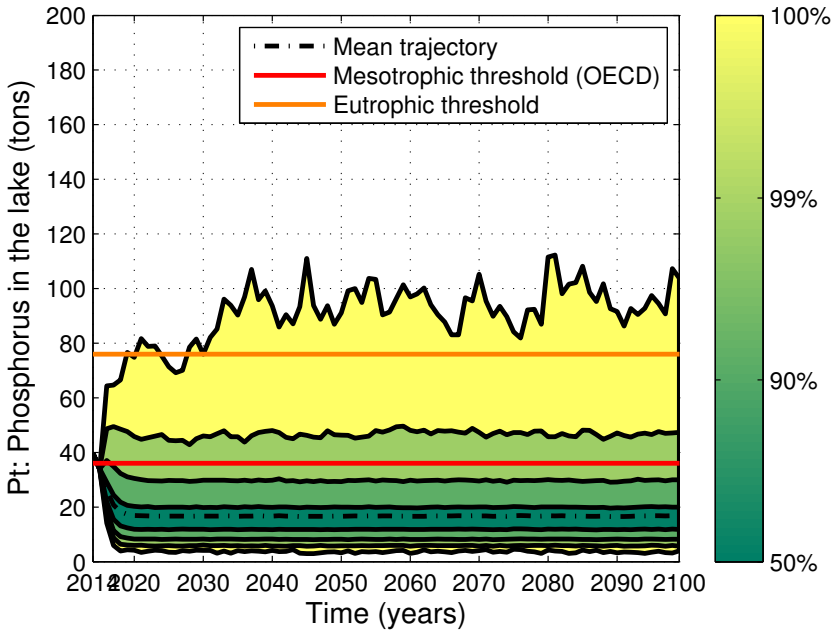

Figure 9: Sets of 10000 trajectories starting from 2014 until 2100 according the nominal scenario. Four stacked sets are represented around the median trajectory. The yellow set contains all the possible states of the lake according the trajectories at each date. We show the different sets respectively excluding the 1%, 10% and 50% of extreme states compared to the median for each year (from the yellow to the greenest). The mean trajectory is below the mesotrophic threshold.

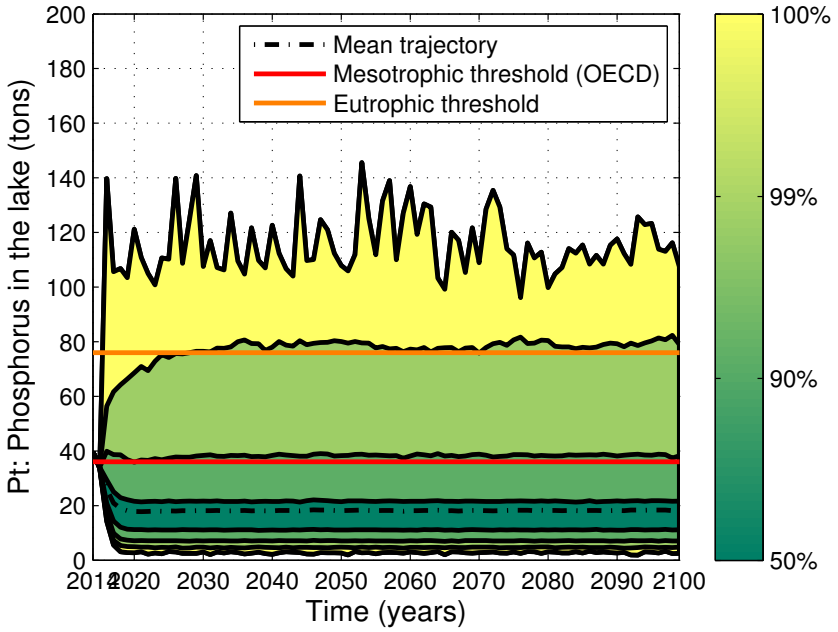

Figure 10: Sets of 10000 trajectories starting from 2014 until 2100 according the high variability precipitation scenario. Four stacked sets are represented around the median trajectory. The yellow set contains all the possible states of the lake according the trajectories at each date. We show the different sets respectively excluding the 1%, 10% and 50% of extreme states compared to the median for each year (from the yellow to the greenest). The mean trajectory is below the mesotrophic threshold but the variability of P inputs increases the number of trajectories reaching the eutrophication threshold (orange horizontal line).

Different pathways can be observed depending on the inputs to the lake. In most cases, phosphorus inputs are not sufficient to shift the lake from a mesotrophic or oligotrophic regime to a eutrophic regime. The dynamics tends to stabilize the P in the lake towards low equilibriums. However, some increases in P concentration to a mesotrophic level can be observed. In the worst cases, with several consecutive years of high P inputs, the dynamics may lead to a eutrophic regime. However, in the same manner, this phenomenon can be reversed by years with low input; the dynamics leading the state of the lake to the low equilibriums. Different trajectories are illustrated in Fig. 11 and Fig. 12 .

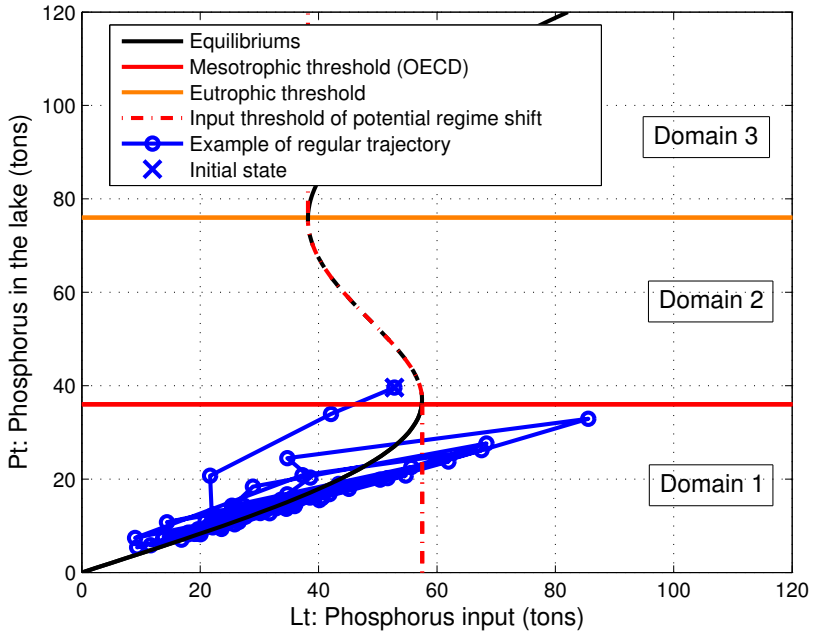

Figure 11: Trajectory leading to stabilization to a oligotrophic states according the nominal scenario.

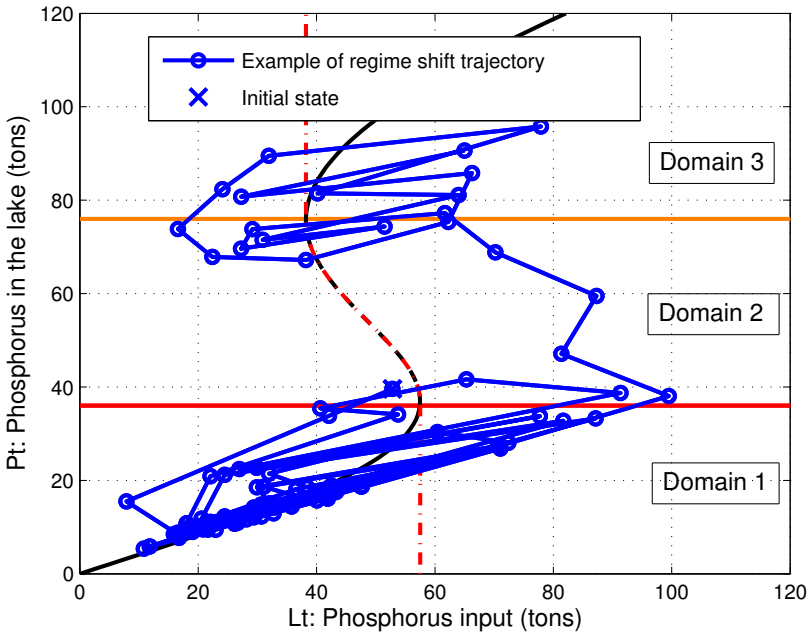

Figure 12: Extreme trajectory leading to a regime shift in case of high precipitation variability scenario.

The mean residence time in each of the three domains according to scenarios is shown in Table 3. In both scenarios the mean time in the eutrophic domain is lower than 1 year. However, we observe that the change in variability significantly affects the probability to reach the eutrophic domain, from 0.5% to 6.1%. This result highlights the role of input variability and thereby, the role of rainfall variability. It may be extended to any lake whose initial state is known. The probability for an oligotrophic lake to become eutrophic may be calculated over a period of time. It is presented in Appendix 3.

	Nominal precipitation scenario	High precipitation variability scenario
Mean time in the oligotrophic domain	≈ 82.3 years	≈ 79 years
Mean time in the mesotrophic domain	≈ 2.6 years	≈ 5.5 years
Mean time in the eutrophic domain	≈ 0 year	≈ 0.5 years
Probability to reach the eutrophic domain	$\approx 0.5\%$	$\approx 6.1\%$

Table 3: Mean time in each domain according each scenario, based on 10000 trajectories between 2015 and 2100. The probability to reach the eutrophic domain is computed too.

5 DISCUSSION AND CONCLUSION

We investigated the effect of increasing the variability of P loading – due to an increase of rainfall variability - on the eutrophication of lake Bourget. The simulations in the short term show a stabilization of the lake state below OECD recommendations, and suggest that the recent drought favored this stabilization. In the long term, an increase in rainfall variability is likely to increase the probability of eutrophication. Even if our quantitative results should be viewed with caution, given the small size of the data set used, the observed qualitative trend is clear. We use a dynamical model of the phosphorus concentration in the lake. Its equilibriums help determine trophic thresholds. We can then assess the probability of lake eutrophication, by running a large number of simulations with sequences of rainfalls of the chosen variability, drawn at random.

The model could be improved by calibrating it on a larger series of data, if available, or by adding mechanisms such as the passive diffusion of nutrients from soil. Other variables may also be included in the model, such as the chlorophyll a or the luminosity (as recommended by OECD). As shown in Fig. 1, our model primarily estimates the impact of changes in rainfall, expressed by the variability of phosphorus input. It might also be relevant to develop a model taking into account other effects of climate change, such as surface water temperature variations which affect the development of algal blooms in summer periods. Besides, a time step smaller than the year may give relevant additional information about the system. For instance, a monthly model could account for the temporal distribution of floods during the year. Indeed, two rainy events close in time do not leach the soil like the same two events separated by a long time [Merceron, 1999].

Acknowledgements

We acknowledge the financial support of Irstea and Région Auvergne. We graciously thank the work of the CISALB and the work of all people involved in Lake Bourget surveys.

References

[Ansari and Gill, 2014] Ansari, A. A. and Gill, S. S. (2014). Eutrophication: Causes, consequences and control: Volume 2. *Eutrophication: Causes, Consequences and Control: Volume 2*, pages 1–262.

[Bennett et al., 1999] Bennett, E. M., Reed-Andersen, T., Houser, J. N., Gabriel, J. R., and Carpenter, S. R. (1999). A Phosphorus Budget for the Lake Mendota Watershed. *Ecosystems*, 2(1):69–75.

[Bryhn et al., 2010] Bryhn, A. C., Girel, C., Paolini, G., and Jacquet, S. (2010). Predicting future effects from nutrient abatement and climate change on phosphorus concentrations in Lake Bourget, France. *Ecological Modelling*, 221(10):1440–1450.

[Carpenter et al., 1999] Carpenter, S., Ludwig, D., and Brock, W. (1999). Management of Eutrophication for Lakes Subject to Potentially Irreversible Change. *Ecological Applications*, 9(3):751–771.

[Carpenter, 2005] Carpenter, S. R. (2005). Eutrophication of aquatic ecosystems: bistability and soil phosphorus. *Proceedings of the National Academy of Sciences of the United States of America*, 102(29):10002–5.

[Carpenter et al., 1998] Carpenter, S. R., Bolgrienr, D., Lathrop, R. C., Stow, C. a., and Reed, T. (1998). Ecological and economic analysis of lake eutrophication by nonpoint pollution. *Australian journal of Ecology*, 23:68–79.

[Chapel et al., 2007] Chapel, L., Martin, S., and Deffuant, G. (2007). Lake eutrophication: using resilience evaluation to compute sustainable policies. *Proceedings of the ...*, (September):5–7.

[Dodds et al., 2009] Dodds, W. K., Bouska, W. W., Eitzmann, J. L., Pilger, T. J., Pitts, K. L., Riley, A. J., Schloesser, J. T., and Thornbrugh, D. J. (2009). Eutrophication of U. S. freshwaters: Analysis of potential economic damages. *Environmental Science and Technology*, 43(1):12–19.

[Fried et al., 2003] Fried, S., Mackie, B., and Nothwehr, E. (2003). Nitrate and phosphate levels positively affect the growth of algae species found in Perry Pond. pages 21–24.

[Górniak and Piekarski, 2002] Górniak, A. and Piekarski, K. (2002). Seasonal and Multiannual Changes of Water Levels in Lakes of Northeastern Poland. *Polish Journal of Environmental Studies*, 11(4):349–354.

[Ipcc, 2013] Ipcc (2013). Summary for Policymakers. *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*, page 33.

[Jacquet, 2015] Jacquet, S. (2015). Suivi Scientifique Du Lac Du Bourget Annee 2014. (avril).

[Jeppesen et al., 2007] Jeppesen, E., Kronvang, B., Meerhoff, M., Søndergaard, M., Hansen, K. M., Andersen, H. E., Lauridsen, T. L., Liboriussen, L., Beklioglu, M., Ozen, A., and Olesen, J. E. (2007). Climate change effects on runoff, catchment phosphorus loading and lake ecological state, and potential adaptations. *Journal of environmental quality*, 38(5):1930–1941.

[Leng, 2009] Leng, R. (2009). The Impacts of Cultural Eutrophication on Lakes: A Review of Damages and Nutrient Control Measures. pages 33–39.

[Ludwig et al., 2012] Ludwig, D., Carpenter, S. R., and Brock, W. A. (2012). Optimal Phosphorus Loading for a Potentially Eutrophic Lake. *Ecological Applications*, 13(4):1135–1152.

[Lung et al., 1976] Lung, W. S., Canale, R. P., and Freedman, P. L. (1976). Phosphorus models for eutrophic lakes. *Water Research*, 10(12):1101–1114.

[Malmaeus and Håkanson, 2004] Malmaeus, J. M. and Håkanson, L. (2004). Development of a Lake Eutrophication model. *Ecological Modelling*, 171(1-2):35–63.

[Martin, 2004] Martin, S. (2004). The Cost of Restoration as a Way of Defining Resilience: a Viability Approach Applied to a Model of Lake Eutrophication. 9(2).

[McCarthy and II., 2001] McCarthy, J. J. and II., I. P. o. C. C. W. G. (2001). *Climate Change 2001: Impacts, Adaptation, and Vulnerability: Contribution of Working Group II to the Third Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press.

[Merceron, 1999] Merceron, M. (1999). *Pollutions diffuses: du bassin versant au littoral : actes de colloques, Ploufragan (Saint-Brieuc), [23-24] Septembre 1999*. Editions Quae.

[Nigel W. Arnell et al., 2014] Nigel W. Arnell, Benito, G., Cogley, J. G., Döll, P., Jiang, T., and Mwakalila, S. S. (2014). Drivers of Change for Freshwater Resources: Climatic Drivers. *Climate Change 2014: Impacts, Adaptation, and Vulnerability*, pages 229–269.

[NRC (National Research Council), 1992] NRC (National Research Council) (1992). Restoration of aquatic ecosystems. Technical report.

[OECD, 1982] OECD, P. F. (1982). OECD 1982 Eutrophication of waters. Monitoring, assessment and control. OECD, Paris.

[Pote et al., 1996] Pote, D. H., Daniel, T. C., Moore, P. A., Nichols, D. J., Sharpley, A. N., and Edwards, D. R. (1996). Relating Extractable Soil Phosphorus to Phosphorus Losses in Runoff. *Soil Science Society of America Journal*, 60(3):855.

[Rougé et al., 2013] Rougé, C., Mathias, J.-D., and Deffuant, G. (2013). Extending the viability theory framework of resilience to uncertain dynamics, and application to lake eutrophication. *Ecological Indicators*, 29:420–433.

[Scheffer and Carpenter, 2003] Scheffer, M. and Carpenter, S. R. (2003). Catastrophic regime shifts in ecosystems: linking theory to observation. *Trends in Ecology & Evolution*, 18(12):648–656.

[Sharpley, 1993] Sharpley, A. N. (1993). Assessing phosphorus bioavailability in agricultural soils and runoff. *Fertilizer Research*, 36(3):259–272.

[Shukla et al., 2008] Shukla, J. B., a.K. Misra, Chandra, P., a.K. Misra, and Chandra, P. (2008). Modeling and analysis of the algal bloom in a lake caused by discharge of nutrients. *Applied Mathematics and Computation*, 196(2):782–790.

[Van Donk and Gulati, 1995] Van Donk, E. and Gulati, R. D. (1995). Transition of a lake to turbid state six years after biomanipulation: Mechanisms and pathways. *Water Science and Technology*, 32(4):197–206.

[Vigiak et al., 2012] Vigiak, O., Rattray, D., McInnes, J., Newham, L. T. H., and Roberts, a. M. (2012). Modelling catchment management impact on in-stream phosphorus loads in northern Victoria. *Journal of environmental management*, 110:215–225.

[Woodbury and Shoemaker, 2013] Woodbury, J. and Shoemaker, C. A. (2013). Stochastic Assessment of Long-Term Impacts of Phosphorus Management Options on Sustainability with and without Climate Change. (October):512–519.

Appendix 1. Correlation between precipitation and P loading

The CISALB reports show that the P contributions are more significant during rainy events, mainly due to soil leaching. Annual data from 2004 to 2014 are available and we used it for computing the correlation between annual precipitation and annual P inputs. The calculation indicates a Pearson correlation index of 0.74. It means a positive correlation between annual precipitation and annual P inputs (Fig. A1.13).

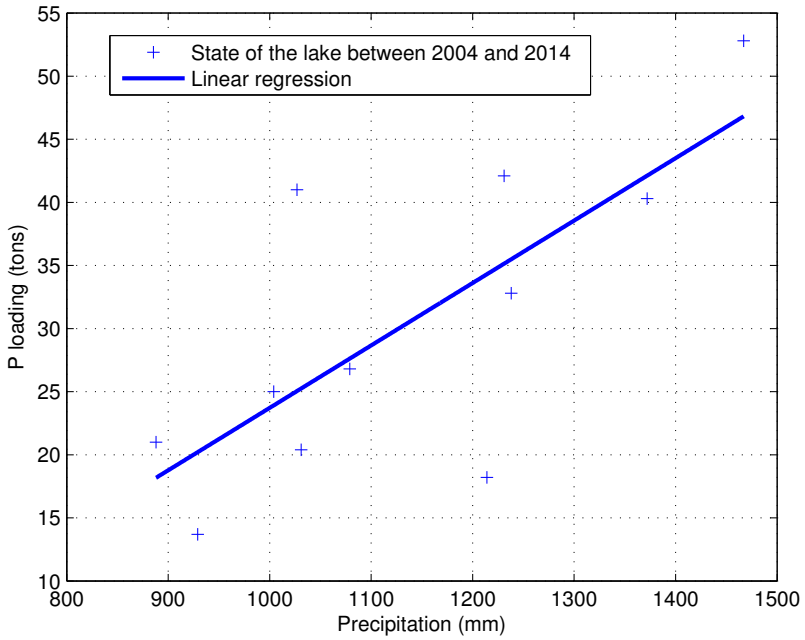

Figure 13: Relation between precipitation and P loading. A positive correlation exists between these two parameters.

Appendix 2. Return time diagram.

We plot the return time diagram in Fig. A2.14. This diagram shows the amount of time (in years) necessary to return to the two types of equilibriums from their attraction basins with constant P input. It shows that a decrease in P input is necessary to switch from the high steady state to a state of the basin of attraction of the low equilibrium states. Conversely, a sudden input of P may cause the lake to go from an oligotrophic state to the attraction basin of eutrophic equilibriums.

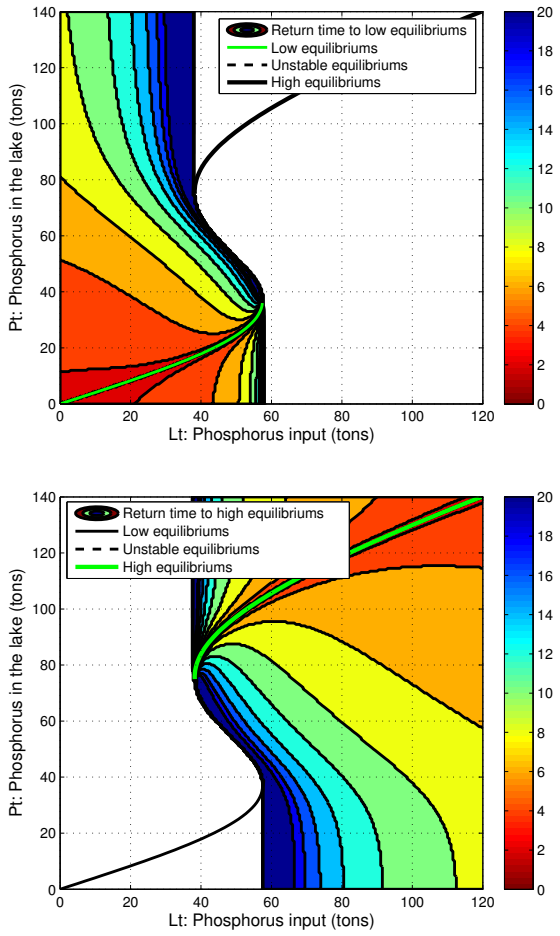

Figure 14: Top: return time (years) to low equilibriums with constant inputs. It corresponds to a return to an oligotrophic state of the lake. Bottom: return time (years) to high equilibriums with constant inputs. It corresponds to a return to a eutrophic state of the lake.

Appendix 3. Reaching the thresholds from an oligotrophic lake.

By using dynamic programming, we compute the mean probability to reach the mesotrophic domain and the eutrophic domain by starting from the oligotrophic domain.

	Nominal precipitation scenario	High precipitation variability scenario
Reaching the mesotrophic domain	$\approx 50\%$	$\approx 86\%$
Reaching the eutrophic domain	$\approx 0.14\%$	$\approx 1.6\%$

Table 4: Mean probability to reach the mesotrophic domain and the eutrophic domain by starting from the oligotrophic domain, during 86 years, for each scenarios.

Relying on the viability theory [Martin, 2004, Chapel et al., 2007], we look for the set gathering all initial lake states for which the probability to become eutrophic until 86 years is less than 10% (until 2100 with initial data from 2014). The result is shown in Fig. A3.15. The lake Bourget state in 2014 belongs to this set but it is not far from its edge. It means that at the end of the century, the lake will have a good probability to be oligotrophic or mesotrophic. The definition of such a set can be used as a standard dynamically computed according the studied lake parameters.

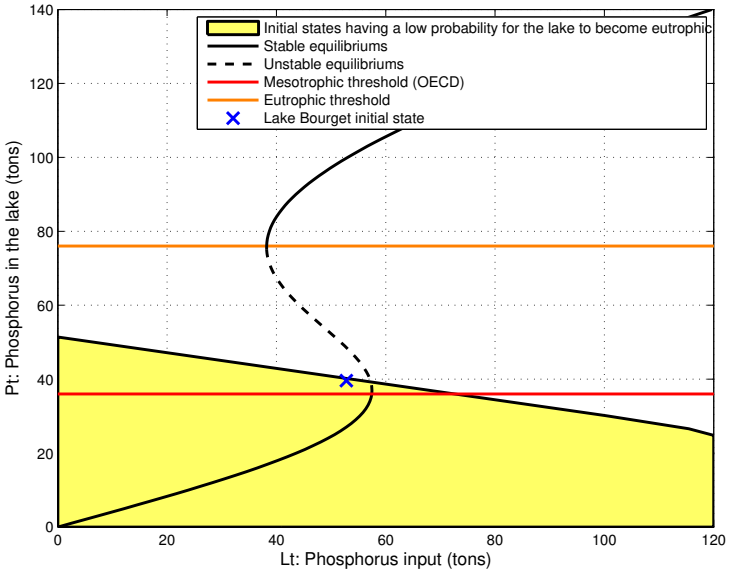

Figure 15: Set of initial states of the lake for which the probability to become eutrophic is less than 10% until 86 years for the high precipitation variability scenario. The state of lake Bourget in 2014 belongs to this set.

Chapitre 5

Application à un problème fiabiliste : la conception fiable d'une poutre soumise à la corrosion

5.1 Utiliser la théorie de la viabilité dans un problème de fiabilité évoluant dans le temps

De la même manière que nous utilisons des outils fiabilistes pour résoudre un problème en viabilité, nous utilisons ici la théorie de la viabilité stochastique pour la résolution d'un problème de fiabilité mécanique. Cet article est donc le pendant du **chapitre 3.** Tandis qu'en viabilité on cherche à maintenir un système dans un domaine de contraintes ; la fiabilité, elle, cherche à calculer la probabilité de défaillance d'un système. Cette défaillance survient quand le système se retrouve dans un domaine de défaillance, *i.e.* qu'il sort d'un domaine de sûreté, équivalent au domaine de contraintes en viabilité. En maintenance des systèmes, les outils viabilistes peuvent donc avoir un intérêt. Cet article utilise la notion de noyau de fiabilité introduite par [Rougé 14] qui s'apparente au noyau de viabilité stochastique. On construit alors une relation de récurrence sur la probabilité de défaillance cumulée afin de calculer ce noyau de fiabilité. La conception fiable d'une poutre soumise à la corrosion est présentée. Dans ce problème, on cherche à savoir quel matériau possède les meilleures propriétés afin de résister à une force et une corrosion. Nous utilisons le volume du noyau de fiabilité comme un indicateur afin de choisir le matériau.

5.2 Présentation et contributions de l'article

Cet article a été présenté à la 8ème conférence IFAC : "Manufacturing Modelling, Management and Control" à Troyes (France) en juin 2016. Une version étendue pour soumission à une revue est en cours de rédaction. Les principales contributions

de ce papier sont les suivantes :

- nous illustrons les liens entre la théorie de la viabilité et la théorie de la fiabilité en utilisant des outils de viabilité comme la programmation dynamique et le noyau de viabilité dans un cadre fiabiliste. On calcule ainsi un noyau de fiabilité qui est l'ensemble des états initiaux du système pour lesquels la probabilité de défaillance est inférieure à un certain seuil pour un horizon de temps donné ;
- nous utilisons le noyau de fiabilité dans un problème de conception fiable. Il s'agit de déterminer quel matériau est le plus à même d'être utilisé dans le cas d'un problème mécanique afin qu'il conserve au maximum ses propriétés pendant une période donnée.

5.3 Conclusions

Cet article est un exemple d'application des outils de la viabilité dans le champs de la fiabilité des systèmes. Nous utilisons le noyau de fiabilité comme un outil normatif en conception fiable. On s'assure ainsi d'avoir une probabilité de défaillance minime si, à la conception, le système est dans le noyau de fiabilité. De la même manière que la théorie de la fiabilité permet de résoudre des problèmes viabilistes, la théorie de la viabilité permet de lever des verrous et d'apporter des outils normatifs pour des problèmes de fiabilité dépendants du temps.

5.4 Texte de l'article

Computing the reliability kernel of a time-variant system: Application to a corroded beam

A. Brias * **J-D. Mathias** ** **G. Deffuant** ***

* *Irstea, UR LISC Laboratoire d'ingénierie des systèmes complexes, Aubière, France (e-mail: antoine.brias@irstea.fr)*
** *Irstea, UR LISC Laboratoire d'ingénierie des systèmes complexes, Aubière, France (e-mail: jean-denis.mathias@irstea.fr)*
*** *Irstea, UR LISC Laboratoire d'ingénierie des systèmes complexes, Aubière, France (e-mail: guillaume.deffuant@irstea.fr)*

Abstract: Time-variant reliability analysis aims at assessing the probability of failure of a time-variant system within a given time horizon. We illustrate in this paper the computation of the reliability kernel which is the set of initial states for which the probability of failure remains under a threshold within the considered time horizon. This paper supposes that the time-variant system is discrete in time and space with given probabilities of transition between space states. We use a recursive relation for computing the cumulative probability of failure of the system, linking the probability of failure at time t with the probability of being at a given state x (for all possible states) at time $t-1$. Applying this relation, it is possible to compute the probability of failure at any starting point in the state space and hence to derive the reliability kernel. The computation of this kernel gives informations about the system which can be further helpful in reliable design. The approach is illustrated on an example of a steel beam under corrosion.

Keywords: Discrete systems, Dynamic systems, Initial states, Reliability analysis, Reliability kernel, System failures, Transition matrix, Reliable design

1. INTRODUCTION

The reliability analysis aims at computing the probability of failure of a system with respect to a defined failure measure and taking into account the uncertainties in the system's properties or context. Reliability theory comes from the field of industrial engineering (Banerjee, 1965; Barlow, 1984), but it is widely applied in different fields, from ecology (Maier et al., 2001) to industrial maintenance (Cazuguel and Cognard, 2006). This paper focuses on time-variant reliability which addresses the case of systems evolving in time.

Methods specifically developed for time-variant processes are mainly based on the time integration of the out-crossing rate, i.e. the rate at which the state may go through the limit state surface (Li and Der Kiureghian, 1997). Some algorithms follow a decomposition of the time-variant problem into a series of time-invariant ones (Hagen and Tvedt, 1991), leading to methods such as PHI2 (Andrieu-Renaud et al., 2004; Sudret, 2008).

We consider a time-variant system that is discrete in time and space, with given probability transitions between the space states (it is a Markovian process when these transition probabilities are constant in time) and we derive a recursive algorithm computing the failure probability within a time horizon from any starting point of the state space. This approach comes from the connection between time-variant reliability and viability theory (Aubin and Saint-Pierre, 2007) recently established in Rougé et al. (2014). It finally provides the reliability kernel, namely the set of all initial states for which probability of failure within the time horizon is below a given threshold. We propose to show an application of the computation of the reliability kernel in this paper, and its possible use in reliable design.

The paper is organized as follows. Section 2 introduces time-variant reliability notions such as the probability of failure, the reliability of the system and the reliability kernel. Then Section 3 shows the recursive algorithm built in order to get an approximation of this reliability kernel. After that, Section 4 proposes an application of a corroding steel beam in order to illustrate the approach. Finally, Section 5 concludes the paper.

2. RELIABILITY KERNEL

In this section, after introducing the concept of cumulated probability of failure, we present the notion of reliability kernel.

2.1 The stochastic system

Uncertainty is represented at each date by the random vector $\mathbf{W}(t)$. $\mathbf{W} = (\mathbf{W}(0), \mathbf{W}(1), ..., \mathbf{W}(T-1))$ is called a scenario and belongs to the set of all scenarios $\mathbb{S}$. The state of the system is a then a random vector $\mathbf{X}(t)$. In discrete time, we assume the following state transition between two consecutive dates t and $t+1$:

$$\mathbf{X}(t+1) = f(\mathbf{X}(t), \mathbf{W}(t)) \quad (1)$$

For the rest of this paper, we use the notation with realizations $x(t)$ of $\mathbf{X}(t)$ and realizations $w(t)$ of $\mathbf{W}(t)$:

$$f(x(t), w(t)) = x(t+1) \quad (2)$$

⋆ Sponsor and financial support acknowledgment goes here. Paper titles should be written in uppercase and lowercase letters, not all uppercase.

for which $x(t)$ is a realization of $\mathbf{X}(t)$, $w(t)$ is a realization of $\mathbf{W}(t)$, and $x(t+1)$ is a realization of $\mathbf{X}(t+1)$.

The initial state is noted :

$$x_0 = x(0) \tag{3}$$

Contrary to Andrieu-Renaud et al. (2004) and Sudret et al. (2002), we made the assumption that the exact initial state x_0 of the system is known.

The system is discrete in space, the possible states being indexed by $i = \{1, .., N\}$. It is then possible to derive from Eq.1 the probability transitions between discrete states, and the probability that the system is at state x_j at time $t+1$ from probabilities of being at any state x_i ($i = \{1, .., N\}$) at time t:

$$\mathbb{P}(x(t+1) = x_j) = \sum_{i=\{1,..,N\}} \theta_{i,j} \mathbb{P}(x(t) = x_i). \tag{4}$$

The matrix $\theta_{i,j}$ is obtained from Eq.1 by:

$$\theta_{i,j} = \mathbb{P}(f(x_i, w(t)) = x_j). \tag{5}$$

When the matrix $\theta_{i,j}$ is constant in time, the system follows a Markovian process.

At each step of time t, the limit state surface $g(t, \mathbf{X}(t)) = 0$ separates the failure domain $F(t)$ and the survival domain $S(t)$.

2.2 Probability of failure and reliability

The definition of the cumulative probability of failure $P_{f,c}$ between two dates t_0 and t_1 ($0 \leq t_0 \leq t_1 \leq T$) and knowing the state x_0 at time t_0 is:

$$P_{f,c}(t_0, t_1, x_0) = \mathbb{P}(\exists t \in \{t_0, t_0+1, ..., t_1\}, \mathbf{X}(t) \in F(t) \,|\, \mathbf{X}(t_0) = x_0) \tag{6}$$

This is the probability that at least one failure occurs during the planning period, knowing the initial state of the system. A distinction is made here between this cumulative probability of failure and the concept of instantaneous probability of failure. The former is calculated all over an interval of time, whereas the latter is computed by freezing time in the limit state function (Andrieu-Renaud et al., 2004).

The complement of the cumulative probability of failure is called reliability or probability of safety P_s:

$$P_s(t_0, t_1, x_0) = \mathbb{P}(\forall t \in \{t_0, t_0+1, ..., t_1\}, \mathbf{X}(t) \in S(t) \,|\, \mathbf{X}(t_0) = x_0) \tag{7}$$

This is the probability that the system belongs to the survival set during the whole planning period. The cumulated failure probability and the safety probability are linked by :

$$P_s(t_0, t_1, x_0) = 1 - P_{f,c}(t_0, t_1, x_0) \tag{8}$$

2.3 Reliability kernel

The cumulative probability of failure $P_{f,c}(0, T, x_0)$ is the probability for a trajectory $(x_0, x(1), ..., x(T))$ to leave the survival set over the planning period $[0, T]$. The set of all reliable initial states at a significance level α is called the reliability kernel :

$$Rel(\alpha, T) = \{x_0 \in S(0) \,|\, P_{f,c}(0, T, x_0) \leq \alpha\} \tag{9}$$

It is important to note that $P_{f,c}(0, T, x_0)$ depends on the time horizon and the initial state x_0. Moreover, we rewrite the definition of the reliability kernel by using the safety probability P_s defined in Eq.7 :

$$Rel(\alpha, T) = \{x_0 \in S(0) \,|\, P_s(0, T, x_0) \geq 1 - \alpha\} \tag{10}$$

We aim at computing the set $Rel(\alpha, T)$ by a recursive algorithm. We focus on the computation of the safety probability for the case where $x_0 \in S(0)$. The concept of reliability kernel was first proposed in Rougé et al. (2014). Note that contrary to this paper, we don't consider a controlled system here.

3. APPROXIMATION OF THE RELIABILITY KERNEL

The computation is made all over a grid of the possible initial states.

3.1 Recursion on the safety probability

We are looking for a backward recursive relation between $P_s(t, T, x_i)$ and the safety probability $P_s(t+1, T, x_k)$ one time step ahead, denoting x_i a realization of $\mathbf{X}(t)$ and x_k a realization of $\mathbf{X}(t+1)$. According to Eq.7, $\forall t \in \{0, 1, ..., T-1\}$, $\forall x_i \in S(t)$:

$$P_s(t, T, x_i) = \mathbb{P}(\forall \tau \in \{t, t+1, ..., T\}, X(\tau) \in S(\tau) \,|\, X(t) = x_i) \tag{11}$$

Starting at state x_i at t and knowing that $x_i \in S(t)$, we write $P_s(t, T, x_i)$ as the probability to go from x_i to another state x_k which is in $S(t+1)$ and having a safety probability $P_s(t+1, T, x_k)$:

$$P_s(t, T, x_i) = \sum_{x_k \in S(t+1)} \theta_{i,k} P_s(t+1, T, x_k) \tag{12}$$

According to Eq.8, we now recursively express $P_{f,c}(t, T, x_i)$ regarding $P_{f,c}(t+1, T, x_k)$:

$$P_{f,c}(t, T, x_i) = 1 - \sum_{x_k \in S(t+1)} \theta_{i,k} \left(1 - P_{f,c}(t+1, T, x_k)\right) \tag{13}$$

Once we have this result, we recursively compute the approximation of the cumulated probability of failure all over the states. The reliability kernel is derived directly because state x_k belongs to the reliability kernel if the cumulated probability of failure $P_{f,c}(t, T, x_k)$ is below the threshold α. Note that Eq.12 leads to the recursive equation used in stochastic viability theory (Rougé et al., 2014).

4. APPLICATION

We use an adapted and simplified model from Sudret et al. (2002) to present our results.

4.1 Beam under loading

Let us consider a steel bending beam with a rectangular cross-section. Its length is $L = 5$ m and we note b_0 for the initial breadth and h_0 for the initial height of the beam. It is submitted to a dead load (noted $\rho_{st} = 7850$ kg/m the steel mass density, this load is $p = \rho_{st} b_0 h_0$ (N/m)) as well as a pinpoint load F applied onto its middle point (see Fig.1).

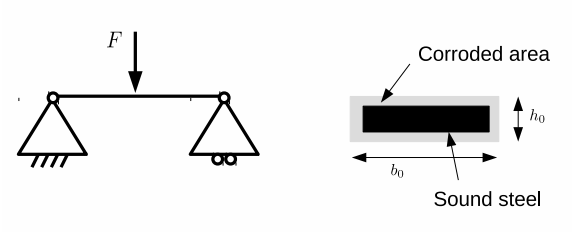

Fig. 1. Corroded bending beam submitted to dead loads, adapted from Sudret, Defaux, & Andrieu 2002.

The maximum of the bending moment is reached at this point at time t :

$$M = \frac{FL}{4} + \frac{\rho_{st} b(t) h(t) L^2}{8} \tag{14}$$

with $F = 3500\,\mathrm{N}$.

By supposing that the elastic plastic constitutive law of the beam is perfect, and denoting the yield stress by σ_e, the ultimate bending moment of the rectangular section is :

$$M_u(t) = \frac{b(t) h(t)^2 \sigma_e}{4} \tag{15}$$

Here, we take $\sigma_e = 240$ MPa.

4.2 Corrosion dynamic

The steel beam is supposed to corrode with time. Here we assume no measures are taken to slow down corrosion. The only protection lies in the judicious choice of the beam material as discussed in section 4.5. The phenomenon of corrosion begins at $t = 0$. Like in Andrieu-Renaud et al. (2004), we assume that the corrosion is isotropic all around the section of the beam and assuming that it leads to the lost of all mechanical stiffness of corroded areas, the sound cross-section entering at time t reads :

$$s(t) = b(t).h(t) \tag{16}$$

with the dynamics for b:

$$b(t+1) = b(t) - 2\kappa \tag{17}$$

and the dynamics for h:

$$h(t+1) = h(t) - 2\kappa \tag{18}$$

The initial state $x_0 = (b(0), h(0))$ of the system belongs to $[0.07, 0.08] \times [0.03, 0.04]$. We make the assumption that the corrosion kinetics is controlled by κ following a log-normal distribution, with a mean of 0.05×10^{-3} and a coefficient of variation equals to 10%.

Using the above notation, the beam fails at a given point in time if $M > M_u(t)$, for which a plastic hinge appears in the middle of the span.

4.3 Reliability problem statement

The limit state function associated with the failure of the beam reads :

$$g(\mathbf{X}(t)) = (M_u(\mathbf{X}(t)) - M(\mathbf{X}(0))) \tag{19}$$

Here, $\mathbf{X}(t) = (b(t), h(t))$. The time interval under consideration is $[0, 20]$ years. We aim to compute $Rel(0.05, 20)$, the set

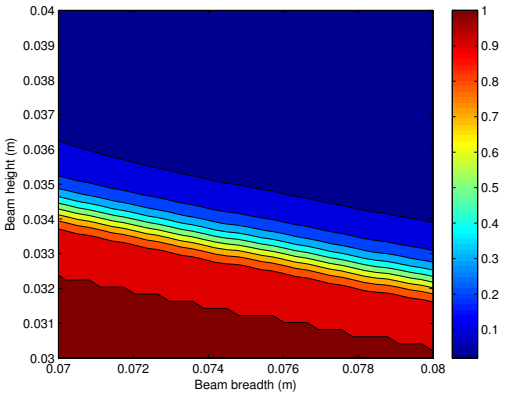

Fig. 2. Contour plot of the cumulated failure probability using our recursive algorithm. The color represents the cumulated failure probability for each initial states of the domain.

of all reliable initial states at the significance level 0.05 for the planning period. The survival domain is defined by :

$$S(t) = \{(b(t), h(t)) \,|M_u((b(t), h(t))) - M > 0\} \tag{20}$$

By using Eq.14 and Eq.15, it leads to a constraint relation between $b(t)$ and $h(t)$:

$$S(t) = \left\{ (b(t), h(t)) \,|b(t) > \frac{2FL}{2\sigma_e h(t)^2 + \rho_{st} h(t) L^2} \right\} \tag{21}$$

4.4 Results of the approximation

The result of the computation of the cumulated failure probability for each $(b_0, h_0) \in [0.07, 0.08] \times [0.03, 0.04]$ is shown in Fig.2. The recursive algorithm is used, computing the cumulative failure probabilities all over a bidimensional grid of 50×50 points. Thus, an initial state for which the cumulative failure probability is under the threshold 0.05 belongs to the reliability kernel. A marked border is observed between initial states which are reliable and those which have a high risk of failure.

From this map of cumulated failure probability, we extract the reliability kernel $Rel(0.05, 20)$, shown in Fig.3. If the initial state of the beam is inside the kernel, the reliability of the beam is insured at 95% during all the planning period. At the contrary, if the initial characteristics of the beam are out of the reliability kernel, the beam will certainly fail during the planning period due to the corrosion phenomenon. There is a moment when the constraint of Eq.21 is no longer fulfilled.

During the computation, the probability to go from one state to another are stored. We can then calculate the possible trajectories starting from a given initial state, following the dynamics, as shown in Fig.3 for the initial state (0.0769,0.0382).

Other informations can be extracted from the reliability kernel computation. We show the cumulated failure probability versus the time for the initial state : $x_0 = (0.0769, 0.0382)$ in Fig.4. From $t = 4$, the risk of failure will rapidly increase. The state of the beam is altered too much by the corrosion. After this date, the thickness of the beam is such that the probability of

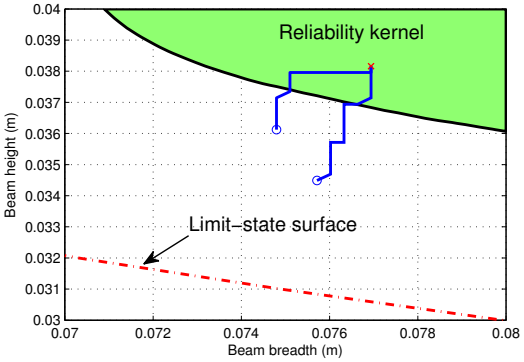

Fig. 3. Reliability kernel at a significance level of 0.05 for a planning period of 20 years. Every initial state inside the reliability kernel has a cumulated failure probability below 0.05. Two trajectories starting from the state (0.0769,0.0382) are shown. The blue circles symbolize the state of the beam after 20 years of corrosion. The red dotted line represents the limit-state surface.

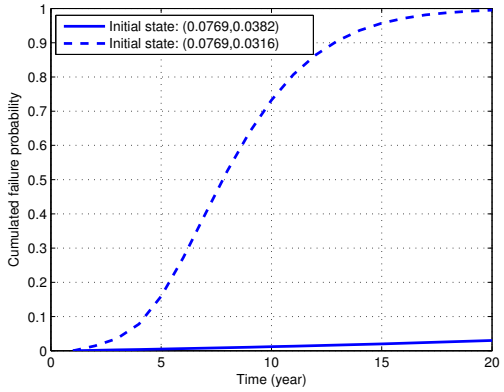

Fig. 4. Cumulated failure probability for two different initial states : $(0.0769, 0.0382)$ and $(0.0769, 0.0316)$. The first one belongs to the reliability kernel $Rel\,(0.05, 20)$. The second one is outside the kernel.

failure is significant and near the end of the planning period ($t > 13$, $P_{f,c}(0, t, x) > 0.9$). On the contrary, If the beam is manufactured to have its initial properties in the reliability kernel, the cumulated failure probability will not exceed the 0.05 threshold. It is the case for a beam starting at $x_0 = (0.0769, 0.0316)$.

4.5 Reliable design

The reliability kernel can be used to find which material is best suited to a given reliability problem. This field of research, the reliable design, has applications in aeronautics, civil engineering, transportation (Mathias and Lemaire, 2013).

We study here the comparison between two materials: the steel which was previously used, and a stainless steel. For this last, the mass density is $\rho_{st} = 8000$ kg/m. The corrosion

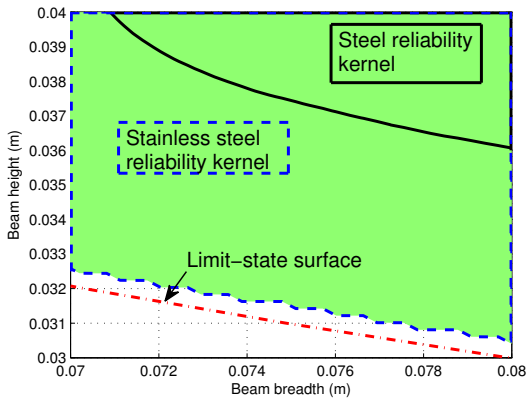

Fig. 5. Reliability kernels ($Rel\,(0.05, 20)$) for two different materials: a steel beam (a black line represents the contour of $Rel\,(0.05, 20)$) is shown and a stainless steel beam (the blue dotted-line represents the contour of $Rel\,(0.05, 20)$). The red dotted line represents the limit-state surface.

kinetics is significantly downgraded, with the same log-normal distribution, except that it has a mean of 6×10^{-7}.

The results in Fig.5 shown the efficiency of the stainless steel compared to a classical steel. For these two materials, the reliability kernel $Rel\,(0.05, 20)$ was computed. Most of the initial states are reliable for a stainless beam. Moreover, the reliability kernel of the classical steel beam is completely included into the reliability kernel of the stainless steel beam. It is therefore relevant to use a stainless steel beam matching the initial specifications.

5. CONCLUSION

In the reliability kernel approach, we reason on a set of states instead of thinking about the trajectories. This paper establishes a recursive relation for computing the probability of failure from starting state within a time horizon, from which reliability kernels can immediately derived. This approach is similar to the recursive algorithm in stochastic viability, and shares the same drawbacks. In case of high-dimension problems, discretizing the state space often exceeds of the available memory space. Parallel computing techniques such as GPU may mitigate these difficulties, since the algorithm can easily be broken down into a large number of parallel processes.

Nevertheless, this first step opens new research directions. In particular, we are currently aiming at extending this algorithm to controlled dynamics. This requires to include an optimization step on the possible control values. Moreover, we consider to develop a version of the algorithm based on the PHI2 method, in order to take into account the auto-correlation of the dynamics.

As a perspective, we aim to propose to use the reliability kernel as a tool providing reliability standards or norms for dynamics systems. An initial design of a system being in the reliability kernel is insured to be reliable at a defined significance level over the planning period. By comparing the size of the reliability kernel, it allows to choose a well-suited material for the given reliability problem.

ACKNOWLEDGEMENTS

This work was supported by grants from Irstea and Région Auvergne. This support is gratefully acknowledged.

REFERENCES

Andrieu-Renaud, C., Sudret, B., and Lemaire, M. (2004). The PHI2 method: A way to compute time-variant reliability. *Reliability Engineering and System Safety*, 84, 75–86. doi: 10.1016/j.ress.2003.10.005.

Aubin, J.P. and Saint-Pierre, P. (2007). An Introduction to Viability Theory and Management of Renewable Resources. In *Advanced Methods for Decision Making and Risk Management in Sustainability Science*, 43–93.

Banerjee, S.K. (1965). Mathematical Theory of Reliability. Richard E. Barlow and Frank Proschan. With contributions by Larry C. Hunter. Wiley, New York, 1965. xiv + 256 pp. Illus. $11. *Science*, 148(3674), 1208–1209. doi: 10.1126/science.148.3674.1208-a.

Barlow, R.E. (1984). Mathematical Theory of Reliability : A Historical Perspective. *IEEE TRANSACTIONS ON RELIABILITY*, R-33(1), 16–20. doi:10.1109/TR.1984.6448269.

Cazuguel, M. and Cognard, J.Y. (2006). Development of a time-variant reliability approach for marine structures subjected to corrosion. *Proc. 3rd Int. {ASRANet} Colloquium, Glasgow, United Kingdom*, (July).

Hagen, O. and Tvedt, L. (1991). Vector process out-crossing as parallel system sensitivity measure. *Journal of Engineering Mechanics*, 117(10), 2201–2220.

Li, C.C. and Der Kiureghian, A. (1997). Large mean out crossing of nonlinear response to stochastic input. In *Engineering Probabilistic Design and Maintenance for Flood Protection*, 141–160.

Maier, H.R., Lence, B.J., Tolson, B.a., and Foschi, R.O. (2001). First-order reliability method for estimating reliability, vulnerability, and resilience. *Water Resources Research*, 37(3), 779–790. doi:10.1029/2000WR900329.

Mathias, J.D. and Lemaire, M. (2013). Reliability analysis of bonded joints with variations in adhesive thickness. *Journal of Adhesion Science and Technology*, 27(10), 1069–1079. doi:10.1080/01694243.2012.727176.

Rougé, C., Mathias, J.D., and Deffuant, G. (2014). Relevance of control theory to design and maintenance problems in time-variant reliability: The case of stochastic viability. *Reliability Engineering & System Safety*, 132, 250–260. doi: 10.1016/j.ress.2014.07.025.

Sudret, B., Defaux, G., and Andrieu, C. (2002). Comparison of methods for computing the probability of failure in time-variant reliability using the outcrossing approach. *Fourth International Conference on Computational Stochastic Mechanics*.

Sudret, B. (2008). Analytical derivation of the outcrossing rate in time-variant reliability problems. *Structure and Infrastructure Engineering*, 4(5), 353–362. doi: 10.1080/15732470701270058.

Chapitre 6

Conclusion

6.1 Bilan

Durant cette thèse nous avons exploré différentes pistes afin de limiter les conséquences de la malédiction de la dimension sur le temps et les besoins en mémoire du calcul de noyau de viabilité. L'intérêt principal de ces travaux est de proposer des outils permettant le calcul de noyau de viabilité en grandes dimensions. Les outils développés accélèrent le calcul des transitions et réduisent l'espace-mémoire utilisé par l'algorithme de programmation dynamique. Ainsi, nous avons tout d'abord accéléré le calcul des transitions, en parallélisant sur carte graphique cette étape. Ce premier travail a mis en lumière les goulots d'étranglement de l'algorithme de programmation dynamique pour le calcul de noyau de viabilité. En particulier, le calcul de la matrice de transition demande un espace de stockage dépendant quadratiquement du nombre d'états de la grille utilisée dans le cas stochastique. Une solution couplant programmation dynamique et méthodes issues du domaine de la fiabilité a été proposée afin de s'affranchir partiellement du calcul de cette matrice de transition. L'espace-mémoire utilisé n'est plus que linéaire au lieu d'être quadratique avec le nombre d'états de la grille. L'algorithme proposé donne une carte des contrôles proche de celle calculée en programmation dynamique classique. Cette carte des contrôles est alors utilisée pour reconstruire le noyau de viabilité. Ne pas calculer la matrice de transition complète permet le calcul de noyau de viabilité pour des systèmes de plus grande dimension qu'actuellement, avec des ressources en temps et en espace-mémoire raisonnables.

La parallélisation sur carte graphique et l'utilisation de méthodes fiabilistes permettent d'aborder des problèmes qui n'étaient pas encore traitables, notamment en gestion de systèmes environnementaux, qui restent généralement de grande dimension. Le calcul parallèle sur carte graphique a été utilisé dans le cadre de la gestion de pêcherie afin de traiter des problèmes en 6 dimensions. Ce calcul était limité par l'espace mémoire de la carte graphique utilisée. L'algorithme couplant programmation dynamique et techniques fiabilistes a été utilisé pour un problème d'eutrophisation en 5 dimensions, incorporant de l'incertitude sur chaque dimen-

sion. C'est un exemple typique de cas où le calcul de la matrice de transition est infaisable en pratique, mais où l'utilisation d'approximations de la fonction valeur permet de donner de premiers résultats exploitables. Cependant, ces modèles restent plutôt théoriques, et la calibration d'un modèle d'eutrophisation basé sur les données du lac du Bourget est un premier pas pour l'utilisation des techniques développées dans cette thèse pour des problèmes réels. Enfin, l'utilisation du noyau de viabilité stochastique en conception fiable illustre un autre domaine d'application de la théorie de la viabilité stochastique et constitue une première ouverture en ingénierie.

6.2 Perspectives

Les solutions données dans ces travaux n'ont pas vocation à être définitives. Elles sont l'aboutissement de nos recherches dans le cadre de cette thèse, mais elles sont aussi le point de départ pour de futures améliorations pour le calcul de noyau de viabilité. D'autres pistes existent afin de repousser la malédiction de la dimension. Ainsi du point de vue technique, des travaux sont en cours, afin d'utiliser des bases de données à l'aide de l'outil Hadoop [Shvachko 10] pour gérer la mémoire demandée par l'algorithme de programmation dynamique. Une fois cette étape terminée, le couplage de l'utilisation des bases de données et de la parallélisation donnera la possibilité de traiter des problèmes en plus grandes dimensions.

Afin d'accélérer les calculs, nous avons utilisé les techniques de parallélisation multicoeur directement intégrées dans le logiciel MATLAB® dans lequel a été implémenté l'algorithme utilisant les méthodes fiabilistes. Les appels à la méthode FORM pourraient être parallélisés sur carte graphique comme dans le **chapitre 2** afin de diminuer le temps d'exécution par rapport à la parallélisation multicoeur. Les deux axes étudiés dans ce travail peuvent donc être rapprochés.

L'algorithme présenté dans le **chapitre 3** calcule des probabilités de transition d'états vers des ensembles d'états. La frontière de ces ensembles d'états pourrait être approchée par un méta-modèle afin de gagner en espace-mémoire. Le méta-modèle est une fonction imitant le comportement d'une autre, tout en étant beaucoup moins coûteux à évaluer (comme c'est le cas dans la méthode FORM, où une approximation linéaire de la fonction d'état limite est effectuée). La théorie de la fiabilité utilise différents méta-modèles comme les approximations polynomiales, les chaos polynomiaux ou le krigeage [Sudret 12]. Ici, une procédure de classification comme les SVMs ou un réseau de neurones pourrait être un méta-modèle pertinent. L'utilité de ces méta-modèles a déjà été démontrée en théorie de la fiabilité [Goh 03, Bucher 08] mais aussi en viabilité déterministe [Djeridane 08b]. Après une étape d'apprentissage basée sur des points d'échantillonnage dans l'espace d'état, une approximation de la frontière d'un ensemble d'états serait calibrée.

Elle serait alors utilisée pour déterminer les probabilités de transition à l'aide de techniques comme FORM comme dans le **chapitre 3**. L'avantage d'étudier des solutions de méta-modèles ne concerne pas seulement l'approximation de la fonction valeur. Il est possible de faire apprendre à des SVM ou des réseaux de neurones la carte de contrôles, au lieu d'avoir en mémoire le contrôle optimal pour chaque état. Empiriquement, comme celle-ci varie peu avec le temps, la même approximation peut être utilisée et mise à jour à chaque pas de temps, selon les changements de contrôle optimal associé aux points d'échantillonnage. L'inconvénient des ces méthodes est que si elle sont générales, il est en revanche difficile de se passer d'une grille ou d'une discrétisation de l'espace d'état.

Les méthodes développées dans cette thèse demandent peu d'information sur le système étudié. Si le type d'aléa est connu, et qu'il est possible de séparer les incertitudes entre des incertitudes normalisées et des événements extrêmes, le concept de *noyau de viabilité garanti* (ou tychastique) est intéressant à étudier [Aubin 07]. Cet ensemble contient tous les états initiaux du domaine de contraintes K pour lesquels il existe une suite de contrôles maintenant le système dans le domaine de contraintes K quelques soient les incertitudes normalisées. Cet ensemble est calculable avec des outils proches de la viabilité déterministe.

En dehors du calcul de noyau de viabilité dans un contexte viabiliste, les méthodes et concepts présentés dans cette thèse sont exploitables, que ce soit en programmation dynamique ou en fiabilité des systèmes. Le **chapitre 5** a présenté une ouverture en ingénierie avec le calcul d'un noyau de fiabilité en conception fiable. Néanmoins, le concept de noyau de fiabilité est récent et il serait intéressant d'établir des relations entre cet objet et d'autres outils fiabilistes comme le taux de défaillance ou le temps moyen entre pannes, des indicateurs utilisés en maintenance [Kargahi 14].

Finalement, ce travail est une illustration de l'importance de croiser les domaines de recherche pour conjurer la malédiction de la dimension. Cela donne des combinaisons efficaces d'outils de différentes disciplines et il paraît pertinent de continuer cette approche interdisciplinaire dans le futur afin d'améliorer les méthodes de calcul de noyau de viabilité en grande dimension.

Bibliographie

[Alvarez 10] Isabelle Alvarez, Sophie Martin & Salma Mesmoudi. *Describing the result of a classifier to the end-user : Geometric-based sensitivity*. In Frontiers in Artificial Intelligence and Applications, volume 215, pages 835–840, 2010.

[Alvarez 11] Isabelle Alvarez & Sophie Martin. *Geometric robustness of viability kernels and resilience basins*. In Understanding Complex Systems, volume 2011, pages 193–218. 2011.

[Aubin 90] Jean-Pierre Aubin. *A Survey of Viability Theory*. SIAM Journal on Control and Optimization, vol. 28, no. 4, pages 749–788, 1990.

[Aubin 01] Jean Pierre Aubin, Noël Bonneuil, Franck Maurin & Patrick Saint-Pierre. *Viability of pay-as-you-go systems*. Journal of Evolutionary Economics, vol. 11, no. 5, pages 555–571, 2001.

[Aubin 07] Jean-Pierre Aubin & Patrick Saint-Pierre. *An Introduction to Viability Theory and Management of Renewable Resources*. In Advanced Methods for Decision Making and Risk Management in Sustainability Science, pages 43–93. 2007.

[Bellman 59] R. Bellman & R. Kalaba. *On adaptive control processes*. IEE Transactions on Automatic Control, vol. 4, no. 2, pages 1–9, 1959.

[Bernard 11] Claire Bernard. *La théorie de la viabilité au service de la modélisation mathématique du développement durable Application au cas de la forêt humide de Madagascar*. PhD thesis, 2011.

[Bernard 13] C Bernard & S Martin. *Comparing the sustainability of different action policy possibilities : application to the issue of both household survival and forest preservation in the corridor of Fianarantsoa.* Mathematical biosciences, vol. 245, no. 2, pages 322–30, oct 2013.

[Bokanowski 06] O. Bokanowski, S. Martin, R. Munos & H. Zidani. *An anti-diffusive scheme for viability problems*. Applied Numerical Mathematics, vol. 56, no. 9, pages 1147–1162, sep 2006.

[Bonneuil 06] Noël Bonneuil. *Computing the viability kernel in large state dimension*. Journal of Mathematical Analysis and Applications, vol. 323, no. 2, pages 1444–1454, nov 2006.

[Bonneuil 14] Noël Bonneuil. *Emotions as dynamic systems in viability sets*. Mathematical and Computer Modelling of Dynamical Systems, no. November 2014, pages 1–20, oct 2014.

[Bucher 08] Christian Bucher & Thomas Most. *A comparison of approximate response functions in structural reliability analysis*. Probabilistic Engineering Mechanics, vol. 23, no. 2-3, pages 154–163, apr 2008.

[Cardaliaguet 00] Pierre Cardaliaguet, Marc Quincampoix & Patrick Saint-Pierre. *Numerical Schemes for Discontinuous Value Functions of Optimal Control*. Set-Valued Analysis, vol. 8, no. 1/2, pages 111–126, 2000.

[Carpenter 99] S.R. Carpenter, D. Ludwig & W.A. Brock. *Management of Eutrophication for Lakes Subject to Potentially Irreversible Change*. Ecological Applications, vol. 9, no. 3, pages 751–771, 1999.

[Chapel 07] Laetitia Chapel, Sophie Martin & Guillaume Deffuant. *Lake eutrophication : Using resilience evaluation to compute sustainable policies*, sep 2007.

[Chapel 08] Laetitia Chapel, Guillaume Deffuant, Sophie Martin & Christian Mullon. *Defining yield policies in a viability approach*. Ecological Modelling, vol. 212, no. 1-2, pages 10–15, mar 2008.

[De Lara 09] M De Lara & V Martinet. *Multi-criteria dynamic decision under uncertainty : a stochastic viability analysis and an application to sustainable fishery management.* Mathematical biosciences, vol. 217, no. 2, pages 118–124, feb 2009.

[DeAngelis 89] D. L. DeAngelis, S. M. Bartell & a. L. Brenkert. *Effects of Nutrient Recycling and Food-Chain Length on Resilience*. The American Naturalist, vol. 134, no. 5, page 778, 1989.

[Deffuant 07] G. Deffuant, L. Chapel & S. Martin. *Approximating Viability Kernels With Support Vector Machines*. IEEE Transactions on Automatic Control, vol. 52, no. 5, pages 933–937, may 2007.

[Deffuant 11] G Deffuant & N Gilbert. Viability and Resilience of Complex Systems : Concepts, Methods and Case Studies from Ecology and Society. 2011.

[Ditlevsen 07] O Ditlevsen & H O Madsen. *Structural Reliability Methods*. no. September, page 360, 2007.

[Djeridane 08a] B Djeridane, E Cruck & J Lygeros. *Randomized Algorithm : A viability computation.* pages 11548–11553, 2008.

[Djeridane 08b] B Djeridane & J Lygeros. *Approximate Viability using Quasi-Random Samples and a Neural Network Classifier.* In The International Federation of Automatic Control, numéro 2, pages 14342–14347, 2008.

[Domenech 11] Pablo Andres Domenech, Patrick Saint-Pierre & Georges Zaccour. *Forest Conservation and CO2 Emissions : A Viable Approach.* Environmental Modeling and Assessment, vol. 16, no. 6, pages 519–539, 2011.

[Doyen 10] Luc Doyen & Michel De Lara. *Stochastic viability and dynamic programming.* Systems & Control Letters, vol. 59, no. 10, pages 629–634, oct 2010.

[Goh 03] Anthony T C Goh & Fred H Kulhawy. *Neural network approach to model the limit state surface for reliability analysis.* vol. 1244, pages 1235–1244, 2003.

[Jacob 10] Arpith Chacko Jacob. *Parallelization of dynamic programming recurrences in computational biology.* PhD thesis, 2010.

[Kargahi 14] Mehdi Kargahi. *System Reliability Concepts.* vol. 2006, no. Chapter 2, pages 1–17, 2014.

[Krawczyk 13a] Jacek B. Krawczyk, Alastair Pharo, Oana S. Serea & Stewart Sinclair. *Computation of viability kernels : a case study of bycatch fisheries.* Computational Management Science, vol. 10, no. 4, pages 365–396, oct 2013.

[Krawczyk 13b] Jacek B. Krawczyk & Alastair S. Pharo. *Viability theory : an applied mathematics tool for achieving dynamic systems' sustainability.* Mathematica Applicanda, vol. 41, no. 1, pages 1–27, 2013.

[Lemaire 10] Maurice Lemaire, Alaa Chateauneuf & Jean-Claude Mitteau. Structural Reliability. Wiley, 2010.

[Maidens 09] John N Maidens, Shahab Kaynama, Ian M Mitchell, Meeko M K Oishi & Guy A Dumont. *Lagrangian methods for approximating the viability kernel in high-dimensional systems.* pages 1–16, 2009.

[Maier 01] Holger R. Maier, Barbara J. Lence, Bryan a. Tolson & Ricardo O. Foschi. *First-order reliability method for estimating reliability, vulnerability, and resilience.* Water Resources Research, vol. 37, no. 3, pages 779–790, 2001.

[Martin 04] Sophie Martin. *The Cost of Restoration as a Way of Defining Resilience : a Viability Approach Applied to a Model of Lake Eutrophication*. vol. 9, no. 2, 2004.

[Mathias 13] Jean-Denis Mathias & Maurice Lemaire. *Reliability analysis of bonded joints with variations in adhesive thickness*. Journal of Adhesion Science and Technology, vol. 27, no. 10, pages 1069–1079, 2013.

[Mathias 15] Jean-Denis Mathias, Bruno Bonté, Thomas Cordonnier & Francis de Morogues. *Using the Viability Theory to Assess the Flexibility of Forest Managers Under Ecological Intensification*. Environmental Management, no. JUNE, 2015.

[Mitchell 05] Ian M. Mitchell, Alexandre M. Bayen & Claire J. Tomlin. *A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games*. IEEE Transactions on Automatic Control, vol. 50, no. 7, pages 947–957, 2005.

[Mullon 04] C Mullon, P Cury & L Shannon. *Viability model of Trophic Interactions in Marine Ecosystem*. Natural Resource Modelling, vol. 17, no. 1, pages 27–58, 2004.

[Naeem 98] Shahid Naeem. *Species redundancy and ecosystem reliability*. Conservation Biology, vol. 12, no. 1, pages 39–45, 1998.

[Powell 09] Warren B Powell. *What You Should Know About Approximate Dynamic Programming*. no. December 2008, 2009.

[Rackwitz 01] Rüdiger Rackwitz. *Reliability analysis-a review and some perspective*. Structural Safety, vol. 23, no. 4, pages 365–395, 2001.

[Rausand 98] M Rausand. *Reliability centered maintenance*. In Reliability Engineering & System Safety Engineering and System Safety, volume 60, pages 12–132. 1998.

[Rougé 13] Charles Rougé, Jean-Denis Mathias & Guillaume Deffuant. *Extending the viability theory framework of resilience to uncertain dynamics, and application to lake eutrophication*. Ecological Indicators, vol. 29, pages 420–433, jun 2013.

[Rougé 14] Charles Rougé, Jean-Denis Mathias & Guillaume Deffuant. *Relevance of control theory to design and maintenance problems in time-variant reliability : The case of stochastic viability*. Reliability Engineering & System Safety, vol. 132, pages 250–260, 2014.

[Saint-Pierre 94] Patrick Saint-Pierre. *MaA dlhem*. vol. 209, pages 187–209, 1994.

[Seube 00] N. Seube, R. Moitie & G. Leitmann. *Aircraft Take-Off in Windshear : A Viability Approach*. Set-Valued Analysis, vol. 8, no. 1/2, pages 163–180, 2000.

[Shvachko 10] Konstantin Shvachko, Hairong Kuang, Sanjay Radia & Robert Chansler. *The Hadoop distributed file system*. In 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010, 2010.

[Spiteri 00] Raymond J. Spiteri, Dinesh K. Pai & Uri M. Ascher. *Programming and control of robots by means of differential algebraic inequalities*. IEEE Transactions on Robotics and Automation, vol. 16, no. 2, pages 135–145, 2000.

[Sudret 12] Bruno Sudret. *Meta-models for structural reliability and uncertainty quantification*. mar 2012.

[Wei 12] W Wei, I Alvarez & S Martin. *Algorithme d'approximation du noyau de viabilité avec procédure de classification*. vol. 2012, pages 24–27, 2012.

Titre de la thèse

Conjurer la malédiction de la dimension dans le calcul du noyau de viabilité à l'aide de parallélisation sur carte graphique et de la théorie de la fiabilité : application à des dynamiques environnementales

Dispel the dimensionality curse in viability kernel computation with the help of GPGPU and reliability theory: application to environmental dynamics

Résumé

La théorie de la viabilité propose des outils permettant de contrôler un système dynamique afin de le maintenir dans un domaine de contraintes. Le concept central de cette théorie est le noyau de viabilité, qui est l'ensemble des états initiaux à partir desquels il existe au moins une trajectoire contrôlée restant dans le domaine de contraintes. Cependant, le temps et l'espace nécessaires au calcul du noyau de viabilité augmentent exponentiellement avec le nombre de dimensions du problème considéré. C'est la malédiction de la dimension. Elle est d'autant plus présente dans le cas de systèmes incorporant des incertitudes. Dans ce cas-là, le noyau de viabilité devient l'ensemble des états pour lesquels il existe une stratégie de contrôle permettant de rester dans le domaine de contraintes avec au moins une certaine probabilité jusqu'à l'horizon de temps donné. L'objectif de cette thèse est d'étudier et de développer des approches afin de combattre cette malédiction de la dimension. Pour ce faire, nous avons proposé deux axes de recherche : la parallélisation des calculs et l'utilisation de la théorie de la fiabilité. Les résultats sont illustrés par plusieurs applications. Le premier axe explore l'utilisation de calcul parallèle sur carte graphique. La version du programme utilisant la carte graphique est jusqu'à 20 fois plus rapide que la version séquentielle, traitant des problèmes jusqu'en dimension 7. Outre ces gains en temps de calcul, nos travaux montrent que la majeure partie des ressources est utilisée pour le calcul des probabilités de transition du système. Cette observation fait le lien avec le deuxième axe de recherche qui propose un algorithme calculant une approximation de noyaux de viabilité stochastiques utilisant des méthodes fiabilistes calculant les probabilités de transition. L'espace-mémoire requis par cet algorithme est une fonction linéaire du nombre d'états de la grille utilisée, contrairement à l'espace-mémoire requis par l'algorithme de programmation dynamique classique qui dépend quadratiquement du nombre d'états. Ces approches permettent d'envisager l'application de la théorie de la viabilité à des systèmes de plus grande dimension. Ainsi nous l'avons appliquée à un modèle de dynamique du phosphore dans le cadre de la gestion de l'eutrophisation des lacs, préalablement calibré sur les données du lac du Bourget. De plus, les liens entre fiabilité et viabilité sont mis en valeur avec une application du calcul de noyau de viabilité stochastique, autrement appelé noyau de fiabilité, en conception fiable dans le cas d'une poutre corrodée.

Mots clés : Théorie de la viabilité, Malédiction de la dimension, Parallélisation, Théorie de la fiabilité, Programmation dynamique, Systèmes environnementaux.

Keywords: Viability theory, Curse of dimensionality, Parallel computing, Reliability theory, Dynamic programming, Environmental systems.