THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE NANTES
COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

« Ran BAO »

« Modélisation formelle de systèmes de drones civils à l'aide de méthodes probabilistes paramétrées»

Thèse présentée et soutenue à « Nantes », le « 07/05/2020 »
Unité de recherche : Informatique
Thèse N° :

Rapporteurs avant soutenance :

Patricia Bouyer-Decitre	Directrice de Recherches CNRS, LSV, ENS Paris-Saclay
Laure Petrucci	Professeure des Universités, LIPN, Université Paris 13

Composition du Jury :

Examinateurs :		
	Olga Kouchnarenko	Professeur des Universités, Femto-st, Université de Franche-Comté
	Claud Jard	Professeur des Universités, LS2N, Université de Nantes
Dir. de thèse :	Christian Attiogbé	Professeur des Universités, LS2N, Université de Nantes
Co-dir. de thèse :	Benoît Delahaye	Maître de Conférences, LS2N, Université de Nantes

Invité(s) :

Philippe BARANGER	Directeur technique chez PIXIEL

REMERCIEMENTS

D'abord je voudrais remercier M. Christian ATTIOGBE (LS2N), M. Benoît DELAHAYE (LS2N) et M. Philippe BARANGER (PIXIEL) de m'avoir donner l'occasion d'effectuer cette thèse intéressante.

Le domaine des Architectures et Logiciels sûrs (précisément de la modélisation et de la vérification) est très vaste et complexe. Merci à M. Christian ATTIOGBE et M. Benoît DELAHAYE de m'avoir beaucoup guidée dans ce domaine.

M. Philippe BARANGER est expert dans le domaine des drones ; grâce à lui, nous avons réussi à comprendre les caractéristiques des drones et à analyser leur fonctionnement et leur défaillance. Merci également à Paulain FOURNIER pour nos discussions et surtout pour avoir mis à ma disposition son outil de *model checking* statistique. Merci aussi à mes camarades de PIXIEL et du LS2N pour les idées et expériences partagées. Je suis très contente de travailler avec eux dans ce génial environnement.

Je voudrais aussi remercier ma famille pour son soutien pendant mes études, pour avoir été à l'écoute de mes nouvelles idées, et pour ses suggestions.

Le français n'est pas ma langue maternelle. Merci beaucoup à mes encadrants M. Christian ATTIOGBE, M. Benoît DELAHAYE et mon mari d'avoir beaucoup corrigé mon français pendant que j'ai écrit cette thèse.

Merci beaucoup à Benoît DELAHAYE, Christian ATTIOGBE, Claud JARD, Laure PETRUCCI, Olga KOUCHNARENKO, Patricia BOUYER-DECITRE et Philippe BARANGER qui ont accepté de participer à la soutenance en visio-conférence pendant le confinement dû au Covid-19.

À la fin merci à PIXIEL [1] et ANRT [2] (pour la convention CIFRE [3]) pour l'aide financière apportée à l'accomplissement de cette thèse.

1. PIXIEL : Le groupe PIXIEL est une entreprise qui fabrique et exploite des drones.
2. ANRT : Association Nationale Recherche Technologie.
3. CIFRE : Conventions Industrielles de Formation par la Recherche

SOMMAIRE

Introduction **7**

1 Contexte de la thèse **11**

2 Les composants, le fonctionnement et les problématiques d'un drone **15**
2.1 L'attitude du drone 15
2.2 Les composants d'un drone 16
2.2.1 Les composants matériels 16
2.2.2 Les composants logiciels 19
2.3 La règlementation en matière de drones 21
2.4 Problématiques du drone 22
2.4.1 Les défaillances matérielles et logicielles 23
2.4.2 Les zones de sécurité 24

3 Problématique de la de thèse et état de l'art sur la modélisation et la vérification des systèmes de drones **25**
3.1 Problématiques et orientations de la thèse 25
3.2 État de l'art sur la modélisation et la vérification des systèmes de drones 27

4 Modélisation formelle et vérification **31**
4.1 Chaînes de Markov 31
4.2 Chaînes de Markov paramétrées (pMC) 32
4.3 Model Checking Statistique 33
4.4 Model Checking Statistique Paramétrique (pSMC) 34
4.5 Les outils de model checking 38
4.5.1 PRISM 39
4.5.2 PARAM 41
4.5.3 MCpMC 42

5 Méthode de modélisation formelle de drones civils **47**
5.1 Un aperçu du modèle de drone final 47
5.2 Méthode de construction du modèle 49
5.3 Modèle du drone à l'aide d'une chaîne de Markov 50
5.4 Première version du modèle (en PRISM) 52
5.5 Calcul de la position par rapport aux zones de sécurité 53
5.6 Deuxième version : prise en compte des zones de sécurité 56
5.7 Expérimentations sur une chaîne de production 59
5.8 Troisième version : prise en compte de la distance exacte 63
5.8.1 Le modèle construit pour PRISM 63
5.8.2 Le modèle construit en Python pour utiliser la méthode pSMC . 64
5.9 Versions 4 à 7 avec la méthode pSMC 71
5.10 Version 8 avec la méthode pSMC : prise en compte de longs trajets . . 75

6 Expérimentations et interprétations **81**
6.1 Implémentation 81
6.1.1 Mise en œuvre de la version 2 avec le model checker PRISM . . 81
6.1.2 Mise en œuvre de la version 3 avec l'outil MCpMC 83
6.1.3 Comparaison des versions 4, 6 et 7 en MCpMC 85
6.1.4 Mise en œuvre de la version finale (version 8) avec MCpMC . . 88
6.2 Réduire la probabilité d'entrer en zone interdite 89
6.2.1 Réduction du nombre de points sur une ligne droite 89
6.2.2 Allongement du temps des trajets 91

7 Conclusion **95**

Bibliographie **101**

8 Annexe **107**
8.1 Correspondance entre versions du modèle et fichiers sources 107
8.2 Polynômes résultant de l'analyse du modèle (version 3) avec MCpMC . 108
8.2.1 Polynôme résultat pour 100k simulations (pour $T_{chemin} = 5$ et $T_{repondre} = 1$) 108
8.2.2 Polynôme résultat pour 100k simulations (pour $T_{chemin} = 7$ et $T_{repondre} = 1$) 112

INTRODUCTION GÉNÉRALE

Cette thèse est une thèse CIFRE [4]. L'entreprise PIXIEL [5] qui la subventionne est une entreprise spécialisée dans le spectacle de drones. Les drones étant de plus en plus utilisés dans le civil, la sécurité de leur environnement devient un sujet très important. Concernant la pratique actuelle, les spectacles avec des drones ne sont donnés que par beau temps et l'espace en dessous de la zone de vol des drones est interdite aux acteurs du spectacle et au public. Cependant, il n'y a aucune preuve ou garantie que les drones suivent toujours les plans de vol prévus. Scott Andersen et George Romanski [1] soulignent l'importance du logiciel et proposent d'adopter les outils de développement pour la sûreté du logiciel. Le système de drone est le cœur du fonctionnement du drone, sa sûreté est donc encore plus importante que celle de logiciels ordinaires. Développer un modèle avec des méthodes formelles pour analyser avant de développer le système est déjà proposé par plusieurs articles [2] [3] [4], mais aujourd'hui les travaux utilisant le *model checking* sont souvent consacrés aux flottes de drones [5]. L'entreprise PIXIEL a décidé de faire cette étude pour rendre ses drones plus fiables pour le public dans les spectacles de drones.

Le but principal de cette thèse est donc de trouver une méthode pour rendre les vols de drones plus sûrs (sans danger) vis à vis de la présence du public. Alors que les pannes physiques sont bien gérées avec les drones *Hexarotors* [6], une panne logicielle peut être beaucoup plus problématique et complexe à étudier. En effet dans ce cas, le comportement du drone peut devenir imprévisible.

Dans cette thèse, nous proposons un modèle formel du système de drone pour analyser les différents problèmes qu'un drone peut rencontrer. Nous cherchons les sources possibles des problèmes et essayons de les corriger. Nous avons ensuite compris que pour PIXIEL la sécurité du public est encore plus importante que la sécurité du drone. Pour cette raison nous étudions plus profondément la sécurité du public. Dès que les drones ne volent pas au dessus du public, le public est en sécurité même

4. CIFRE : Conventions Industrielles de Formation par la Recherche.
5. PIXIEL : Nous donnons une présentation plus générale dans la suite.
6. *Hexarotors* : ils disposent de six moteurs.

si le drone tombe.

L'un des problèmes critiques dans ce contexte est l'éventuelle inexactitude de l'estimation de la position dans les systèmes de drones, ce qui peut provenir soit des mesures imprécises des capteurs, soit d'une interprétation erronée des données provenant de ces capteurs. En dehors des défaillances des composants physiques du drone, il y a aussi un aspect beaucoup plus critique à prendre en compte : les conditions météorologiques. Par conséquent, une approche générale visant à améliorer la sécurité des drones consiste à étudier l'impact des imprécisions des mesures des positions du drone sur sa trajectoire par rapport à un plan de vol fixé, tout en prenant en compte les conditions météorologiques.

Pour cela nous nous basons sur une organisation en zones, où se trouve le public et où évolue le drone ; les zones où il y a plus de public sont plus dangereuses lorsque le drone y pénètre. Nous calculons donc la probabilité pour que le drone entre dans chaque zone, en faisant le lien avec les composants du drone et ses paramètres de vol. Dans ce but, nous avons construit une chaîne de Markov comme modèle formel du drone. Ce modèle a été encodé en PRISM [6, 7] puis en Python (dédié aux outils de *model checking*). Il comporte un état initial, des transitions qui présentent les comportements des composants du drone, et différents états finaux qui correspondent aux différentes zones ; nous pouvons donc obtenir la probabilité qu'a le drone d'arriver dans chaque zone.

Nous avons effectué des tests statistiques sur un composant (le filtre) du drone, et avons construit plusieurs modèles pour étudier si nous pouvons améliorer la précision tout en ralentissant le moins possible le drone. Malheureusement, à cause de l'environnement et de la taille des tests, le résultat des tests statistiques ne semble pas refléter les données obtenues en situation de vol réel. Pour cette raison nous ne pouvons pas utiliser les résultats des tests statistiques pour choisir un filtre qui correspond au mieux aux besoins de PIXIEL. Nous avons alors décidé d'utiliser les capacités des filtres comme paramètres du modèle formel du drone. Un modèle paramétré est alors construit progressivement, puis vérifié à l'aide d'une nouvelle méthode : le *model checking statistique paramétré*.

Cette méthode de vérification est plus libre et rapide que les méthodes standard telles que le *model checking*. En revanche le résultat en sortie n'est plus un résultat

exact mais un résultat estimé. La méthode prend un modèle formel en entrée et donne un polynôme en sortie. Ce polynôme exprime la relation entre les paramètres d'entrée (du modèle) pour produire une probabilité en fonction des entrées. La taille du modèle augmente avec la gamme et le nombre des paramètres. Pour l'analyse des propriétés de ce modèle paramétré, il s'avère que les techniques et outils classiques (PRISM, PARAM) sont limités et ne parviennent pas à encoder le modèle ou achever son analyse. Afin de mettre en œuvre notre méthode de *model checking statistique paramétré*, un nouvel outil prototype (MCpMC) a été développé à l'occasion dans l'équipe, par extension d'un prototype existant. Notre modèle paramétré du drone a alors été vérifié par ce prototype ; ses performances sont plutôt bonnes comme nous allons le présenter dans la thèse.

Notre modèle du drone permet d'expérimenter différents paramètres de vol du drone dans le but de traiter la question initiale de la sécurité du public (en évitant les vols dans les zones réservées au public). Par exemple, en considérant différents trajets de vols de drone dans notre modèle, nous avons finalement trouvé certaines règles pour diminuer la probabilité que le drone vole au dessus du public ; ceci répond aux préoccupations à la base de notre projet de recherche et à l'intérêt de l'entreprise PIXIEL.

CHAPITRE 1

CONTEXTE DE LA THÈSE

Cette thèse est financée par le dispositif de Conventions Industrielles de Formation par la Recherche (CIFRE) qui a subventionné l'entreprise PIXIEL pour une collaboration de recherche avec le Laboratoire des Sciences du Numérique de Nantes (LS2N).

Présentation générale de PIXIEL

Depuis sa création en 2011, la société PIXIEL a fait le choix de positionner l'innovation au cœur de son activité. La société a toujours conçu et créé ses propres drones, hier pour la prestation audiovisuelle, aujourd'hui pour l'industrie dans le domaine du spectacle (ATMOS), pour la sécurité avec des drones automatiques (NeoSafe) et pour la formation en pilotage de drones (EMD).

PIXIEL maîtrise l'ensemble des étapes de conception et de développement de ses solutions de drones autonomes. En l'absence de normes industrielles pour les drones, la société respecte au plus près celles de l'aéronautique pour produire des dispositifs sûrs et fiables. Nous nous sommes donc inspirés des règles de l'avionique pour notre modélisation.

Pour relever les défis technologiques et concevoir ces véritables robots volants, PIXIEL maîtrise à la fois les éléments physiques constitutifs du drone (châssis, cartes électroniques, ...) mais également l'ensemble des logiciels. Afin de conserver sa capacité d'innovation, PIXIEL a décidé de développer toutes ses briques technologiques en interne.

Pour résumer, l'entreprise a beaucoup d'expérience sur tous les aspects de la vie du drone, que ce soit de la fabrication à leur usage professionnel, sans oublier les formations pour les futurs pilotes.

PIXIEL a regroupé ses nombreuses activités en 3 parties : *ATMOS* pour les spectacles, *EMD* pour la formation et *NeoSafe* pour la sécurité automatisée. ATMOS a été le cadre de nos travaux de recherche.

ATMOS - Spectacle de drones

FIGURE 1.1 – Spectacle au Puy du Fou en partenariat avec PIXIEL.

Dans la majorité des spectacles traditionnels, l'espace que représente le ciel n'est que très peu exploité. Intégrer des drones permet de remplir cet espace et de proposer un spectacle en 3 dimensions.

Les spectacles organisés avec le Puy du Fou [1] ne sont pas les seuls opérés par l'entreprise ; PIXIEL est en effet connu dans le monde entier pour ses spectacles de drones.

Lors des spectacles nocturnes, les drones doivent voler devant du public, ce qui rend critique le sujet de la sécurité. Par exemple en cas de panne, le dysfonctionnement du drone peut entraîner sa chute. C'est principalement pour cette raison que PIXIEL a décidé de proposer ce sujet de thèse.

Le but est d'améliorer la fiabilité (minimiser l'impact des défaillances du drone sur le public) de leur système à l'aide des techniques informatiques et notamment les méthodes formelles (modélisation et vérification).

Organisation pratique pour la thèse

Pour ma thèse, j'ai rejoint l'équipe qui s'occupe du système de drone en R&D (Recherche et Développement) à Nantes. Cette équipe s'occupe à la fois de la conception mais aussi de la fabrication du système. J'ai donc eu l'opportunité de collaborer avec l'équipe qui a réalisé le système, ce qui inclut les techniciens et les électroniciens.

1. Puy du Fou : Un parc d'attraction en Vendée, France.

Le laboratoire LS2N est impliqué dans ce projet à travers un partenariat de recherche avec l'équipe AeLoS [2]. La thèse s'est déroulée pendant les 3 années en étroite collaboration avec mon équipe de recherche et l'équipe R&D de PIXIEL. Une partie de la semaine est passée dans l'entreprise et l'autre au laboratoire.

La thèse est organisée de la manière suivante. Dans le chapitre 2, nous donnons les bases essentielles pour comprendre le fonctionnement et le comportement des drones, puis les règlementations et les problématiques du drone. Le chapitre 3 présente l'orientation de cette thèse et les travaux existants. Le chapitre 4 est une introduction aux chaînes de Markov, chaînes de Markov paramétrées, *model checking statistique*, *model checking statistique paramétrique* et aux outils ou prototypes de vérification. La construction progressive du modèle formel est présentée dans le chapitre 5, puis nous présentons les implémentations et les résultats dans le chapitre 6. Enfin le chapitre 7 présente les conclusions et les perspectives de cette thèse.

2. AeLoS : Architectures et Logiciels Sûrs.

CHAPITRE 2

LES COMPOSANTS, LE FONCTIONNEMENT ET LES PROBLÉMATIQUES D'UN DRONE

Dans ce chapitre, nous présentons les connaissances fondamentales sur les drones et qui sont nécessaires à la compréhension de la suite du document. Dans un premier temps, nous présentons les composants matériels d'un drone. Ensuite, nous expliquons comment le logiciel de contrôle d'un drone peut être décomposé. Nous allons aussi nous focaliser sur les composants logiciels les plus importants. Nous présentons ensuite les réglementations en matière de drone, notamment en Europe mais aussi dans le monde. Finalement nous introduisons une liste de problématiques relatives au fonctionnement de drones.

2.1 L'attitude du drone

Le mouvement d'un drone dans l'espace est dicté par la position relative (orientation) de ses différents moteurs. Ainsi, pour commander efficacement le drone, il est important de connaître avec précision l'angle de ses moteurs dans un repère relatif. Ces mesures sont appelées l'*attitude* du drone.

Notons la différence entre position et *attitude* : alors que la position du drone est définie par les coordonnées tridimensionnelles x, y et z, son *attitude* est définie par les coordonnées géographiques ; ce sont les mesures de lacet, tangage et

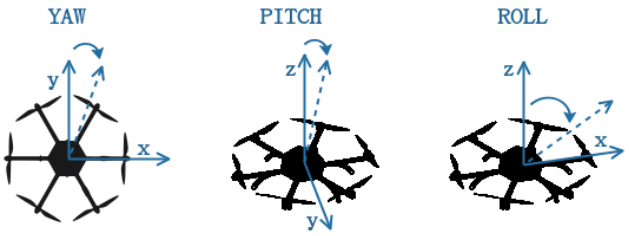


FIGURE 2.1 – Les coordonnée de l'*attitude*.

roulis (en anglais yaw, $pitch$ et $roll$) par rapport à l'axe vertical (voir la Figure 2.1). L'orientation permet de contrôler le mouvement du drone ; en contrôlant la vitesse de chaque moteur, il est possible de contrôler quel moteur sera le plus rapide, et donc de contrôler la direction dans laquelle le drone volera.

2.2 Les composants d'un drone

FIGURE 2.2 – Composants d'un drone.

Afin de modéliser fidèlement le système de drone, il faut tout d'abord étudier le drone lui-même. La Figure 2.2 montre un drone typique civil. Ce drone contient 4 moteurs et des composants basiques que nous détaillons dans la suite. Pour ses spectacles, l'entreprise PIXIEL utilise des drones plus avancés équipés de 6 moteurs et davantage de capteurs.

Nous décomposons maintenant un drone en composants matériels et composants logiciels. Dans la Figure 2.3, les composants en bleu clair sont des matériels que nous présentons dans la section 2.2.1, les composants en bleu foncé font certains calculs basiques et échangent des valeurs avec le Contrôleur de Vol. Nous présentons ces composants dans la section 2.2.2.

2.2.1 Les composants matériels

Nous présentons ici les composants matériels du drone, avec un focus sur les capteurs de position. Certains drones sont aussi équipés de matériels qui ne sont pas présentés ici, comme le capteur de courant électrique, l'équipement de stabilisation de carte, etc. Ces composants apportent plus de possibilités au drone mais entraînent aussi plus de consommation d'énergie et plus de poids pour le drone. Les bons matériels conditionnent souvent un vol plus long et permettent de porter des objets plus lourds.

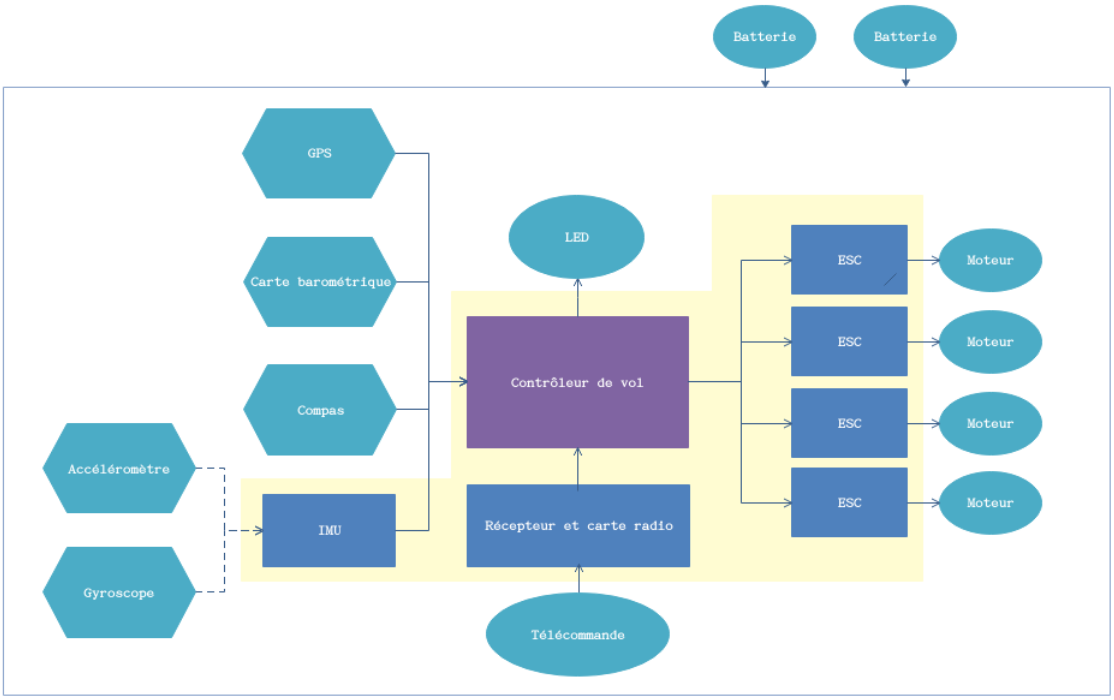


FIGURE 2.3 – Assemblage des composants d'un drone.

Le moteur. Le moteur transforme une énergie électrique en énergie mécanique. Les drones *Quadrotor* disposent de quatre moteurs ; les *Hexarotors* disposent de six moteurs. En fonction du moteur, le drone est plus puissant, il peut voler plus haut et plus rapidement ; et dans certains types de perturbations il est plus stable. Les drones *Hexarotors* peuvent voler avec quatre moteurs lors de défaillances matérielles ; Ils sont plus puissants et peuvent porter les objets lourds en volant.

La batterie. La batterie est la source d'énergie du drone. Les drones sont toujours équipés d'au moins une batterie. Dans les situations non désirées, par exemple quand le drone tombe, c'est souvent la batterie qui endommage le drone et le rend irrécupérable.

La télécommande. La télécommande est destinée au pilote, mais les drones automatiques peuvent aussi être équipés d'une ou plusieurs télécommandes afin qu'un humain puisse prendre la main dans les situations d'urgence.

Les LED. Les lumières sur un drone sont à base de LED. Tous les drones ne sont pas équipés en LED mais c'est un composant pratique pour trouver la position du drone visuellement dans la nuit.

La carte barométrique. La carte barométrique est aussi appelée altimètre. Elle agit comme un capteur pour le drone. Son rôle est de mesurer l'altitude à l'aide de la pression atmosphérique. Ce composant est particulièrement critique car il génère beaucoup de bruit, qui peut perturber les mesures de position.

Le compas. Le compas est un capteur du drone qui donne une référence de direction (le nord) sur le plan horizontal et permet ainsi la mesure d'angles horizontaux par rapport à cette direction.

Le GPS. Le GPS (*Global Positioning System*) est un capteur du drone qui consomme plus d'énergie que les autres. Il utilise un système de positionnement par satellites appartenant au gouvernement des États-Unis. Les signaux transmis par les satellites peuvent être librement reçus et exploités par quiconque. Il est un peu plus lent que les autres capteurs, mais il est plus stable et a un bon fonctionnement sur une grande trajectoire extérieure.

L'accéléromètre. L'accéléromètre est un capteur qui permet de mesurer l'accélération linéaire du drone. Il s'agit en fait de 3 accéléromètres qui calculent les accélérations linéaires selon 3 axes orthogonaux. L'accéléromètre est sensible aux perturbations, mais il est rapide, précis, et en l'absence de perturbations, ses performances sont constantes dans le temps.

Le gyroscope. Le gyroscope est un capteur qui mesure la position angulaire et l'orientation par rapport à un inférentiel inertiel. Il consomme moins d'énergie que les autres capteurs, il est rapide et précis, mais il souffre des erreurs d'intégration, c'est-à-dire qu'il a un bruit qui augmente avec le temps. Il est complémentaire à l'accéléromètre.

2.2.2 Les composants logiciels

La zone jaune de la Figure 2.3 représente le système de drone ; le cœur du système de drone est celle qui est en violet, c'est le contrôleur de vol. C'est le composant qui nous intéresse le plus car c'est celui qui a potentiellement le plus d'impact sur la trajectoire du drone. Nous introduisons ici les composants logiciels du drone ; ils sont pour certains fortement intégrés aux matériels (ESC, Radio, IMU ...).

Le contrôleur de moteur (*Electronic Speed Controller* - ESC). Chaque moteur du drone est couplé avec un contrôleur de moteur. Ces contrôleurs permettent de réguler la vitesse des moteurs et ainsi de contrôler l'attitude du drone.

Le récepteur radio. Le récepteur radio est un appareil électronique destiné à capter, sélectionner et décoder les ondes radioélectriques émises par les télécommandes.

FIGURE 2.4 – Capteur de mesure inertielle (IMU).

Le capteur de mesure inertielle (*Inertial measurement unit* - IMU). Le capteur de mesure inertielle (IMU) est un composant qui intègre la plupart du temps un accéléromètre et un gyroscope. C'est un instrument intégrant du logiciel, utilisé en navigation, capable d'intégrer les mouvements d'un mobile (accélération et vitesse angulaire) pour estimer son orientation (angles de roulis, de tangage et de lacet), sa vitesse linéaire et sa position. L'estimation de la position est relative au point de départ ou au dernier point de recalage. Les IMUs des drones sont souvent très petits ; la Figure 2.4 en montre un exemple.

Certains drones sont équipés de capteurs supplémentaires, comme nous l'avons présenté. Plus le contrôleur de vol a de capteurs et plus l'estimation de sa position sera précise. Nous considérons cependant que l'IMU est un des composants les plus importants dans l'estimation de la position du drone dans l'espace.

Le contrôleur de vol. Nous introduisons maintenant le plus important parmi les composants logiciels : le contrôleur de vol. Le contrôleur de vol est responsable de la

collecte des données venant des différents capteurs. Ces données permettent de calculer *la position* et *l'attitude* précises du drone ainsi que d'ajuster l'*attitude* de manière à suivre le plan de vol indiqué du mieux possible.

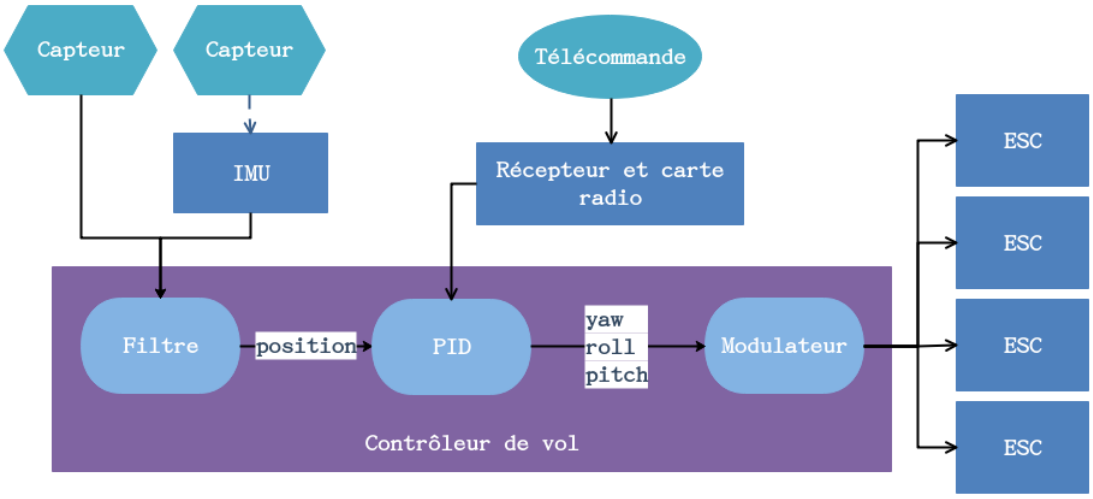


FIGURE 2.5 – Architecture d'un système de drone.

Nous allons modéliser le contrôleur de vol général représenté dans la Figure 2.5. Les composants les plus importants du contrôleur de vol sont : le PID (*Proportional Integral Derivative*), le modulateur et le filtre.

— **Le PID.** Le PID (*Proportional-Integral-Derivative*) est le composant qui calcule l'attitude dont le drone a besoin pour atteindre le prochain point. Il existe une diversité de PID [8] ; ils sont tous très bien documentés.

— **Le modulateur.** Le modulateur est le traducteur entre le PID et le ESC. La modulation peut être définie comme le processus par lequel le signal est transformé de sa forme originale en une forme adaptée au canal de transmission.

— **Le filtre.** Les signaux issus des capteurs contiennent du bruit. Les filtres permettent de les atténuer voire de les supprimer. Cependant, chaque type de filtre a ses avantages et convient plus ou moins bien à la situation d'utilisation.
Aujourd'hui, le filtre le plus utilisé est le filtre EKF (*Extended Kalman Filter*) [9] car il est rapide et offre une précision correcte. C'est pour cela que nous le prenons comme référence pour la comparaison avec les autres filtres.
D'autres filtres comme le *Gradient Descent* [10], le UKF (*Unscented Kalman Filter*) [11] et le CKF (*Explicit Complement Filter*) [12] permettent d'obtenir de

meilleurs résultats tout en sacrifiant la précision ou la rapidité. Certains articles comparent les différents filtres [13, 14].

Tous les filtres proposent des fréquences et des précisions différentes. La fréquence et la précision du filtre doivent donc être incluses dans notre modèle. Cela permettra de voir si un filtre précis mais à basse fréquence est plus avantageux pour la sécurité du drone qu'un filtre moins précis mais plus rapide.

2.3 La règlementation en matière de drones

L'utilisation des drones dans le domaine civil est une activité encore récente. Par conséquent il n'existe que très peu de règlementations encadrant leur utilisation. Selon les pays, on peut tout de même trouver des lois locales en attendant la mise en place de lois internationales spécifiques. Une réglementation européenne a été publiée le mardi 11 juin 2019 et mise en application le 1 juillet 2019 [1].

Utilisation en France

Pour l'usage d'un drone de loisir, le Ministère de la Transition écologique et solidaire a mis en ligne un document listant les "*10 commandements*" suivants, à respecter pour l'usage des drones. Ils n'ont pas changé après la publication du règlement européen.

1 Je ne survole pas les personnes.
2 Je respecte les hauteurs maximales de vol.
3 Je ne perds jamais mon drone de vue et je ne l'utilise pas la nuit.
4 Je n'utilise pas mon drone au-dessus de l'espace public en agglomération.
5 Je n'utilise pas mon drone à proximité des aérodromes.
6 Je ne survole pas de sites sensibles ou protégés.
7 Je respecte la vie privée des autres.
8 Je ne diffuse pas mes prises de vues sans l'accord des personnes concernées et je n'en fais pas une utilisation commerciale.
9 Je vérifie dans quelles conditions je suis assuré pour la pratique de cette activité.

1. Règlement : https ://www.ecologique-solidaire.gouv.fr/drones-usages-professionnels

10 En cas de doute, je me renseigne...

Il existe aussi certaines lois plus spécifiques, mais aucune n'est assez précise et n'atteint le niveau de la certification DO-178C [15] (*Software considerations in airborne systems and equipment certification*), qui fixe les conditions de sécurité applicables aux logiciels critiques de l'avionique pour l'aviation commerciale et générale. Sont précisées notamment les contraintes de développement liées à l'obtention de la certification pour un logiciel d'avionique.

Malgré l'inexistence d'une telle certification internationale pour les logiciels de drones, les entreprises essayent d'améliorer la sécurité de leurs systèmes au maximum. Il est donc important d'étudier les défaillances qui peuvent apparaître lors de l'utilisation d'un drone, afin de réduire leurs impacts si elles se produisent.

2.4 Problématiques du drone

Afin d'améliorer le système de drone, la première étape est d'identifier les points problématiques et critiques. Les défaillances les plus fréquentes sont matérielles et se traduisent généralement par une perte du drone plus ou moins violente (chute, incendie, ...). Au niveau des logiciels, la présence d'un contrôleur de vol est très recommandée.

Nous étudions les défaillances des drones et notons qu'il existe deux types de situations. Soit le drone peut s'arrêter de voler et tomber dans une zone sécurisée, soit il entre dans une zone "interdite" où il met le public en danger. Certains drones professionnels gèrent le risque de chute à l'aide de la redondance des composants matériels.

La méthode *Failure Mode Effects and Criticality Analysis (FMECA)* est un processus pour déterminer les conséquences que des défaillances peuvent avoir sur le fonctionnement d'un système complexe aéronautique [16]. JiaLin Wang et Wei Chen ont utilisé cette méthode pour analyser les défaillances possibles dans ce contexte. Dans leur analyse, nous pouvons voir, en dehors des problèmes de défaillance de composants physiques d'avion, que les problèmes de vol hors de la trajectoire prévue sont parmi les plus critiques. C'est aussi vrai pour les drones. Cette analyse nous a permis

de mettre en évidence les problématiques des drones et nous a guidé pour le choix des propriétés à vérifier par la suite.Dans la suite, nous analysons les différents types de défaillances.

2.4.1 Les défaillances matérielles et logicielles

Les composants matériels du drone peuvent être défectueux et entraîner un dysfonctionnement ou une défaillance du drone. De plus comme l'altitude des drones est beaucoup plus basse que celle d'un avion, les drones volent à proximité du public. Concernant les drones, les pannes physiques sont mieux gérées avec les drones *Hexarotors* grâce à leur nombre de moteurs. Les pannes logicielles, elles, peuvent être beaucoup plus problématiques et complexes à étudier.

Nous détaillons ci-après quelques sources de défaillances en liaison avec le logiciel.

La perte de signal. Un des problèmes les plus critiques est celui qualifié du "*drone perdu*" : lorsque les drones présentent un problème d'*attitude*, le sens ou la vitesse de vol ne sont pas ceux commandés par le pilote ou le programme en mode automatique. En conséquence, une fois que le drone est trop éloigné de sa télécommande et de la station de sol, tout signal envoyé sera perdu ; le drone risque alors d'être perdu de vue et devient alors un "*drone perdu*".

FIGURE 2.6 – Défaillances du drone.

La perte de trajectoire. Pendant les spectacles utilisant des drones, par exemple celui montré dans la Figure 2.7, nous nous intéressons particulièrement à la distance entre le public, les acteurs et les drones. Une trajectoire de vol est alors définie de façon que le vol ne

FIGURE 2.7 – Spectacle avec des drones.

présente aucun danger pour le public. Pour savoir comment le drone peut voler sans danger pour le public et les acteurs, il faut pouvoir détecter s'il perd sa trajectoire. Différentes situations peuvent en être la source, par exemple le vent, la perte de signal, etc.

2.4.2 Les zones de sécurité

Afin de limiter les impacts des défaillances possibles, la notion de zone de sécurité a été définie. Dans le cadre aéronautique, par exemple, la norme DO-178C (*Software considerations in airborne systems and equipment certification*) recommande différentes zones de sécurité.

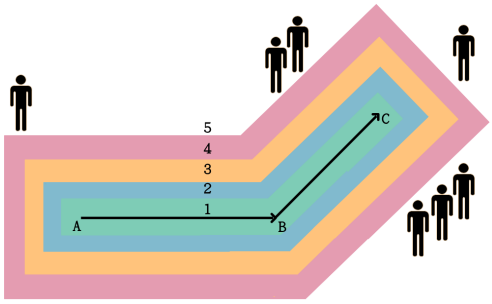


FIGURE 2.8 – Zones de sécurité.

La taille de chaque zone de sécurité n'est pas définitivement fixée ; elle peut être définie pour un besoin spécifique ou pour une application donnée. En pratique, les zones de sécurité sont spécifiquement définies pour un environnement de vol et pour un plan de vol donné. Autour d'un drone, les zones où peuvent se trouver du public sont catégorisées. Le principe général de cette définition est que personne ne doit se trouver dans les zones 1 à 3, alors qu'il peut y avoir quelques personnes qui surveillent les drones (ou autres matériels du spectacle) dans la zone 4 et que la plus grande partie des personnes se trouvent dans la zone 5. Les zones 1 à 5 correspondent donc aux niveaux E à A de la norme DO-178C.

Le but de cette thèse est d'étudier des méthodes pour améliorer la sécurité des drones. Nous allons nous concentrer sur le calcul de la probabilité pour qu'un drone reste dans les bonnes zones de sécurité. Pour cela, nous allons développer un modèle formel du drone et de sa trajectoire et utiliser des méthodes statistiques paramétrées pour l'analyser.

CHAPITRE 3

PROBLÉMATIQUE DE LA DE THÈSE ET ÉTAT DE L'ART SUR LA MODÉLISATION ET LA VÉRIFICATION DES SYSTÈMES DE DRONES

Dans ce chapitre, nous listons d'abord les problématiques que nous voulons résoudre, puis nous présentons les travaux existants. En effet, nous avons besoin de comprendre les problématiques dans le domaine des drones et les travaux existants pour savoir sur quels aspects nous pouvons apporter des contributions et trouver les solutions pour le faire.

3.1 Problématiques et orientations de la thèse

Nous précisons le périmètre de modélisation et vérification du bon fonctionnement du drone. Nous allons nous concentrer sur la problématique de la perte de trajectoire.

La perte de signal. Dans la situation de perte de signal (drone perdu), le comportement du drone devient imprévisible. Il existe très peu de méthodes pour résoudre ce problème. Pour assurer la sécurité, la solution actuelle de PIXIEL est de couper l'énergie des moteurs lorsque le drone s'éloigne de la trajectoire pour aller vers le public. Une fois que les moteurs sont coupés, le drone tombe. Il est risqué de réutiliser les pièces après ces incidents. L'évitement de la situation est donc une meilleure solution.

La perte de trajectoire. Pour étudier le problème de la perte de trajectoire, nous avons deux solutions. La première solution est de calculer la probabilité de perdre le

contrôle du drone et chercher à la réduire au maximum. La seconde solution serait de décider d'une distance minimale entre le drone et le public et calculer la probabilité qu'il dépasse cette distance. Ensuite nous pourrons trouver un moyen de réduire la probabilité pour que la limite de sécurité soit franchie.

La première solution n'est pas réalisable car nous ne pouvons pas contrôler le vent qui peut engendrer des pertes de contrôle. De même si le problème vient du drone lui-même, il est aussi impossible de contrôler toutes les interférences (par exemple avec les réseaux téléphoniques). Nous avons donc choisi d'orienter nos recherches vers la seconde solution.

L'évitement des zones de sécurité. La probabilité que le drone mette en danger des humains est directement proportionnelle à la probabilité que le drone entre dans les zones de sécurité 4 ou 5. Notre objectif sera donc de calculer cette probabilité, et ensuite étudier comment la minimiser sur la base d'un modèle.

La configuration optimale des drones. L'entreprise PIXIEL utilise les drones dans le domaine du spectacle. Cela signifie que le trajet des drones est déterminé à l'avance. L'utilisation d'un filtre plus précis mais plus lent parait donc raisonnable puisque nous pouvons effectuer à l'avance certains calculs avant de commencer le vol.

Jusqu'à présent, les drones développés par PIXIEL sont équipés d'un filtre EKF (rapide mais peu précis) mais nous conjecturons qu'il existe une meilleure combinaison pour notre cas, par exemple l'emploi d'un filtre UKF (puis lent mais plus précis). En effet, le problème de lenteur peut être compensé par les calculs effectués à l'avance.

Notre problématique est de faire diminuer la probabilité pour que le drone prenne la mauvaise direction. Nous devons donc vérifier si le filtre peut faire varier cette probabilité. Pour cette raison, nous allons nous appuyer sur un modèle abstrait formel du drone. Il doit être le plus proche possible du fonctionnement du drone. Dans la suite nous exposons notre démarche de modélisation pour atteindre ces objectifs.

L'impact du vent. La plupart des travaux étudient le fonctionnement du drone dans les conditions idéales ; or le vent par exemple est un facteur qui influence grandement la trajectoire d'un drone. Nous allons donc, en plus des paramètre liés aux capteurs et au filtre, prendre en compte le vent comme un paramètre de notre modélisation du drone.

3.2 État de l'art sur la modélisation et la vérification des systèmes de drones

Aujourd'hui les drones sont de plus en plus utilisés autant pour des usages militaires, civils professionnels que pour des usages personnels où leur pilotage devient un vrai loisir. Le prix de ces appareils volants est relativement accessible pour des petits modèles. De plus, les applications qu'on peut leur prêter n'ont de limite que notre imagination. Il est donc important d'en assurer un bon fonctionnement à priori. C'est pourquoi beaucoup de chercheurs ont consacré des travaux de recherche autour de l'univers des drones.

Nous organisons les travaux existants en trois catégories. Dans la première catégorie, nous considérons les travaux relatifs au fonctionnement général des drones.

Les recherches de Koppány Máthé et Lucian Buşoniu [17] constituent une très bonne base car ils donnent une vision générale des drones et de leur fonctionnement. Dans ces travaux, les auteurs abordent l'inspection ferroviaire en utilisant la méthode de vision et contrôle. La section consacrée au *Flight Control and Planning* est la plus intéressante pour nous. Ils présentent en particulier certains composants du contrôleur de vol (en particulier le PID), le comportement du *Quadrotor*. Ils présentent aussi la technique MPC (*Model Predictive Control*), une technique de commande avancée de l'automatique. Cette technique utilise un modèle dynamique pour prévoir le futur comportement, puis détecte et corrige les problèmes éventuels.

Certains des travaux sont consacrés aux drones équipés de reconnaissance de mouvements [18]. Ils donnent une possibilité d'utiliser des gestes pour piloter le drone. D'autres travaux sont consacrés à la vidéo ; par exemple l'atterrissage automatique sur cible [19] ; cet article a beaucoup intéressé PIXIEL. Avec une grande taille de QR code, le drone peut atterrir automatiquement sur son QR code. L'article [20], consacré à la vidéo, traite la surveillance et la maintenance.

Andres Hernandez, Harold Murcia, Cosmin Copot et Robin De Keyser ont proposé une méthode pour qu'un drone autonome surveille le volume de grains dans un chariot [21]. Ils ont introduit le comportement du Quadrotor, l'IMU (*Inertial Measurement Unit*) et le MPC. Sans l'aide de caméras, Ben G. Fitzpatrick utilise les modifications idempotentes stochastiques, l'analyse bayésienne et les MDPs (*Markov Decision Processes*) pour contrôler le drone sans pilote [22]. Il existe beaucoup d'articles sur ce

sujet [23–26], qui proposent aussi des méthodes de planification de parcours. Dans notre cas, la trajectoire est définie avant le vol donc nous ne nous focalisons pas sur ces derniers travaux.

Dans la deuxième catégorie, nous rassemblons des sujets plus pragmatiques qui vont chercher à détecter les pannes et les dysfonctionnements dont les drones peuvent être victimes.

La méthode FMECA (*Failure Mode Effects and Criticality Analysis*) détecte toutes les défaillances et les trie en fonction de la gravité de leur niveau prédéfini [16]. Dans [27], les auteurs présentent l'élaboration d'un modèle qui sert de base de diagnostic pour la résolution de problèmes du système global. Il existe aussi des travaux sur la détection des erreurs dans un cadre multi-drones [28]. Les méthodes de détection et d'isolation de fautes (FDI *Fault Detection and Isolation*) des capteurs ont attiré notre attention ; mais puisque nous visons des drones sans caméra, nous ne pouvons pas utiliser cette méthode dans notre travail. Elle nous a cependant apporté des informations sur les bruits des capteurs. Nous avons aussi exploré avec intérêt d'autres travaux sur les FDI comme [29–31], qui nous ont permis d'appréhender les différentes catégories de problème.

Nos travaux se rapprochent plus de cette seconde catégorie de sujets. Nous recherchons en effet une méthode fiable pour analyser le système de drone, détecter les principaux paramètres de son fonctionnement ou de sa défaillance, afin de pouvoir ajuster ces paramètres pour réduire les probabilités de pannes.

Les travaux de la troisième catégorie sont des travaux plus spécifiques. Nous avons découvert et approfondi le fonctionnement des drones avec les travaux de la première catégorie ; nous avons mieux cerné la problématique de recherche avec les travaux de la deuxième catégorie ; enfin les travaux de la troisième catégorie vont nous aider à construire notre modèle formel.

Dans le domaine des systèmes embarqués [32], il est possible d'utiliser différentes méthodes afin de construire un modèle du système. HUANG Zhi-Qiu, XU Bing-Feng, KAN Shuang-Long, HU Jun et CHEN Zhe [2] font l'analyse des méthodes et des outils dans le domaine de l'avionique. Stanislav Emel'yanov, Dmitry Makarov, Aleksandr I. Panov et Konstantin Yakovlev proposent dans [33] l'analyse de l'architecture en multi-couches du contrôleur de vol.

Il existe aussi d'autres méthodes pour aboutir à la réalisation du modèle du système : l'utilisation des chaînes de Markov pour le calcul des trajectoires [34] mais aussi la classification et le diagnostic [4]. D'après nos recherches, il n'y a cependant pas d'articles ou travaux de modélisation des systèmes de drone à l'aide de méthodes probabilistes paramétrées.

Dans cette thèse, nous mettons l'accent sur un des composants du contrôleur de vol : le filtre. Celui-ci doit nettoyer les bruits des capteurs. Les travaux autour de la FDI nous montrent que ce bruit existe toujours. Nous avons donc étudié les travaux sur les filtres [9–13], pour voir si nous pouvions en construire un modèle formel, mais il s'avère que tous les filtres n'ont pas les mêmes bases, par conséquent nous ne pouvons pas construire un modèle permettant de représenter toutes les sortes de filtres. Nous avons alors choisi d'utiliser les caractéristiques des filtres (fréquence, précision) en tant que paramètres (variables au constants) de notre modèle.

CHAPITRE 4

MODÉLISATION FORMELLE ET VÉRIFICATION

Dans ce chapitre, nous introduisons les notions et notations nécessaires afin de développer un modèle probabiliste paramétrique pour représenter le comportement de notre drone en fonction d'un plan de vol donné.

Nous commençons par introduire les chaînes de Markov en tant que modèle probabiliste et nous introduisons ensuite le formalisme de modélisation que nous avons utilisé : les chaînes de Markov paramétrées (*parametric Markov Chains - pMCs*). Ensuite nous présentons les bases d'une technique classique de vérification qui est le *model checking statistique* (SMC), et finalement nous montrons comment SMC peut être adapté pour les chaînes de Markov paramétrées.

4.1 Chaînes de Markov

Une chaîne de Markov (MC) est un modèle purement probabiliste $\mathcal{M} = (S, s_0, P)$, où S est un ensemble d'états, $s_0 \in S$ est l'état initial, et $P : S \times S \to [0, 1]$ est la fonction de transition probabiliste qui, étant donné un couple d'états (s_1, s_2), donne la probabilité d'aller de s_1 à s_2.

Étant donnée une chaîne de Markov $\mathcal{M}$, on peut définir une mesure de probabilité sur les exécutions infinies de $\mathcal{M}$ en utilisant une construction standard basée sur une σ-algèbre de cylindres [35].

Une exécution d'une MC est une séquence d'états $s_0 s_1 \dots$ telle que pour tout i, $P(s_i, s_{i+1}) > 0$. Étant donnée une exécution finie $\rho = s_0 s_1 \dots s_l$, sa longueur notée $|\rho|$ représente le nombre de transitions qu'elle contient (en incluant les répétitions). Ici $|\rho| = l$. Nous notons $\Gamma_{\mathcal{M}}(l)$ (ou simplement $\Gamma(l)$ lorsque $\mathcal{M}$ est sans ambigüité dans le contexte) l'ensemble de toutes les exécutions de longueur l de $\mathcal{M}$, et $\Gamma_{\mathcal{M}}$ pour

toutes ses exécutions finies, $i.e.\ \Gamma_{\mathcal{M}} = \cup_{l\in\mathbb{N}}\Gamma_{\mathcal{M}}(l)$. De façon habituelle, nous définissons la mesure de probabilité, notée $\mathbb{P}_{\mathcal{M}}$, sur les exécutions à l'aide d'une σ-algèbre de cylindres (voir e.g. [35]). Cela donne, pour toute exécution finie $\rho = s_0 s_1 \dots s_l$, $\mathbb{P}_{\mathcal{M}}(\rho) = \prod_{i=1}^{l} P(s_{i-1}, s_i)$. Dans le reste du chapitre, nous considérons uniquement des exécutions $finies$. Étant donnée une fonction de récompense $r : \Gamma(l) \to \mathbb{R}$, nous notons $\mathbb{E}^l_{\mathcal{M}}(r)$ pour l'espérance de r sur les exécutions de longueur l d'une MC donnée $\mathcal{M}$.

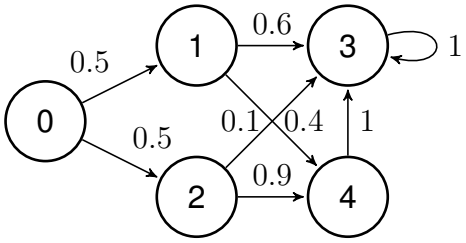


FIGURE 4.1 – Exemple de Chaîne de Markov

Dans la chaîne de Markov $\mathcal{M} = (S, s_0, P)$ de la figure 4.1, nous avons un ensemble d'état $S = \{0, 1, 2, 3, 4\}$, un état initial $s_0 = 0$, la fonction de transition probabiliste P telle que $P_{(s_0,s_1)} = 0.5, P_{(s_0,s_2)} = 0.5, P_{(s_1,s_3)} = 0.6$, etc, ... Pour une exécution finie $\rho = s_0 s_1, s_1 s_3$ de longueur $l = 2$, la probabilité $\mathbb{P}_{\mathcal{M}}(\rho) = P_{(s_0,s_1)} * P_{(s_1,s_3)} = 0.5 * 0.6 = 0.3$. l'ensemble de toutes les exécutions de longueur $l = 2$: $\Gamma_{\mathcal{M}}(2) = \{\rho_1, \rho_2, \rho_3, \rho_4\}$ où $\rho_1 = s_0 s_1, s_1 s_3$, $\rho_2 = s_0 s_1, s_1 s_4$, $\rho_3 = s_0 s_2, s_2 s_3$, $\rho_4 = s_0 s_2, s_2 s_4$. L'espérance r d'arriver à l'état 3 avec une longueur $l = 2$, est $\mathbb{E}^2_{\mathcal{M}}(3) = \mathbb{P}_{\mathcal{M}}(\rho_1) + \mathbb{P}_{\mathcal{M}}(\rho_3)$.

4.2 Chaînes de Markov paramétrées (pMC)

Les chaînes de Markov ne sont pas appropriées à l'analyse de plans de vol de drones car les modèles développés dans ce contexte sont sujets à des données incertaines que nous traitons à l'aide de paramètres, telles que la précision des estimations de position et d'orientation et aussi la force du vent. Les modèles résultants ne sont donc pas purement probabilistes puisqu'ils sont de nature paramétrée. En conséquence nous avons besoin de modèles plus expressifs qui prennent en compte les paramètres de probabilité, tels que les chaînes de Markov paramétrées (*Parametric Markov Chains* - pMC) (voir par exemple [36]).

Une pMC est un n-uplet $\mathcal{M} = (S, s_0, P, \mathbb{X})$ où S est un ensemble fini d'états, $s_0 \in S$ est l'état initial, $\mathbb{X}$ est un ensemble fini de paramètres, et $P : S \times S \to Poly(\mathbb{X})$ est une fonction de transition probabiliste et paramétrique, exprimée comme un polynôme sur les variables contenues dans $\mathbb{X}$. Une valuation de paramètres est une fonction $v : \mathbb{X} \to [0, 1]$ qui affecte des valeurs aux paramètres. Une valuation v est valide par

rapport à une pMC $\mathcal{M}$ lorsque, en y remplaçant les paramètres par leur valeur, on obtient une chaîne de Markov (i.e. dans laquelle la somme des probabilités en sortie de tout état est égale à 1). Lorsque v est une valuation de paramètre valide par rapport à $\mathcal{M}$, on note $\mathcal{M}^v$ la chaîne de Markov résultante.

Étant donnée une pMC $\mathcal{M}$, une exécution ρ de $\mathcal{M}$ est une séquence d'états $s_0 s_1 \ldots$ telle que pour tout $i \geq 0$, $P(s_i, s_{i+1}) \neq 0$ ($i.e.$ la probabilité est soit une constante réelle strictement positive, soit une fonction des paramètres). De la même façon que pour les MCs, nous notons $\Gamma_{\mathcal{M}}(l)$ l'ensemble de toutes les exécutions de longueur l et $\Gamma_{\mathcal{M}}$ l'ensemble de toutes les exécutions finies. Notons que pour toute valuation valide v, $\Gamma_{\mathcal{M}^v}(l) \subseteq \Gamma_{\mathcal{M}}(l)$ puisque la valuation de certaines transitions peut être nulle (transition avec la probabilité 0). La probabilité d'une exécution $\rho = s_0 \ldots s_l$ est définie de manière similaire à celle d'une MC : $P_a^{\mathcal{M}}(\rho) = \prod_{i=1}^{l} P(s_{i-1}, s_i)$. Cette probabilité $P_a^{\mathcal{M}}(\rho)$ est une fonction (polynomiale) des paramètres. Étant donnée une valuation $v : \mathbb{X} \to [0, 1]$ et une fonction f des paramètres, nous notons $f(v)$ l'évaluation de f dans le contexte de la valuation v. En particulier, $P_a^{\mathcal{M}}(\rho)(v)$ représente la probabilité valuée de l'exécution ρ. Ainsi, il est évident que, pour toute valuation valide v et toute exécution $\rho \in \Gamma_{\mathcal{M}}$ nous avons $\mathbb{P}_{\mathcal{M}^v}(\rho) = P_a^{\mathcal{M}}(\rho)(v)$.

4.3 Model Checking Statistique

Le *Model Checking statistique* [37], est une technique d'approximation qui permet de calculer une estimation de la probabilité pour qu'un système purement probabiliste satisfasse une propriété donnée [1]. En particulier, la technique de *Monte Carlo* utilise des échantillons des exécutions de longueur l, $\Gamma(l)$, d'une chaîne de Markov donnée $\mathcal{M}$ pour estimer la probabilité pour que $\mathcal{M}$ satisfasse une propriété linéaire bornée donnée. Elle peut aussi être utilisée pour approximer l'espérance d'une fonction de récompense r sur les exécutions $\Gamma(l)$ de $\mathcal{M}$. Pour donner une intuition, nous rappelons brièvement comment fonctionne la méthode standard d'analyse de Monte Carlo dans le contexte du *model checking statistique* de chaînes de Markov. Dans ce contexte, nous avons un ensemble de n échantillons des exécutions de la chaîne de Markov. Ces exécutions sont générées de façon aléatoire en utilisant la distribution de probabilité définie au travers de la chaîne de Markov. Chacune de ces exécutions est évaluée,

1. Des techniques spécifiques des SMC permettent aussi d'estimer la satisfaction de propriétés qualitatives [38].

fournissant une valeur en fonction de la fonction de récompense r.

Selon la loi des grands nombres [39], la valeur moyenne des échantillons fournit un bon estimateur pour l'espérance de la fonction de récompense r sur les exécutions de la chaîne donnée. De plus, le théorème central limite (ou théorème de la limite centrale) fournit un intervalle de confiance qui dépend seulement du nombre d'échantillons (pourvu que ce nombre soit suffisamment grand).

Le théorème central limite. Soit $\Omega = X_1, X_2, \ldots$ un ensemble de variables aléatoires, contenant des variables aléatoires indépendantes distribuées identiquement ; chaque variable a une moyenne γ et une variance σ^2.

Alors la distribution de

$$\frac{X_1 + \cdots + X_n - n\gamma}{\sigma\sqrt{n}}$$

tend vers la distribution standard normale lorsque $n \to \infty$. Pour $-\infty < a < \infty$, nous obtenons

$$\lim_{n\to\infty} \mathbb{P}\left(\frac{X_1 + \cdots + X_n - n\gamma}{\sigma\sqrt{n}} \leq a\right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{a} e^{-x^2/2} dx.$$

Ce résultat permet en particulier de borner la précision de l'estimation en fonction du nombre d'échantillons n (si n est suffisamment grand).

4.4 Model Checking Statistique Paramétrique (pSMC)

Le *model checking statistique* (SMC) standard ne peut pas être utilisé tel quel pour les pMC (chaînes de Markov paramétrées) du fait de la nature des paramètres. En effet nous ne pouvons pas fournir des échantillons selon les transitions probabilistes paramétrées. Néanmoins, la théorie sous-jacente au SMC peut être étendue pour prendre en compte des paramètres. La méthode de *model checking statistique paramétrique* que nous proposons dans la suite est similaire à la technique nommée *importance sampling*—échantillonage préférentiel [39]. Le but de cette dernière technique est d'échantillonner un système stochastique en utilisant une distribution de probabilité choisie (différente de celle donnée dans le système) et de compenser les résultats

en utilisant un coefficient de vraissemblance (*likelihood ratio*) afin d'estimer une mesure en fonction de la distribution initiale. Dans le contexte du SMC, l'échantillonnage préférentiel a surtout été utilisé pour estimer la probabilité d'événements rares [40] ou pour réduire le nombre d'échantillons nécessaires pour obtenir un certain niveau de garantie [41]. Cette technique a aussi été utilisée dans le contexte de chaînes de Markov à temps continu paramétrées pour estimer la valeur d'une fonction d'objectif donnée, sur tout l'espace d'états des paramètres tout en utilisant un nombre réduit d'échantillons [42]. Cependant, à notre connaissance, la technique d'échantillonnage préférentiel n'avait jamais été utilisée pour obtenir des fonctions symboliques des paramètres, comme nous le proposons ici.

L'intuition de la méthode (SMC paramétrique) que nous proposons est de fixer les probabilités de transition à une fonction arbitraire f que nous appelons fonction de normalisation, et d'utiliser ces probabilités de transition pour produire des échantillons de la chaîne de Markov probabiliste $\mathcal{M}$.

Cependant, au lieu d'évaluer les exécutions obtenues en utilisant directement la fonction de récompense r comme dans le SMC classique, nous définissons une nouvelle fonction de récompense paramétrique r' qui tient compte des probabilités de transition paramétriques et des coefficients de vraisemblance. Nous montrons alors que pour toute valuation valide des paramètres v, l'évaluation de la valeur moyenne de r' sur un ensemble d'échantillons est un bon estimateur de la valeur moyenne de r sur $\mathcal{M}^v$. Le théorème central limite [39] permet aussi de fournir des intervalles de confiance paramétriques, nous détaillons cela à la fin de cette section.

Remarque. Le choix de la fonction de normalisation est crucial. En particulier, les résultats présentés ci-dessous exigent que la structure de graphe de la chaîne de Markov obtenue avec cette fonction soit identique à celle de la chaîne de Markov obtenue en utilisant la valuation de paramètres choisie. Le lecteur trouvera plus de détails dans [43]. Dans la suite, nous considérons uniquement des valuations de paramètres qui affectent des probabilités non nulles aux transitions paramétrées. Puisque nous utilisons une fonction de normalisation uniforme, les graphes des MCs obtenues sont identiques ; c'est la garantie que les résultats présentés ci après sont bien ceux attendus.

Soit $\mathcal{M}$ une pMC. Nous avons montré précédemment que pour toute valuation valide v et toute exécution $\rho \in \Gamma_{\mathcal{M}}$ on a $\mathbb{P}_{\mathcal{M}^v}(\rho) = {P_a}^{\mathcal{M}}(\rho)(v)$. Pour toute fonction de normalisation valide f et toute exécution $\rho \in \Gamma_{\mathcal{M}}$, considérons une fonction de récom-

pense paramétrique r' telle que $r'(\rho) = \frac{{P_a}^{\mathcal{M}}(\rho)}{\mathbb{P}_{\mathcal{M}^f}(\rho)} r(\rho)$.
Prouvons maintenant que les espérances (notées $\mathbb{E}^l$) de r et r' sont égales pour toute valuation valide des paramètres. Soit $\rho \in \Gamma_{\mathcal{M}^f}(l)$ un échantillon aléatoire de $\mathcal{M}^f$ et soit Y la variable aléatoire définie par $Y = r'(\rho)$. Les calculs suivants démontrent que, pour toute valuation valide v telle que $\mathcal{M}^f$ et $\mathcal{M}^v$ ont la même structure, on a $\mathbb{E}^l(Y)(v) = \mathbb{E}^l_{\mathcal{M}^v}(r)$.

$$
\begin{aligned}
\mathbb{E}^l(Y)(v) &= \left(\sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} \mathbb{P}_{\mathcal{M}^f}(\rho) r'(\rho) \right)(v) \\
&= \left(\sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} \mathbb{P}_{\mathcal{M}^f}(\rho) \frac{{P_a}^{\mathcal{M}}(\rho)}{\mathbb{P}_{\mathcal{M}^f}(\rho)} r(\rho) \right)(v) \\
&= \sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} {P_a}^{\mathcal{M}}(\rho)(v) r(\rho) \\
&= \sum_{\rho \in \Gamma_{\mathcal{M}^f}(l)} \mathbb{P}_{\mathcal{M}^v}(\rho) r(\rho) \\
&= \sum_{\rho \in \Gamma_{\mathcal{M}^v}(l)} \mathbb{P}_{\mathcal{M}^v}(\rho) r(\rho) \\
&= \mathbb{E}^l_{\mathcal{M}^v}(r)
\end{aligned}
$$

Notre adaptation de la technique de Monte Carlo pour les pMC consiste ainsi à estimer l'espérance de Y pour obtenir un bon estimateur de l'espérance de la fonction r.

Soit $\{\rho_1, \ldots, \rho_n\}$ un ensemble de n exécutions de longueur l de $\mathcal{M}^f$. Soit Y_i la variable aléatoire à valeurs dans $Poly(\mathbb{X})$ telle que $Y_i = r'(\rho_i)$. Notons que les Y_i sont des copies indépendantes de la variable aléatoire Y. Les Y_i sont par conséquent indépendantes et distribuées de façon identique.

Soit γ la fonction paramétrique donnant leur valeur moyenne. A travers ces résultats, pour toute valuation valide v telle que $\mathcal{M}^v$ et $\mathcal{M}^f$ ont la même structure, $\mathbb{E}^l_{\mathcal{M}^v}(r) = \mathbb{E}(Y)(v) = \mathbb{E}(\sum_{i=1}^n Y_i/n)(v) = \gamma(v)$. Notre approximation paramétrique de la valeur attendue est par conséquent :

$$\widehat{\gamma} = \textstyle\sum_{i=1}^n Y_i/n.$$

Nous utilisons maintenant le théorème central limite pour calculer l'intervalle de confiance associé à cette estimation.

L'intervalle de confiance. Comme notre estimation $\widehat{\gamma}$, l'intervalle de confiance obtenu sera donné par des fonctions paramétriques. L'ensemble des variables aléatoires (Y_i) sont indépendantes et suivent la même distribution. Chaque variable a une moyenne γ et une variance σ^2.

L'espérance et la variance de leur somme $Y_1+\cdots+Y_n$ sont les suivantes : $\mathbb{E}(Y_1+\cdots+Y_n) = n\gamma$ et $Var(Y_1+\cdots+Y_n) = n\sigma^2$. Rappelons que γ est une fonction paramétrique. Soit Z une variable aléatoire paramétrique telle que

$$Z = \frac{(Y_1+\cdots+Y_n-n\gamma)}{\sqrt{n\sigma^2}}.$$

D'après le théorème central limite, $Z(v)$ tend vers la distribution normale standard $\mathcal{N}(0,1)$, pour toutes les valuations valides des paramètres v telles que $\mathcal{M}^v$ et $\mathcal{M}^f$ ont la même structure, lorsque $n \to \infty$. Par conséquent, pour toutes les valuations valides des paramètres v telles que $\mathcal{M}^v$ et $\mathcal{M}^f$ ont la même structure, en supposant que n est assez grand, nous obtenons

$$\mathbb{P}(-z \leq Z(v) \leq z) \approx \varphi(z) - \varphi(-z) = 2\varphi(z) - 1,$$

où $\varphi(z) = \int_{-\infty}^{z} e^{-x^2/2}dx/(\sqrt{2\pi})$ est la fonction de distribution cumulative de la distribution normale standard $\mathcal{N}(0,1)$. Alors, par la définition de Z, nous obtenons que pour toutes les valuations des paramètres valides v telles que $\mathcal{M}^v$ et $\mathcal{M}^f$ ont la même structure, et pour un n assez grand,

$$2\varphi(z) - 1 \approx \mathbb{P}\left(\gamma(v) - z\frac{\sigma(v)}{\sqrt{n}} \leq \widehat{\gamma}(v) \leq \gamma(v) + z\frac{\sigma(v)}{\sqrt{n}}\right).$$

L'intervalle aléatoire paramétrique $I = (\widehat{\gamma} - z\sigma/\sqrt{n}, \widehat{\gamma} + z\sigma/\sqrt{n})$ est donc un intervalle de confiance pour γ avec une précision $2\varphi(z) - 1$. Nous utilisons généralement $z = 1.96$ puisque $2\varphi(1.96) - 1 = 0.95$ (ou $z = 2.56$ pour la précision 0.99). Selon notre hypothèse, cet intervalle de confiance paramétrique n'est valable que pour les valuations des paramètres valides v telles que $\mathcal{M}^v$ et $\mathcal{M}^f$ ont la même structure. Il est important de rappeler que la taille de l'intervalle de confiance pour $\widehat{\gamma}$ est également paramétrique : elle est égale à $2 \cdot z\frac{\sigma}{\sqrt{n}}$. En effet, la valeur de σ dépend de la valuation des paramètres et du choix de la fonction de normalisation. Nous expliquons maintenant comment calculer l'estimation paramétrique de σ. De la théorie des probabilités classique, nous

savons qu'un estimateur non biaisé pour la variance σ^2 est :

$$\widehat{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n} Y_i^2 - \widehat{\gamma^2}.$$

TABLE 4.1 – Exemple de pSMC avec 10 simulations

$0.1Poly_1$	$0.2Poly_1$	$0.3Poly_1$	$0.1Poly_2$	$0.1Poly_2$	$0.1Poly_2$	$0.2Poly_2$	$0.2Poly_2$	0	0

Ici nous montrons un exemple avec juste 10 simulations pour expliquer comment pSMC calcule. Dans les 10 simulations présentées dans le tableau 4.1, nous avons atteint 8 fois notre espérance r, avec les variables aléatoires définies par $Y_i = r_i'(\rho_i) = c_i Poly_1 (i = 1 \text{ à } 3)$, $Y_i = r_i'(\rho_i) = c_i Poly_2 (i = 4 \text{ à } 8)$ et $Y_i = 0 (i = 9 \text{ et } 10)$; r_i' est la fonction de récompense de chaque exécution finie de chaque simulation ρ_i ; les c_i sont des coefficients et $Poly_j$ sont des paramètres. L'espérance de ce pMC $\mathcal{M}$ avec valuation valide v est donc $\mathbb{E}_{\mathcal{M}^v}(r) = \mathbb{E}(\sum_{i=1}^{n} Y_i/n)(v) = \gamma(v)$ où $\gamma(v)$ est la fonction paramétrique donnant leur valeur moyenne. Notre approximation paramétrique de la valeur attendue avec une valuation valide v ($Poly_1 = 0.1, Poly_2 = 0.9$) est donc $\widehat{\gamma} = \sum_{i=1}^{n} Y_i/n = \frac{(c_1+c_2+c_3)Poly_1+(c_4+c_5+c_6+c_7+c_8)Poly_2}{10} = 0.06Poly_1 + 0.07Poly_2 = 0.069$.
L'espérance et la variance de somme $Y_1 + \dot{+} Y_n$ sont $n\gamma$ et $n\sigma^2$. D'après le théorème central limite, notre intervalle aléatoire paramétrique $I = (\widehat{\gamma} - z\sigma/\sqrt{n}, \widehat{\gamma} + z\sigma/\sqrt{n})$ est donc un intervalle de confiance pour γ avec une précision $2\varphi(z) - 1$. Pour une précision de 0.95, $z = 1.96$ et la variance σ^2 avec une valuation valide v l'estimateur est $\widehat{\sigma}^2 = \frac{\sum_{i=1}^{n}(Y_i - \widehat{\gamma})^2}{n} = \frac{1}{n}\sum_{i=1}^{10} Y_i^2 - \widehat{\gamma}^2 = \frac{0.14Poly_1^2+0.11Poly_2^2}{10} - (0.069)^2 = 0.004289$. Notre intervalle avec une valuation valide v est $I = \widehat{\gamma} \pm z\sigma/\sqrt{n} = 0.069 \pm 1.96 * \sqrt{0.004289}/\sqrt{10} = (0.0284, 0.110)$.

Dans la suite nous utiliserons ce *model checking statistique paramétrique* (*Parametric Statistical Model Checking — pSMC)* pour vérifier le modèle formel construit pour le drone.

4.5 Les outils de model checking

Il existe de nombreux outils de *model checking*. Dans cette section, nous introduisons les outils que nous avons utilisés.

4.5.1 PRISM

PRISM [6, 7] est un outil de modélisation et d'analyse formelle de systèmes affichant un comportement aléatoire ou probabiliste. Il a été utilisé pour analyser des systèmes appartenant à de nombreux domaines d'application, notamment les protocoles de communication et multimédia [44], les algorithmes distribués aléatoires [45], les protocoles de sécurité [46], les systèmes biologiques [47] et bien d'autres [48] [49].

```
dtmc

//p0n: probability of transition state 0 -> state 1n
const double p01 = 0.2;
const double p02 = 0.2;
const double p03 = 0.2;
const double p04 = 0.2;
const double p05 = 1-p01-p02-p03-p04;

//p1m: probability of transition state 1 -> state 2m
const double p11 = 0.2;
const double p12 = 0.2;
const double p13 = 0.2;
const double p14 = 0.2;
const double p15 = 1-p11-p12-p13-p14;

module main

                // 0 state 0
                // 1 state 1i (i is zone1)
                // 2 state 2j (j is zone2)
                s: [0..2] init 0;

                // 0 null
                // 1-5 zone 1-5
                zone1: [0..5] init 0;
                zone2: [0..5] init 0;

                [ ] s=0 -> p01: (zone1' = 1) & (s' = 1)
                              + p02: (zone1' = 2) & (s' = 1)
                              + p03: (zone1' = 3) & (s' = 1)
                              + p04: (zone1' = 4) & (s' = 1)
                              + p05: (zone1' = 5) & (s' = 1);

                [ ] s=1 -> p11: (zone2' = 1) & (s' = 2)
                              + p12: (zone2' = 2) & (s' = 2)
                              + p13: (zone2' = 3) & (s' = 2)
                              + p14: (zone2' = 4) & (s' = 2)
                              + p15: (zone2' = 5) & (s' = 2);

endmodule
```

Listing 4.1 – Code PRISM d'un modèle simplifié.

Pour illustrer, le Listing 4.1 contient le code PRISM d'un automate représentant une version simplifiée du fonctionnement du drone. Les constantes p_{01}, p_{02}, p_{03}, p_{04} et p_{05} correspondent aux transitions $s1$, $s2$, $s3$, $s4$ et $s5$ de l'automate (voir Figure 4.2). Les constantes p_{11}, p_{12}, p_{13}, p_{14} et p_{15} correspondent aux transitions a, b, c, d et e (voir Figure 4.2).

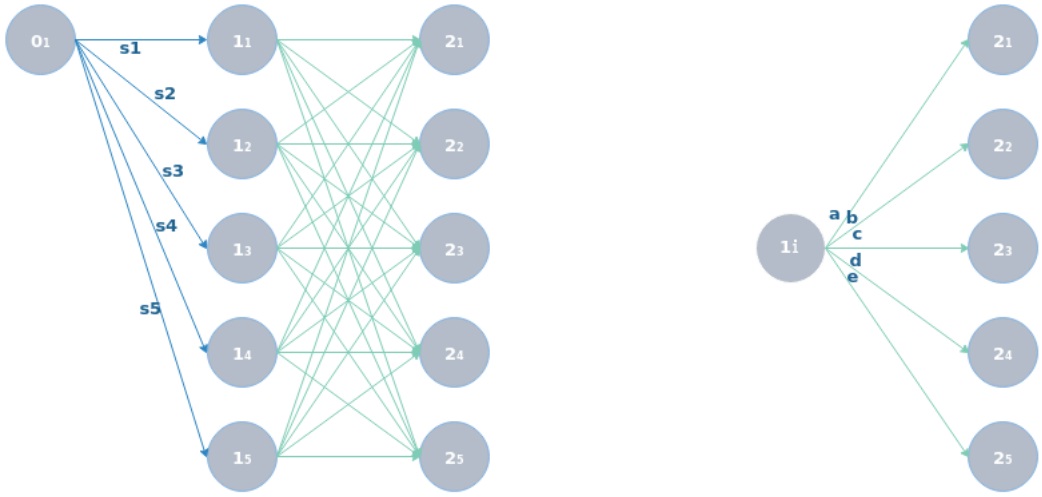


FIGURE 4.2 – L'automate du code PRISM.

Ces transitions sont des transitions probabilistes. Comme nous l'avons présenté plus haut, la somme des probabilités des transitions sortant d'un état est égale à 1. Les principaux états sont 0_1, 1_i et 2_i. Sur les lignes 29 et 35, les $[\,]$ montrent l'absence de signal [2]. Dans signal, cette notation permet d'exprimer la synchronisation entre plusieurs modules ; elle sert aussi à ajouter des conditions dans le modèle. Nous l'avons utilisée (lignes 29 et 35) pour synchroniser des calculs.

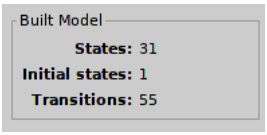


FIGURE 4.3 – Nombre d'états et de transitions dans PRISM.

Le modèle du Listing 4.1 décrit 31 états. Les états sont de la forme : $s, zone_1, zone_2$, nous avons donc $(0, 0, 0)$ l'état initial, $(1, [1, 5], 0)$ qui représente 5 états liés par 5 transitions à l'état $(0, 0, 0)$ et $(2, [1, 5], [1, 5])$ qui représente 25 états liés par 50 transitions aux états $(1, [1, 5], 0)$. PRISM nous informe donc que nous avons 31 états, 1 état initial et 55 transitions (voir Figure 4.3).

2. PRISM permet l'utilisation de transitions avec ou sans signal.

PRISM est un outil adapté pour la technique de *Model Checking probabiliste standard*. Il permet de calculer la probabilité pour qu'un système purement probabiliste satisfasse une propriété donnée.

4.5.2 PARAM

PARAM [50] est un vérificateur de modèles pour les chaînes de Markov paramétriques (pMC) à temps discret. PARAM peut évaluer les propriétés temporelles des pMC et certaines extensions de cette classe. En raison de la prise en compte des paramètres, les résultats de l'évaluation sont des fonctions rationnelles dont les termes sont exprimés à l'aide des paramètres présents dans le modèle d'entrée. Avec des valeurs des paramètres, la fonction se réduit à une valeur de probabilité inférieure ou égale à 1.

PARAM utilise le langage de PRISM en entrée, mais permet l'utilisation de paramètres à la place des constantes dans le modèle. Les paramètres sont définis de la façon suivante `param float <parameter>;` ou `param <parameter> : [<p1>..<p2>];`.

Notre modèle en PARAM sera comme dans le Listing 4.1, en remplaçant les lignes 4 à 15 par celles du Listing 4.2.

```
1  dtmc
2
3  //p0n: probability of transition state 0 -> state 1n
4  param float p01;
5  param float p02;
6  param float p03;
7  param float p04;
8  param float p05;
9
10 //p1m: probability of transition state 1 -> state 2m
11 param float p11;
12 param float p12;
13 param float p13;
14 param float p14;
15 param float p15;
```

Listing 4.2 – Code PARAM d'un modèle simplifié.

PRISM ne peut pas vérifier un pMC, mais PARAM est fait pour cela. Il permet de calculer la probabilité pour qu'un système probabiliste avec des paramètres satisfasse une propriété donnée.

4.5.3 MCpMC

MCpMC [43] [3] est un prototype développé dans le but d'expérimenter le *model checking statistique paramétrique* (pSMC) pour les chaînes de Markov paramétriques. Il est basé sur une adaptation paramétrique de l'analyse de Monte Carlo standard (présentée en section 4.4). Comparé aux vérificateurs avec les techniques de vérification exactes, il a les mêmes avantages que le *model checking statistique* pour les modèles non paramétriques : une meilleure évolutivité et une indépendance vis à vis de la complexité du modèle. Contrairement aux techniques de vérification de modèles statistiques existantes pour les systèmes non déterministes, pSMC permet de calculer des intervalles de confiance paramétriques, ce qui offre une garantie de la précision des estimations.

```
1  import model
2  from sympy import symbols
3  import random
4
5  ProbaFilter1, ProbaFilter2, ProbaFilter3, ProbaFilter4, ProbaFilter5 = \
6          symbols('ProbaFilter1, ProbaFilter2, ProbaFilter3, ProbaFilter4,
               ProbaFilter5')
7
8  DistanceZoneSecurity0 = 60
9  DistanceZoneFilter = [20,40,60,80,100]
10 DistanceZoneFilter_ProbaFilter = [[20, ProbaFilter1], [40, ProbaFilter2],
11                  [60, ProbaFilter3], [80, ProbaFilter4], [100, ProbaFilter5]]
12
13 period = 1
14 Time = 5
15
16 # States of drone are represented as:
17 # [lastCorrection , times]
18 class Drone(model.AbstractPMC):
19
20         def getDistance0(self):
21                 return DistanceZoneSecurity0
22         def getFrequency(self):
23                 return period
24         def getTime(self):
25                 return Time
26
27         def initial(self):
28                 return [0, 0]
29
```

3. Disponible à l'adresse : *https ://github.com/paulinfournier/MCpMC*

```
30          def next(self, a_state):
31                  startingPoint, times = a_state
32
33                  y = random.choice(range(-100,100))
34
35                  if (y <= DistanceZoneFilter[0])
36                  and (y >= -DistanceZoneFilter[0]):
37                          P = ProbaFilter1/(41/200)
38
39                  elif (y <= DistanceZoneFilter[1])
40                  and (y >= -DistanceZoneFilter[1]):
41                          P = ProbaFilter2/(40/200)
42
43                  elif (y <= DistanceZoneFilter[2])
44                  and (y >= -DistanceZoneFilter[2]):
45                          P = ProbaFilter3/(40/200)
46
47                  elif (y <= DistanceZoneFilter[3])
48                  and (y >= -DistanceZoneFilter[3]):
49                          P = ProbaFilter4/(40/200)
50
51                  else:
52                          P = ProbaFilter5/(39/200)
53
54                  proportion = period / (Time - period * a_state[1])
55                  a = startingPoint+(0-(y+startingPoint))*proportion
56
57                  return [P], [[ a, times + 1]]
58
59          # return [finish condition, reward condition]
60          def end(self, a_state):
61                  position, times = a_state
62                  return [(times >= Time/period) or
63                          (position > DistanceZoneSecurity0) or
64                          (position < -DistanceZoneSecurity0) ,
65                          (position > DistanceZoneSecurity0) or
66                          (position < -DistanceZoneSecurity0)]
```

Listing 4.3 – Code en Python (pour MCpMC) pour un modèle de drone.

Le Listing 4.3 présente un modèle en entrée de MCpMC. Le codage des modèles d'entrée de MCpMC se fait avec le langage Python. Le langage Python est plus connu des programmeurs que le langage PRISM, et il est plus libre et plus facile à utiliser pour les programmeurs. Tout modèle d'entrée de MCpMC contient 3 fonctions principales : `initial`, `next` et `end`.

La fonction `initial` permet l'initialisation du modèle, en donnant les valeurs de

toutes les variables. La fonction `next` permet de boucler en changeant certaines des valeurs de l'état, elle réalise le changement d'état dans une chaîne de Markov. La fonction `end` permet de vérifier si le prochain état est un des états finaux, et renvoie la valeur de la fonction de récompense de la trace obtenue si c'est le cas.Dans la fonction `initial`, nous donnons l'état initial du modèle, ici les variables `startingPoint=0`, `times=0` (voir les lignes 27 et 28). La fonction `next` permet de calculer le prochain état de la trace courante, et renvoie la probabilité avec laquelle il est obtenu afin de permettre à l'outil de calculer le coefficient de vraisemblance (voir section 4.4). Ce prochain état sera de la forme `(startingPoint, times)` ; avant de rappeler `next`, la fonction `end` est appelée pour tester si cet état est terminal (voir les lignes 30 à 57). Si c'est le cas, la simulation se termine et `end` renvoie un booléen permettant de déterminer si la trace obtenue satisfait la propriété souhaitée (vois les lignes 60 à 66). Si la simulation est terminée, MCpMC utilise ce booléen et les informations fournies par la fonction `next` pour calculer la valeur de r' appliquée à la trace obtenue. Sinon, la fonction `next` est appelée à nouveau.

```
1  import drone2
2  from sympy import symbols
3  import time
4
5  ProbaFilter1, ProbaFilter2, ProbaFilter3, ProbaFilter4, ProbaFilter5 = \
6          symbols('ProbaFilter1,ProbaFilter2,ProbaFilter3,ProbaFilter4,ProbaFilter5')
7
8  start = time.time()
9  p, icw_p, q, icw_q = drone2.Drone().simulate(10000)
10 print("polynomial p(reward): ", p)
11
12 d = {ProbaFilter1:0.5, ProbaFilter2:0.2, ProbaFilter3:0.15, ProbaFilter4:0.1,
       ProbaFilter5:0.05}
13
14 pd = p.subs(d)
15 icw_pd = icw_p.subs(d)
16 print(" p: ",pd, " icw: ", icw_pd)
17
18 print("Computation duration (s): ", time.time()-start)
```

Listing 4.4 – Code en Python pour un modèle de drone.

L'importation et l'utilisation du modèle avec MCpMC est illustré dans le Listing 4.4, où la ligne 9 permet de lancer 10 000 simulations du modèle *drone2*. La variable `p`

contient le polynôme pour calculer la probabilité de satisfaire notre propriété (arriver dans l'état zone de sécurité 4 ou 5) ; la variable `icw_p` contient l'intervalle de confiance correspondant. De la même façon `q` et `icw_q` représentent le polynôme et l'intervalle de confiance pour ne pas satisfaire la propriété. En effet, la probabilité de satisfaire une propriété n'est pas calculée directement, mais exprimée en fonction des différentes probabilités utilisées comme paramètres ; ce qui est exprimé par un polynôme ; nous détaillerons cette construction dans la section 5.8.2.

Dans certains cas, nous donnons des valeurs à certains paramètres afin d'évaluer les polynômes générés. Nous codons cela comme un dictionnaire en Python `(d = paramètre1: valeur, ...`, voir ligne 12 du Listing 4.4). La ligne 14 (Listing 4.4) exprime l'évaluation du polynôme `p` avec les valeurs de paramètres contenues dans `d`.

CHAPITRE 5

MÉTHODE DE MODÉLISATION FORMELLE DE DRONES CIVILS

Dans ce chapitre, nous présentons notre méthode pour construire un modèle formel d'un drone civil. Rappelons que nous sommes intéressés par l'étude de la sécurité des drones, c'est-à-dire l'étude de la probabilité qu'un drone se trouve dans des situations dangereuses pour son environnement, notamment humain. D'abord nous présentons notre modèle global. Ensuite nous allons présenter pas à pas la construction de ce modèle en partant d'une version de base qui est améliorée successivement, par rapport aux limitations identifiées à chaque étape de la construction.

5.1 Un aperçu du modèle de drone final

Le modèle global du système de contrôle de vol du drone est présenté dans la Figure 5.1. Le but de ce modèle est de représenter les calculs qui ont lieu dans le contrôleur de vol (FCS - *Flight control system*) afin d'adapter la trajectoire du drone au plan de vol prévu, aussi bien en fonction des imprécisions des estimations des mesures de position et d'orientation, que des perturbations par le vent. Nous utilisons le modèle pour calculer les positions du drone et compter le nombre de simulations qui entrent dans une zone de sécurité donnée, en fonction des paramètres.

Dans ce modèle, la position exacte du drone est encodée avec ses deux coordonnées dans le plan. En effet, le plan de vol est établi sans prendre en compte l'altitude dans la plupart des cas. Ces coordonnées sont ensuite comparées avec celles décrivant le plan de vol pour décider à quelle zone de sécurité elles correspondent. Dès que le drone a atteint une zone interdite (les zones 4 ou 5), le calcul s'arrête, sinon il continue jusqu'à épuiser le temps de parcours correspondant au plan de vol.

Le modèle utilise plusieurs paramètres probabilistes : les paramètres FilterProba1,

FilterProba2, FilterProba3, FilterProba4 et FilterProba5 (figure 5.1) représentent la précision de la position et de l'orientation estimées par le filtre et les capteurs. Le choix probabiliste résultant, représenté dans la boîte étiquetée **Filter Computation** (figure 5.1) donne alors la distance entre la position exacte du drone et celle estimée par les capteurs et le filtre.

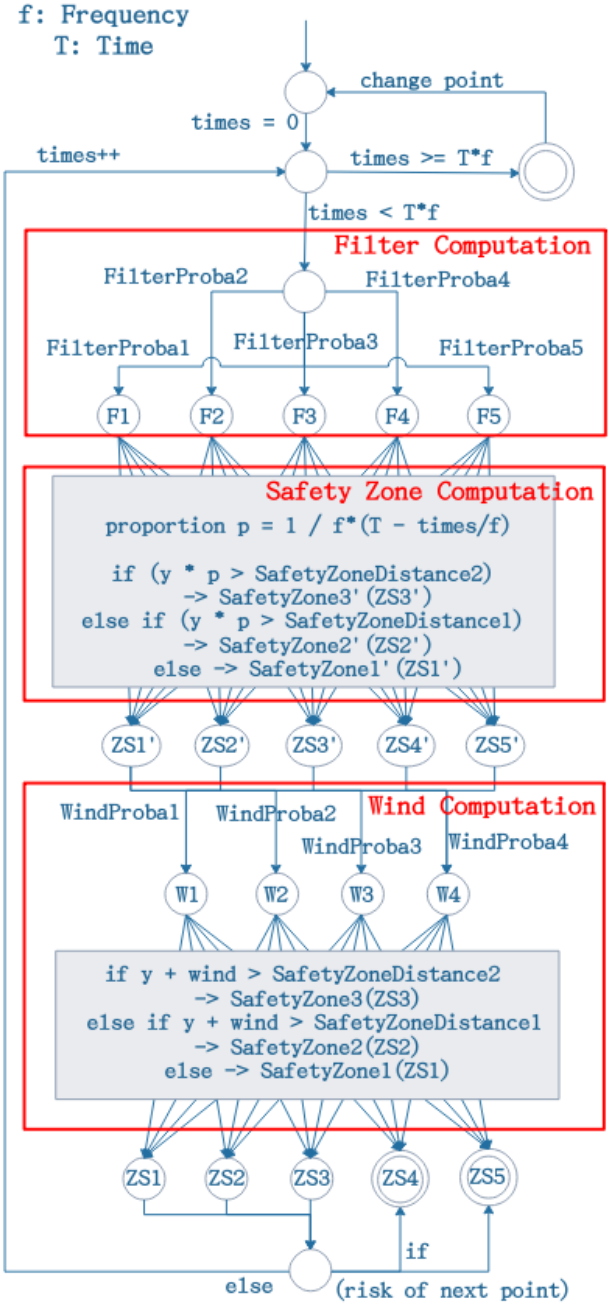


FIGURE 5.1 – Modèle global du FCS.

Ce choix est suivi par un calcul dans la boîte étiquetée **Safety Zone Computation** (voir Figure 5.1) qui détermine les coordonnées exactes de la prochaine position du drone, et permet de décider à quelle zone de sécurité appartient cette position. Lorsque le vent n'est pas pris en compte, le résultat de ce calcul est suffisant pour décider si le modèle doit poursuivre son déroulement. Lorsque le vent est pris en compte, une autre étape suit. Elle est indiquée par la boîte étiquetée **Wind Computation** ; d'autres paramètres probabilistes y sont utilisés pour décider de la force du vent (nous supposons une direction constante) et une nouvelle position tenant compte de ces perturbations est calculée. Finalement, la zone de sécurité à laquelle appartient cette dernière position est calculée. Pour une durée de temps (T) donnée, nous effectuons, en fonction de la fréquence du filtre considéré, plusieurs itérations du calcul de la position du drone et la simulation s'arrête dès qu'il arrive dans un état des zones de

sécurité interdites.

Notons que la fréquence du filtre (f) ainsi que les positions et distances des points de contrôle dans le plan de vol sont données comme entrées du modèle. La position du point de contrôle dans le plan de vol permet de calculer la vitesse du drone, alors que la fréquence du filtre permet de fixer le nombre d'estimations de positions qu'il y aura dans un plan de vol donné (i.e. le nombre maximum d'itérations que le modèle doit effectuer). Ce modèle permet ainsi de simuler l'évolution des drones pour un plan de vol et pour des paramètres choisis. Les résultats des simulations sont exploités pour analyser le comportement du drone et régler ainsi les valeurs des paramètres, afin d'obtenir le minimum d'entrées dans les zones de sécurité interdites.

5.2 Méthode de construction du modèle

Une première approximation, systématique, pour abstraire le comportement du drone consiste à construire un modèle à états où de l'état initial représentant la position exacte actuelle, nous passons à l'état après la lecture des capteurs du drone ; de cet état il y a une famille de transitions vers un état correspondant à l'état après le traitement effectué par le filtre. Ensuite une famille de transitions mènera à l'état après les traitements du PID, puis une autre famille de transitions permettra d'atteindre l'état final.

Les transitions suivent donc la structuration du contrôleur de vol. Ce cycle de transitions est continuellement répété pendant l'évolution du drone. Dans notre approche de modélisation, les états les plus importants sont ceux issus du filtre et du PID. A partir de cette première approximation, il s'agit de préciser avec divers focus de recherche, les effets et états possibles avec un filtre ou un modèle de filtre, et de même pour le PID. Dans la mesure où nous focalisons notre étude sur les problématiques de perte de trajectoire, nous approfondissons l'étude du filtre. Il y a différents modèles de filtre ; ils utilisent des algorithmes différents donc il n'est pas possible de construire un modèle formel général unique pour les filtres. Par conséquent, au lieu de chercher à modéliser le filtre, nous allons de façon plus pratique, juste utiliser les données du filtre ; c'est-à-dire construire un modèle du contrôleur de vol d'un drone automatique qui commence par estimer la position à l'aide des capteurs, puis corrige cette position à l'aide des données du filtre, transmet cette information au PID puis à la modulation et finalement aux ESC (moteurs).

Les familles de transitions évoquées à chaque étape seront matérialisées avec des probabilités. En conséquence, notre méthode de modélisation de drone consiste à :

— construire une chaîne de Markov (paramétrique) du contrôleur de vol général,
— compléter les transitions avec les probabilités,
— préciser certains états du modèle en utilisant les signaux sur les transitions,
— améliorer et compléter le modèle en fonction des limitations identifiées à chaque étape.

Nous détaillons maintenant cette méthode générale.

5.3 Modèle du drone à l'aide d'une chaîne de Markov

Dans un environnement idéal, les composants qui influent le plus sur un changement de trajectoire sont les capteurs et le filtre. Nous proposons à l'aide d'une chaîne de Markov standard un premier modèle formel du drone avec ces deux composants. Le modèle simule la position exacte avec l'état 0_1. Les états 1_i représentent les positions estimées par les capteurs. Finalement les états 2_j représentent les états pour les positions après le filtrage des valeurs des capteurs par le filtre (voir Fig 5.2).

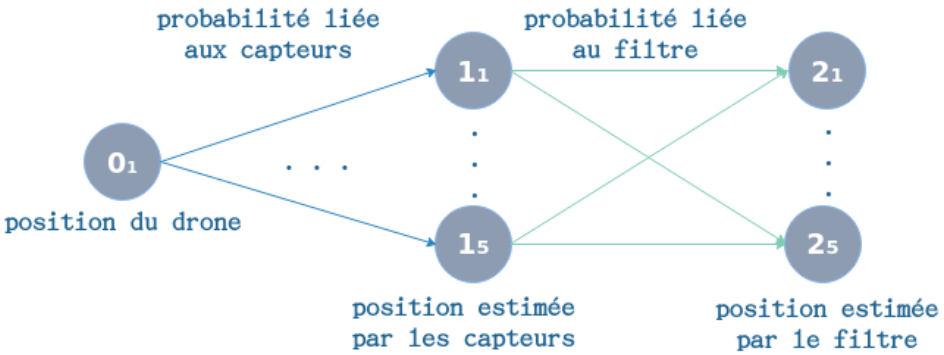


FIGURE 5.2 – Abstraction du modèle.

L'état initial 0_1 est donc la position exacte. Les états correspondant aux positions mesurées à l'aide des capteurs sont 1_1, 1_2, 1_3, 1_4 et 1_5. Les états correspondant aux positions après correction par le filtre sont 2_1, 2_2, 2_3, 2_4 et 2_5. Les probabilités des transitions dépendent donc de la précision des capteurs et du filtre. Ces transitions signifient que le drone estime sa position dans les zones : `zone1`, `zone2`, `zone3`, `zone4`, `zone5`. Avec la probabilité d'erreur de mesure des capteurs et de correction du

filtre, nous pouvons donc calculer la probabilité pour le drone d'entrer dans les zones interdites.

Nous notons $(0_1, P_{0i}, 1_i)$ les transitions de l'état 0_1 à l'état 1_i avec la transition étiquetée par P_{0i}, pour les différentes valeurs de i.

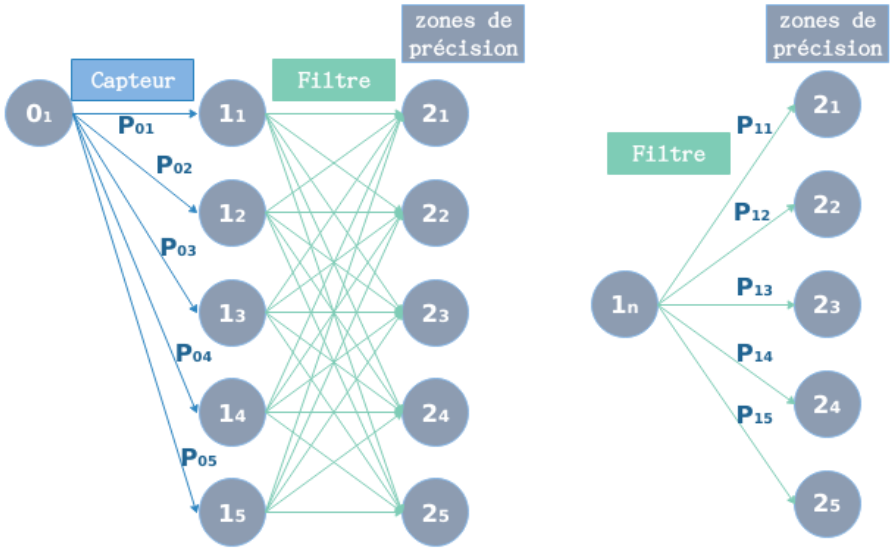
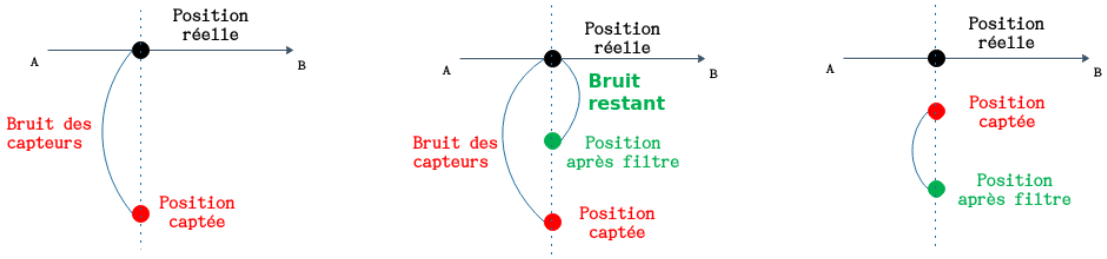


FIGURE 5.3 – Aperçu du modèle formel du drone avec une chaîne de Markov standard.

Ce premier modèle avec une chaîne de Markov standard est schématisé dans la Figure 5.3. Nous intégrons dans le comportement du drone pour aller d'un état à l'autre les transitions intermédiaires avec les probabilités relatives aux comportements des capteurs et du filtre. Les transitions bleues sont celles des capteurs, leur probabilité de donner une mauvaise ou une bonne position sont : P_{01}, P_{02}, P_{03}, P_{04}, P_{05}. Ces transitions signifient que le drone estime sa position dans les zones : `zone1`, `zone2`, `zone3`, `zone4`, `zone5`, alors que sa position est en `zone1` (l'état initial étant 0_1). Les transitions vertes sont celles du filtre ; les probabilités de corriger ou ne pas corriger le signal qu'il reçoit en entrée, sont : P_{11}, P_{12}, P_{13}, P_{14}, P_{15}. Nous ne connaissons pas les probabilités P_{01}, P_{02}, P_{03}, P_{04}, P_{05} des capteurs. Un test en conditions réelles est donc nécessaire pour compléter ce modèle avec des valeurs expérimentales. Après correction par le filtre, nous appelons la zone obtenue "la zone de précision". Ce sont les états $2_1, 2_2, 2_3, 2_4, 2_5$ dans la Figure 5.3.

Le contrôleur de vol que nous avons utilisé permet d'obtenir la position donnée par le capteur et celle à la sortie du filtre *EKF (Extended Kalman filter)*. La seule information

manquante est la position réelle du drone.



(a) (b) (c)

FIGURE 5.4 – Position réelle du drone, position captée et position après filtre.

La Figure 5.4 présente la position réelle du drone, la position captée et la position après correction par le filtre. Nous utilisons les trois valeurs pour connaître la capacité du filtre. Les bruits des capteurs en Figure 5.4a correspondent aux probabilités P_{01}, P_{02}, P_{03}, P_{04}, P_{05} dans le modèle de la Figure 5.3. La zone de précision dépend donc du bruit restant calculé avec la méthode de la Figure 5.4b. Selon le bruit dans chaque zone, la probabilité P_{1j} que la position venant des capteurs soit bien corrigée en $\texttt{zone}j$ peut ne pas être identique. Notons que la correction par le filtre peut aussi empirer la précision de l'estimation, comme illustré dans la Figure 5.4c.

5.4 Première version du modèle (en PRISM)

Nous avons premièrement développé un modèle sous forme d'une DTMC (*discrete-time Markov chain*) en PRISM [6, 7] (voir section 4.5.1). Ce modèle prend en compte les estimations des positions des capteurs, et les positions en sortie du filtre.

Dans la suite la notation $\mathcal{M}[\mathcal{E}, \mathcal{P}]$ indiquera que nous traitons du modèle $\mathcal{M}$ avec les états $\mathcal{E}$ (que nous détaillerons) et les probabilités (réelles ou paramètres) $\mathcal{P}$ (que nous détaillerons également). Cette liste d'états et de probabilités va s'allonger au fur et à mesure de l'évolution des versions du modèle.

Dans le modèle de la version 1 $\mathcal{M}_{v1}[\mathcal{E}_{v1}, \mathcal{P}_{v1}]$, nous avons $\mathcal{E}_{v1} = \{0_1, 1_1, 1_2, 1_3, 1_4, 1_5, 2_1, 2_2, 2_3, 2_4, 2_5\}$ avec 0_1 l'état initial. Les transitions probabilistes sont dans $\mathcal{P}_{v1} = \{P_{01}, P_{02}, P_{03}, P_{04}, P_{05}, P_{11}, P_{12}, P_{13}, P_{14}, P_{15}\}$; pour toutes les probabilités $p \in \mathcal{P}_{v1}$, $p \in [0, 1]$. Comme indiqué précédemment, nous aurons les transitions $(0_1, P_{0i}, 1_i)$ puis les transitions $(1_i, P_{1j}, 2_j)$. Remarquons que dans ce modèle, les probabilités d'at-

teindre un état 2_j donné sont les mêmes dans tous les états 1_i. Ainsi, quelle que soit la précision de l'estimation de la position par le capteur, la probabilité P_{2j} d'arriver dans un état 2_j donné est identique :

$$P_{2j} = \sum_{i=1}^{5}(P_{0i} * P_{1j}) = P_{1j}$$

Ceci constitue une limitation importante de notre modèle (nommée limitation L0 dans la suite). En effet, dans la réalité, la zone obtenue après correction par le filtre doit dépendre non seulement de la précision du filtre mais aussi de la précision du capteur, et donc de la position estimée. Ainsi, les probabilités d'aller dans un état 2_j donné doivent dépendre de l'état 1_i obtenu dans la première phase. Il n'est donc pas réaliste d'utiliser la même probabilité P_{1j} dans tous les états 1_i. Dans la suite, nous utiliserons donc des probabilités différentes P_{1ij}.

La probabilité P_{2j} sera alors :

$$P_{2j} = \sum_{i=1}^{5}(P_{0i} * P_{1ij})$$

De plus, une autre limitation de cette version (nommée limitation L1 dans la suite) est que la zone obtenue après correction par le filtre (appelée zone de précision) ne correspond pas directement à la zone de sécurité dans laquelle se trouvera le drone après correction de sa trajectoire (que nous appellerons zone de sécurité par la suite). Cette distinction est expliquée dans la section à venir.

5.5 Calcul de la position par rapport aux zones de sécurité

Pour résoudre la limitation L1, nous avons besoin de préciser la différence entre les zones de sécurité et les zones de précision. Il existe une différence entre la probabilité de correction du filtre et la probabilité d'être dans une zone. Les probabilités de correction du filtre donnent les probabilités d'arriver dans les zones de précision mais pas dans les zones de sécurité.

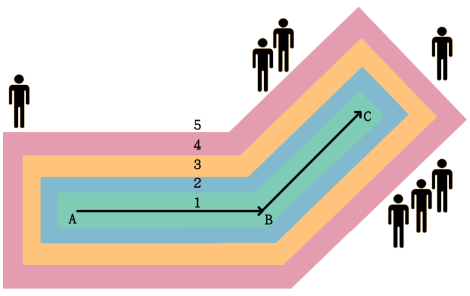


FIGURE 5.5 – Zones de sécurité.

Nous expliquons maintenant comment la prochaine position du drone est calculée en fonction de l'estimation de sa position actuelle. En particulier, nous montrons que l'imprécision dans l'estimation peut conduire le drone à pénétrer dans une zone interdite.

Nous nous inspirons de la norme DO-178C [15] définie pour l'avionique. Nous avons donc découpé les zones comme dans la Figure 5.5 pour le trajet de A à C. Nous considèrerons que la `zone1` correspond au niveau E dans DO-178C, la `zone2` au niveau D, la `zone3` au niveau C, la `zone4` au niveau B et la `zone5` au niveau A.

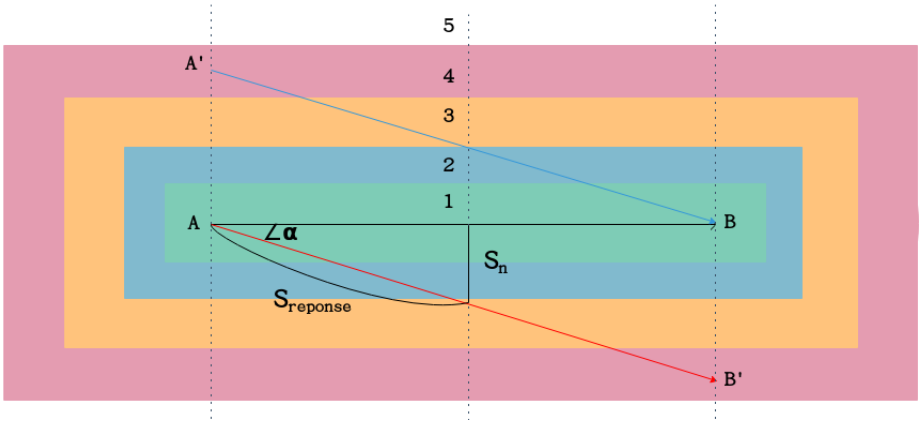


FIGURE 5.6 – Difficultés sur la localisation du drone et les positions erronées.

Dans un souci de simplicité, nous supposons que le drone se déplace seulement dans le plan et que des imprécisions apparaissent uniquement sur un des axes (ici l'axe y). Nous avons corrigé cette limitation dans la version 4. La Figure 5.6 illustre la situation. Supposons que le plan de vol envisagé consiste à aller d'un point A à un point B. Considérons aussi que la position actuelle du drone est exactement en A mais que la position estimée (prenant en compte les imprécisions des capteurs et du filtre) est en A'. Par conséquent, le PID essayera de corriger la déviation actuelle en changeant l'angle du drone de façon à le ramener vers B. Cependant, puisque le drone est effectivement en A, la correction va plutôt le conduire vers la position B' donc dans une zone interdite. Heureusement, l'estimation de la position se fait plusieurs fois entre A et B en fonction de la fréquence f du filtre. Une nouvelle position sera donc estimée avant d'atteindre B', avec l'espoir d'une meilleure précision, qui permettra au PID de corriger la trajectoire à nouveau. Nous devons aussi tenir compte du fait que la vitesse du drone est aussi calculée par rapport au plan de vol, ce qui permet d'obtenir le temps restant avant le point de contrôle suivant. Nous montrons maintenant comment nous pouvons calculer la zone de sécurité qui sera atteinte par le drone avant que sa posi-

tion ne soit de nouveau estimée. Dans la Figure 5.6 S_n représente la déviation avant la nouvelle estimation de la position.

Soit $S_{reponse}$ la distance que parcourt le drone avant l'estimation d'une nouvelle position. Nous avons :

$$S_n = \sin \alpha * S_{reponse} \tag{5.1}$$

V est la vélocité du drone, qui est calculée par le PID afin d'atteindre B à temps et $T_{reponse}$ est le temps passé avant la prochaine correction. Ainsi, nous avons :

$$S_{reponse} = V * T_{reponse} \tag{5.2}$$

Soit T_{chemin} le temps restant pour arriver au point B. Nous pouvons écrire :

$$V = \frac{A'B}{T_{chemin}} \tag{5.3}$$

Le temps complet pour aller au point B est représenté par T_B et $T_{maintenant}$ est le temps utilisé jusqu'à maintenant. Donc nous avons :

$$T_{chemin} = T_B - T_{maintenant} \tag{5.4}$$

Avec l'équation 5.2 et l'équation 5.3 nous obtenons :

$$S_{reponse} = \frac{A'B * T_{reponse}}{T_{chemin}} \tag{5.5}$$

Ici comme AA' est orthogonal à AB, la zone de précision de A' est égale à AA'. Nous avons donc : $\sin \alpha = \frac{AA'}{A'B}$. Avec l'équation 5.5 et l'équation 5.1 nous obtenons finalement :

$$\begin{aligned} S_n =& \frac{A'B * T_{reponse}}{T_{chemin}} * \frac{AA'}{A'B} \\ =& \frac{AA' * T_{reponse}}{T_{chemin}} \end{aligned} \tag{5.6}$$

Nous voyons ainsi que S_n dépend de la zone de précision, du temps de réponse et

du temps de trajet. Nous pouvons dire que S_n est *proportionnel* à AA'. Dans la suite du document, nous appelons "proportion α" le rapport $\frac{S_n}{AA'} = \frac{T_{reponse}}{T_{chemin}}$.

5.6 Deuxième version : prise en compte des zones de sécurité

En plus des éléments présents dans le modèle $\mathcal{M}_{v1}$, le modèle $\mathcal{M}_{v2}[\mathcal{E}_{v2}, \mathcal{P}_{v2}]$ contient des états 3_k qui correspondent aux zones de sécurité obtenues après correction de la trajectoire. L'ensemble d'états est $\mathcal{E}_{v2} = \{0_1, 1_i, 2_j, 3_k\}$, avec 0_1 l'état initial. Les probabilités des transitions sont $\mathcal{P}_{v2} = \{P_{0i}, P_{1ij}\}$. Le modèle $\mathcal{M}_{v2}$ contient aussi un ensemble de variables $\{f, T_{chemin}, T_{total}, \alpha\}$, qui sont utilisées sur certaines transitions et initialisées au chargement du modèle. La signification des variables est donnée ci-dessous.

Le modèle est construit à partir de deux modules PRISM qui fonctionnent de la manière suivante :

Le module 1 (voir extrait dans le Listing 5.1) représente la mise à jour des coordonnées et des zones de précision calculées. Nous utilisons le module 2 (voir Listing 5.2) pour calculer dans quelle zone de sécurité le drone se trouvera après la correction effectuée suite à l'estimation de sa position. Ce calcul correspond à celui présenté en figure 5.6.

```
1  module Main
2
3          // 0 state 0
4          // 1 state 1i
5          // 2 state 2j
6          // 3 state proportion(alpha) (Tanswer/Tremain)
7          // 4 state final (safety zone)
8          s: [0..4] init 0;
9
10         // 0 null
11         // 1−5 zone 1−5
12         zone1: [0..5] init 0; // precision zone with sensors
13         zone2: [0..5] init 0; // precision zone after filter
14         zone3: [0..5] init 0; // safety zone
15
16
17         [ ] s=0 −> p01: (zone1'=1) & (s'=1)
18                        + p02: (zone1'=2) & (s'=1) + p03: (zone1'=3) & (s'=1)
19                        + p04: (zone1'=4) & (s'=1) + p05: (zone1'=5) & (s'=1);
```

```
20
21          [ ] s=1 & zone1=1 -> p111: (zone2'=1) & (s'=2)
22                              + p112: (zone2'=2) & (s'=2) + p113: (zone2'=3) & (s'=2)
23                              + p114: (zone2'=4) & (s'=2) + p115: (zone2'=5) & (s'=2);
24
25          [ ] s=1 & zone1=2 ->
26
27          ...
28
29          [ zp1begin ] s=2 & zone2=1 -> (s'=3);
30          [ zp2begin ] s=2 & zone2=2 -> (s'=3);
31          [ zp3begin ] s=2 & zone2=3 -> (s'=3);
32          [ zp4begin ] s=2 & zone2=4 -> (s'=3);
33          [ zp5begin ] s=2 & zone2=5 -> (s'=3);
34
35          [ zpend ] s=3 -> (s'=4) & (zone3'=sz);
36
37          ...
38  endmodule
```

Listing 5.1 – Extrait du module 1

Une fois que le module 1 a terminé l'estimation de la position (zone de précision), il appelle le module 2 avec les signaux `zp1begin`, `zp2begin`, `zp3begin`, `zp4begin` ou `zp5begin` afin de calculer la zone de sécurité obtenue. Le module 2 fait le calcul comme nous l'avons présenté dans la section 5.5. Le module 2 renvoie le résultat dans une variable globale `(sz)` commune aux deux modules et envoie le signal `zpend` pour que le module 1 reprenne la main. Comme nous pouvons le voir dans le Listing 5.2, la nouvelle position n'est pas calculée à partir des coordonnées estimées mais à partir de la limite supérieure de la zone de précision dans laquelle la position du drone est estimée. Ceci ajoute donc un peu de dérivation, notre position obtenue est donc supérieure ou égale à la position réelle du drone.

Les variables $\{f, T_{chemin}, T_{total}, \alpha\}$ que contient notre modèle sont un peu différentes dans notre codage en PRISM. La fréquence du filtre f correspond à `1/Tanswer`, la variable `Tanswer` correspondant à $T_{reponse}$ dans notre équation 5.2. Le temps restant pour arriver au prochain point T_{chemin} correspond à `Tremain` dans le Listing 5.2. Le temps total T_{total} entre les deux points correspond à `Tpath`. Notons que la proportion α n'apparait pas directement dans le code PRISM du Listing 5.2. En effet, comme cette proportion doit être calculée à nouveau à chaque boucle, nous utilisons directement l'expression $\frac{\texttt{Tanswer}}{\texttt{Tremain}}$.

```
module CalculePrecision

        // state calcule precision
        // 0 begin
        // 1 calcule
        scp: [0..2] init 0;

        // time remain for this path
        Tremain: [0..Tpath] init Tpath;
        distanceArrive: [0..100] init 0;

        [zp1begin] scp=0 & (floor (zp1*Tanswer/Tremain) < 100)
                        -> (distanceArrive' = floor(zp1*Tanswer/Tremain)) & (scp'=1);
        [zp2begin] scp=0 & (floor (zp2*Tanswer/Tremain) < 100)
                        -> (distanceArrive' = floor(zp2*Tanswer/Tremain)) & (scp'=1);
        [zp3begin] scp=0 & (floor (zp3*Tanswer/Tremain) < 100)
                        -> (distanceArrive' = floor(zp3*Tanswer/Tremain)) & (scp'=1);
        [zp4begin] scp=0 & (floor (zp4*Tanswer/Tremain) < 100)
                        -> (distanceArrive' = floor(zp4*Tanswer/Tremain)) & (scp'=1);
        [zp5begin] scp=0 & (floor (zp5*Tanswer/Tremain) < 100)
                        -> (distanceArrive' = floor(zp5*Tanswer/Tremain)) & (scp'=1);

        [ ] scp=1 & distanceArrive <=zp1 -> (sz'=1) & (scp'=2);
        [ ] scp=1 & distanceArrive >zp1 & distanceArrive <= zp2 -> (sz'=2) & (scp'=2);
        [ ] scp=1 & distanceArrive >zp2 & distanceArrive <= zp3 -> (sz'=3) & (scp'=2);
        [ ] scp=1 & distanceArrive >zp3 & distanceArrive <= zp4 -> (sz'=4) & (scp'=2);
        [ ] scp=1 & distanceArrive >zp4 & distanceArrive <= zp5 -> (sz'=5) & (scp'=2);

        [zpend] (scp=2) & (Tremain-Tanswer>0)
                        -> (scp'=0) & (Tremain'= Tremain-Tanswer);

endmodule
```

Listing 5.2 – Module 2

Ce module met alors à jour la position du drone et donne à nouveau la main au module 1 pour le prochain pas de calculs. La communication entre ces modules est assurée par les signaux `zp1begin`, `zp2begin`, `zp3begin`, `zp4begin`, `zp5begin` et `zpend`.

Notons que les variables T_{chemin} et f ne sont pas des paramètres au même sens que ceux des pMC, mais plutôt des constantes que l'on doit fixer pour pouvoir exécuter le modèle.

Le modèle ne change pas qu'en fonction de la proportion α, mais aussi en fonction des distances choisies pour les zones de précision et de sécurité et de la longueur du trajet. Nous avons construit un modèle avec 106 états et 130 transitions avec PRISM, lorsque $T_{chemin} = 1$. Lorsque $T_{chemin} = 2$, nous obtenons 456 états et 680 transitions.

Ce modèle (version 2) comporte encore plusieurs limitations. Limitation L2.1 : la position précédente du drone n'est pas prise en compte dans les nouveaux calculs ; limitation L2.2 : la distance maximale de la zone de précision est utilisée pour le calcul à la place de la distance réelle.

Avant de continuer d'améliorer notre modèle, nous allons maintenant essayer d'obtenir les probabilités pour les transitions (P_{0i} et P_{1ij}) correspondant aux capteurs et filtre de la Figure 5.3. Pour cela nous avons fait des expérimentations sur une chaîne de production pour compléter les probabilités sur les transitions des versions 1 et 2.

5.7 Expérimentations sur une chaîne de production

La principale méthode de recherche pour connaître la position réelle du drone utilise beaucoup de caméras. Sur un grand parcours, cette méthode devient complexe et coûteuse. Nous avons donc essayé une autre méthode : utiliser les possibilités de déplacement sur une chaîne de production.

FIGURE 5.7 – Chaîne de production.

Nous avons utilisé une chaîne de montage [51] (voir Figure 5.7) de l'IUT de Nantes. Cette chaîne n'est pas en l'extérieur mais à l'intérieur d'une grande salle (atelier).

FIGURE 5.8 – Boîte contenant le contrôleur de vol et les capteurs.

Voyons ce que cette chaîne peut faire et comment elle fonctionne. Dans les usines qui fabriquent des produits, par exemple, les chaînes de production servent à guider le produit à travers plusieurs stations. Par exemple, dans une fabrique de beignets, un beignet est mis sur la chaîne pour être conduit à chaque station. Une station peut être consacrée à la cuisine, une autre peut rassembler les beignets dans une boîte et enfin la dernière station met un autocollant sur la boîte. La chaîne de production que nous avons utilisée compte 5 stations et la chaîne peut être déviée à certains endroits.

Dans cette expérience, nous avons le contrôle sur la chaîne. Il est possible de

choisir le chemin (éviter certaines stations) et d'arrêter la chaîne à un instant donné.

Plutôt que de mettre tout le drone sur la chaîne de production, nous avons uniquement mis les capteurs et le contrôleur de vol dans une boîte. Les capteurs sont fixés à cette boîte comme sur la Figure 5.8. La chaîne de production déplace la boîte en suivant le chemin programmé.

Nous n'avons pas utilisé tous les chemins de cette chaîne de production. Les chemins que nous avons utilisés pour l'expérimentation sont représentés dans la Figure 5.9.

La boîte contenant le contrôleur de vol et les capteurs part du point bleu puis suit la ligne jaune. Au premier point noir (`wait 1`), la boîte tourne à gauche pour atteindre la ligne verte. Les points noirs (`wait 1` à `wait 12`) représentent les endroits où plusieurs chemins sont accessibles. À certains endroits nous choisissons de faire un arrêt comme `stop1`, `stop2`, `stop3` et `stop4`. À d'autres endroits nous choisissons de faire une ligne droite comme en bas à gauche de la Figure 5.9. La chaîne fonctionne de la façon suivante. Le processus pour choisir un chemin prend moins d'une seconde. Une fois à l'arrêt 1 (`stop1`), la boîte est stoppée pendant 5 secondes pour les tours impairs et 10 secondes pour les tours pairs. Le but de ces arrêts est d'étudier le comportement du filtre lors de changements brutaux de trajectoire. La boîte passe ensuite par les arrêts `stop2`, `stop3` et `stop4` où le temps d'attente est d'une seconde.

La longueur du trajet est de 44,5 m

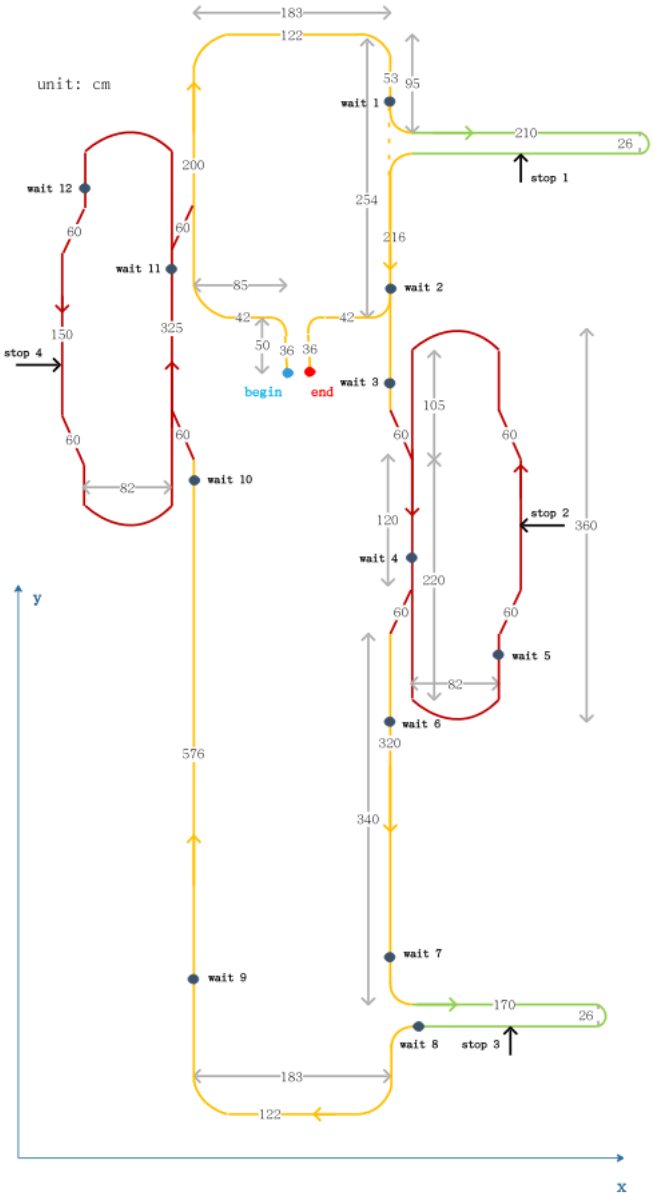


FIGURE 5.9 – Chaîne de production.

par tour ; nous faisons 3 tours soit 136,5 m pour les 3 tours. En ajoutant le chemin pour sortir du point `begin` et revenir au point `end`, la longueur totale est de 144,25 m.

L'idée de notre expérimentation est d'enregistrer les positions détectées par les capteurs pour obtenir les probabilités de bruit des capteurs correspondant aux transitions bleues sur la Figure 5.3 et les passer dans différents types de filtres. En comparant le résultat de chaque filtre à la position réelle, nous pourrons identifier le meilleur filtre pour ce scénario. Pour connaître la position réelle de la boîte, nous mesurons la longueur de la ligne d'assemblage (figure 5.9).

L'environnement de l'expérimentation n'a pas permis d'ajouter le GPS dans la boîte. En effet, le GPS n'est pas efficace en intérieur ni assez précis pour une chaîne aussi réduite. Nous n'avons donc pas la position GPS (x, y, z) de la boîte.

Cependant, il est possible de calculer cette position avec les données de l'IMU. Prenons par exemple l'accéléromètre ; soit S_x la longueur du déplacement par rapport à l'axe des x. Soit V_{0x} la vitesse initiale, a_x l'accélération et t le temps du déplacement. Nous avons $S_x = V_{0x}t + (a_x t^2)/2$.

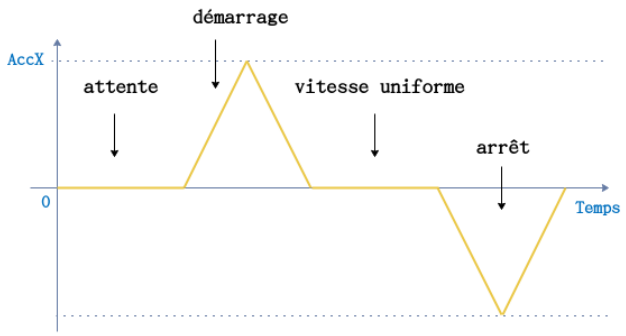


FIGURE 5.10 – Comportement théorique de l'accéléromètre.

Le comportement théorique de l'accéléromètre de l'axe x est présenté sur la Figure 5.10. Lorsque la courbe augmente fortement au dessus de zéro, la boîte se déplace jusqu'à atteindre une vitesse uniforme. Nous devons pouvoir observer des pics de valeurs positives au moment du départ.

Un pic de valeur négative signifie un arrêt sur la chaîne de production. Nous pouvons constater un temps à zéro entre les pics de valeurs positives et négatives, cela signifie que la boîte se déplace à une vitesse constante. Dans notre test (figure 5.9) le plan horizontal est l'axe x et le plan vertical est l'axe y.

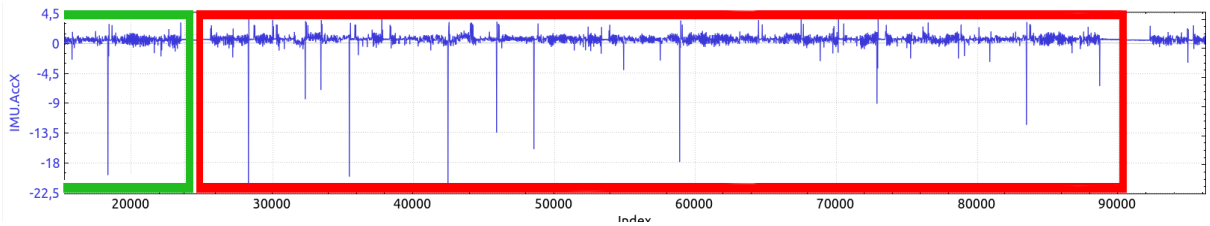


FIGURE 5.11 – Comportement observé de l'accéléromètre.

Comme présenté plus haut, nous avons fait trois tours pour chaque test. Nous étudions les tests en 4 parties :

1. un parcours de *begin* à *stop1*,
2. un tour complet de *stop1* à *stop1*,
3. un deuxième tour complet de *stop1* à *stop1*,
4. un parcours de *stop1* à *end*.

La 3ème partie est représentée par un cadre rouge dans la Figure 5.11. Nous pouvons voir une grande différence entre la fin de la 3ème partie (rouge) et la fin de la 2ème partie (vert). Nous observons que la précision de l'IMU n'est pas aussi bonne que celle utilisée réellement par PIXIEL pour les vols dans le parc du Puy du Fou. Il nous est en effet difficile d'observer dans la Figure 5.11 les motifs que nous avons identifiés dans la Figure 5.10. Cela peut également être dû aux conditions du test qui s'est déroulé en intérieur. La salle de la chaîne de production est connectée à d'autres salles, ce qui entraîne beaucoup d'ouvertures et fermetures de portes qui peuvent perturber l'accéléromètre. Également, la longueur du parcours sur la chaîne de production est plus petite que celle des parcours lors des spectacles de drones. Ces perturbations ne sont pas celles que nous voulons détecter. Nous n'avons donc pas pu utiliser nos mesures pour obtenir des informations plus précises sur les capteurs et le filtre. Malgré tout, cette expérience nous a permis d'approfondir notre analyse du fonctionnement du drone et d'écarter certaines hypothèses.

Pour corriger cette limitation notre solution est d'ajouter des paramètres dans notre modèle. En effet les éléments non pris en compte dans l'expérience peuvent être traités comme des paramètres du modèle.

Un autre limitation est que nous ne pouvons pas être sûrs que les probabilités du filtre soient toujours constantes. Elles pourraient avoir un lien avec la précision des capteurs. Pour cette raison, nous ne pouvons pas utiliser un paramètre constant sur la transition correspondant au filtre. En revanche, la probabilité d'arriver dans les `zone1` à

`zone5` est à peu près constante. Il s'agit du reste du bruit présenté dans la Figure 5.4c. Nous ajouterons donc ce seul paramètre à la place des probabilités P_{0i} et P_{1ij} dans la Figure 5.3.

5.8 Troisième version : prise en compte de la distance exacte

La version 3 de notre modèle corrige les limitations L2.1 (non prise en compte de la position précédente) et L2.2 (distance maximale utilisée) que nous avons présentées à la fin de la version 2. Nous avons ajouté deux valeurs *startPointX* et *startPointY* pour représenter les coordonnées du point de départ. Nous avons également ajouté un module *loop* pour répéter plusieurs fois les estimations de la position.

5.8.1 Le modèle construit pour PRISM

Le modèle $\mathcal{M}_{v3Prism}[\mathcal{E}_{v3Prism}, \mathcal{P}_{v3Prism}]$ de la version 3 est encodé dans le langage d'entée de PRISM similairement au modèle $\mathcal{M}_{v2}$ présenté en section 5.6. L'ensemble d'états est $\mathcal{E}_{v3Prism} = \{0_1, 2_j, 3_k\}$, où 0_1 est l'état initial. L'ensemble des probabilités change en $\mathcal{P}_{v3Prism} = \{P_{0j}\}$ où P_{0j} représente une abstraction unique des probabilités P_{0i} et P_{1ij} présentes dans la version 2 (comme expliqué en fin de section 5.7). C'est pour cela que ce modèle n'a plus les états 1_i. Dans ce nouveau modèle, des modules similaires sont utilisés mais deux variables supplémentaires sont ajoutées : `startPointX` et `startPointY`. Ces variables permettent de définir les coordonnées du point de départ, et de les mettre à jour pour boucler.

```
1  global startPointY:[0..100] init 0;
2  ...
3
4  [zp1begin] scp=0 & (floor((−zp1+startPointY)*Tanswer/Tremain)>−100)
5  & (floor((zp1+startPointY)*Tanswer/Tremain)<100)
6  −> (distanceArrive' = floor((zp1+startPointY)*Tanswer/Tremain)) & (scp'=1);
7  ...
8
9  [ ] scp=1 & distanceArrive+startPointY <= zp1
10 −> (sz'=1) & (scp'=2) & (startPointY'= distanceArrive+startPointY);
```

Listing 5.3 – Changement de point.

Par exemple, la ligne 12-13 du Listing 5.1 est modifiée par l'expression donnée dans la ligne 4-5 du Listing 5.3 afin de prendre en compte la position réelle plutôt que la position extrémale de la zone. Les lignes 10-11 présentent la prise en compte de la position précédente.

```
1   [ ]s=4 & Tremain > Tanswer −> (s'=0) & (Tremain'=Tremain−Tanswer);
```

Listing 5.4 – Modification du modèle pour réaliser les boucles.

Comme présenté dans le Listing 5.4, nous relançons les calculs lorsque la durée restante est positive.

Avec la version 2, dès que `Time=2`, l'analyse du modèle nécessite quatre fois plus de calculs. En ajoutant les points de départ et d'arrivée, nous avons attendu 2h sans obtenir de résultat, même en testant une propriété qui vérifie si le modèle quitte l'état initial pour aller au deuxième état, ce qui doit arriver avec une probabilité de 100%.

PRISM ne prend pas en compte les paramètres. Nous avons alors expérimenté le modèle avec PARAM (voir section 4.5.2). De manière générale, nous testons ici avec `Time=5` donc avec 5 estimations de position. Nous avons transformé les probabilités des transitions en paramètres pour tester le modèle encodé en PARAM, mais cela n'a pas abouti. L'outil PARAM n'est pas arrivé à analyser notre modèle car il est limité par la taille du modèle ; une méthode plus rapide est donc nécessaire.

5.8.2 Le modèle construit en Python pour utiliser la méthode pSMC

Face à ces difficultés, nous avons décidé d'utiliser la méthode statistique pSMC qui devrait moins souffrir de la taille de notre modèle. Nous avons ainsi développé la version 3 de notre modèle en Python dans le but d'utiliser la méthode pSMC et notre outil MCpMC. Le modèle $\mathcal{M}_{v3Python}[\mathcal{E}_{v3Python}, \mathcal{P}_{v3Python}]$ de la version 3 est encodé en Python similairement au modèle $\mathcal{M}_{v3Prism}$ présenté en section 5.8.1.

Dans ce nouveau modèle, l'ensemble d'états change en $\mathcal{E}_{v3Python} = \{0_1, 3_k\}$, l'état initial est toujours 0_1. L'ensemble de probabilité $\mathcal{P}_{v3Python}$ contient les paramètres `ProbaFilter1` à `ProbaFilter5` (qui correspondent à la combinaison des probabilités P_{0i} et P_{0ij} des modèles précédents). Les variables sont α, `period`, `T`, `startPoint`, `DistanceZoneSecurity0`, `y`(position estimée sur axe y). `ProbaFilter1` à `ProbaFilter5` sont utilisés comme paramètres du polynôme de sortie. Nous commençons avec un *single thread* afin de bien contrôler l'évolution du modèle, puis nous ajoutons du *multi-*

threading pour accélérer la vitesse de calcul.

Nous introduisons `DistanceZoneSecurity0` comme la distance qui sépare la position du drone de la zone de sécurité 4. Cela nous permettra de vérifier si le drone est hors des zones de sécurité 1 à 3. Nous introduisons comme paramètres les probabilités que le filtre estime la position du drone dans l'une des zone de sécurité, soit : `ProbaFilter1`, `ProbaFilter2`, `ProbaFilter3`, `ProbaFilter4` et `ProbaFilter5`. La variable `period` représente la période entre deux exécutions des capteurs et du filtre (1/fréquence). La variable `Time` représente le temps du trajet (T_{chemin}).

La construction du modèle se fait de la façon suivante. Nous prenons avec le code Python une valeur aléatoire `y = random.choice(range(-100, 100))`, y est donc dans l'intervalle $[-99, 100]$ comme valeur estimée pour le capteur après corrections du filtre. Nous pouvons aussi changer en `y = random.choice(range(-101, 100))` pour l'intervalle $[-100, 100]$. Plus précisément nous paramétrons la façon de tirer les valeurs de y :

- nous avons une probabilité `ProbaFilter1` pour y $\in [-20, 20]$,
- nous avons une probabilité `ProbaFilter2` pour y $\in (20, 40] \cup [-40, -20)$,
- nous avons une probabilité `ProbaFilter3` pour y $\in (40, 60] \cup [-60, -40)$,
- nous avons une probabilité `ProbaFilter4` pour y $\in (60, 80] \cup [-80, -60)$,
- nous avons une probabilité `ProbaFilter5` pour y $\in (80, 100] \cup [-99, -80)$.

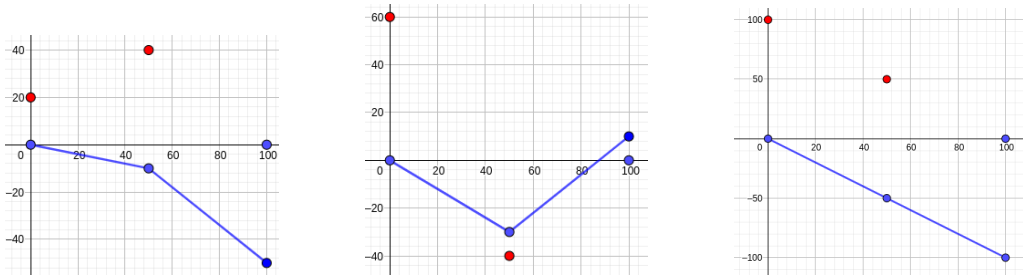
Dans la simulation, nous compensons `ProbaFilter`i en le divisant par la probabilité d'avoir un y au hasard (uniforme), ce qui permettra, à la fin de la simulation, d'évaluer la fonction de récompense r' présentée dans la section 4.4 ; cela donne :

- pour y $\in [-20, 20]$: `ProbaFilter1/(41/200)`,
- pour y $\in (20, 40] \cup [-40, -20)$: `ProbaFilter2/(40/200)`,
- pour y $\in (40, 60] \cup [-60, -40)$: `ProbaFilter3/(40/200)`,
- pour y $\in (60, 80] \cup [-80, -60)$: `ProbaFilter4/(40/200)`,
- pour y $\in (80, 100] \cup [-99, -80)$: `ProbaFilter5/(39/200)`.

Une fois cette étape effectuée, nous ajoutons la position précédente (`startPoint`) pour obtenir le position estimée. Nous calculons la nouvelle valeur de y en utilisant la proportion α définie en section 5.5. Le nouveau y sera la nouvelle valeur de `startPoint` pour la prochaine boucle. Ensuite, on boucle jusqu'à la fin du trajet ou jusqu'à ce que

le drone pénètre dans la zone interdite.

Par exemple, nous effectuons 3 simulations avec `T=2` et `periode=1` (i.e. $T_{reponse} = 1$) pour tester la propriété suivante : "le drone a t-il dépassé la distance de sécurité (`DistanceZoneSecurity0`) 100 ?"



(a) (b) (c)

FIGURE 5.12 – Simulations du modèle $\mathcal{M}_{v3Python}$.

La Figure 5.12 présente les 3 simulations, les points rouges sont les positions estimées et les points bleus correspondent au trajet réellement obtenu.

TABLE 5.1 – Valeurs de la simulation 1.

T	`startPoint`	Tirage	Position estimée	Nouvelle position (y)	**récompense**
1	0	20	20	-10	`ProbaFilter1/(41/200)`
2	-10	50	40	-50	`ProbaFilter3/(40/200)`

Pour la 1ère simulation quand `T=1` (voir Table 5.1 ligne T=1), `startPoint=0`, nous tirons la valeur aléatoire 20. Après le calcul de la proportion α, nous obtenons la nouvelle coordonnée y=-10. La récompense obtenue est exprimée par le monôme `ProbaFilter1/(41/200)`. Étant donné que le trajet n'est pas terminé et que |y| < `DistanceZoneSecurity0`, nous continuons la simulation.

Ensuite, pour `T=2` (voir Table 5.1 ligne T=2), nous prenons pour `startPoint` la valeur précédente de y. Ensuite nous tirons la valeur aléatoire 50, la position estimée est donc $-10 + 50 = 40$. Après le calcul de la proportion α, nous obtenons la nouvelle coordonnée y=-50. La récompense est cette fois exprimée par le monôme `ProbaFilter3/(40/200)`.

Le trajet est fini, mais |y| < `DistanceZoneSecurity0`, donc la simulation ne satisfait pas la propriété définie plus haut. La récompense finale est donc `ProbaFilter1/(41/200) * ProbaFilter3/(40/200)`, mais elle ne sera pas prise en compte puisque la propriété n'est pas satisfaite.

TABLE 5.2 – Valeurs de la simulation 2.

T	`startPoint`	Tirage	Position estimée	Nouvelle position (y)	**récompense**
1	0	60	60	-30	`ProbaFilter3/(40/200)`
2	-30	-10	-40	10	`ProbaFilter1/(41/200)`

La 2ème simulation (voir la Table 5.2) ne satisfait pas non plus la propriété donnée, la récompense associée est `ProbaFilter3/(40/200) * ProbaFilter1/(41/200)`.

TABLE 5.3 – Valeurs de la simulation 3.

T	`startPoint`	Tirage	Position estimée	Nouvelle position (y)	**récompense**
1	0	100	100	-50	`ProbaFilter5/(39/200)`
2	-50	100	50	-100	`ProbaFilter5/(39/200)`

Pour finir, la 3ème simulation (voir Table 5.3) satisfait la propriété et sa récompense associée est `ProbaFilter5/(39/200)*ProbaFilter5/(39/200)`.

À la fin des simulations, il n'y a que la troisième simulation qui satisfait notre propriété (|y| $\geq$ `DistanceZoneSecurity0`). Grâce à pSMC, la probabilité de satisfaire la propriété est estimée par le polynôme :

$$\frac{1}{3} * \frac{\texttt{ProbaFilter5}}{39/200} * \frac{\texttt{ProbaFilter5}}{39/200} \approx 8.77 * \texttt{ProbaFilter5}^2.$$

Inversement, la probabilité de ne pas satisfaire la propriété est estimée à :

$$\frac{1}{3} * \frac{\texttt{ProbaFilter3}}{40/200} * \frac{\texttt{ProbaFilter1}}{41/200} + \frac{1}{3} * \frac{\texttt{ProbaFilter3}}{40/200} * \frac{\texttt{ProbaFilter1}}{41/200}$$

$$\approx 16.26 * \texttt{ProbaFilter3} * \texttt{ProbaFilter1}.$$

Remarque : A première vue, les valeurs présentées ci-dessus peuvent sembler aberrantes (par exemple leur somme pourrait être supérieure à 1). Ceci est du au fait que le nombre d'échantillons utilisé ici, pour l'exemple, est très faible. En effet, les résultats garantis par la loi des grands nombres et le théorème central limite ne sont valides que pour n suffisamment grand ($n \to \infty$), ce qui n'est pas le cas dans notre exemple.

Étude de la correction de notre modèle. Nous voulons nous assurer que notre modèle et notre encodage comme entrée de MCpMC sont corrects. Les résultats ne sont en effet pas toujours très faciles à vérifier. Pour les trajets de T=2 nous avons vérifié à la main que les traces de notre modèle étaient correctes. Après plusieurs changements, nous avons estimé nécessaire le développement d'une vérification plutôt visuelle que manuelle.

Notre modèle est correct si les calculs effectués au cours des différentes transition (probabilisées) donnent des valeurs cohérentes et qui ont pour conséquence de prendre une décision cohérente par rapport à la trajectoire. Pour s'assurer que le modèle est correct nous devons donc établir la cohérence des valeurs calculées en fonction des données reçues en entrée.

Nous avons choisi de vérifier cette cohérence en utilisant GeoGebra sur la base du modèle abstrait mathématique.

GeoGebra

GeoGebra [52] est un logiciel de géométrie dynamique en 2D/3D, c'est-à-dire qu'il permet de manipuler des objets géométriques (cercle, droite et angle, par exemple) et de voir immédiatement le résultat. Il est de plus équipé d'un ensemble de fonctions algébriques.

GeoGebra calcule les tracés exacts en fonction des équations fournies en entrée. Nous avons alors cherché à comparer les résultats fournis par GeoGebra pour les même équations que celles utilisées dans notre modèle. Si les résultats ne correspondent pas, ils sont incorrects et nous devons revoir le modèle.

Une phase importante de cette étude de correction consiste alors à intégrer toutes nos équations dans GeoGebra. Lorsque notre modèle est correct, nous poursuivons

avec son encodage dans MCpMC (voir section 4.5.3) pour obtenir le polynôme correspondant à l'estimation de la probabilité de satisfaire la propriété.

Nous avons pour $i \in [1, 5]$ un paramètre `ProbaFiltre`i qui correspond à la probabilité d'arriver dans chaque zone de précision $[0dm, 20dm]$, $(20dm, 40dm]$, $(40dm, 60dm]$, $(60dm, 80dm]$ et $(80dm, 100dm]$ où dm correspond au décimètre. En effet, il est plus facile dans notre outil d'utiliser des décimètres entiers plutôt que des mètres qui nous obligeraient à utiliser des réels.

Considérons un exemple avec la Figure 5.13. Le trajet va de `PointBegin` = $(0, 0)$ à `PointEnd` = $(200, 0)$ et la zone de sécurité est entre $(-60, 60)$. Soit `LC` (LC_x, LC_y), la position réelle du drone. Au début, nous avons `LC=PointBegin=(0,0)`. Notre temps pour ce trajet est `T=4` secondes, et la fréquence est `f=1`, soit un calcul une fois par seconde. Le temps initial ($T_{maintenant}$) est `times=0`.

Avant d'arriver au point `PointEnd`, le drone devrait estimer 4 fois sa position.

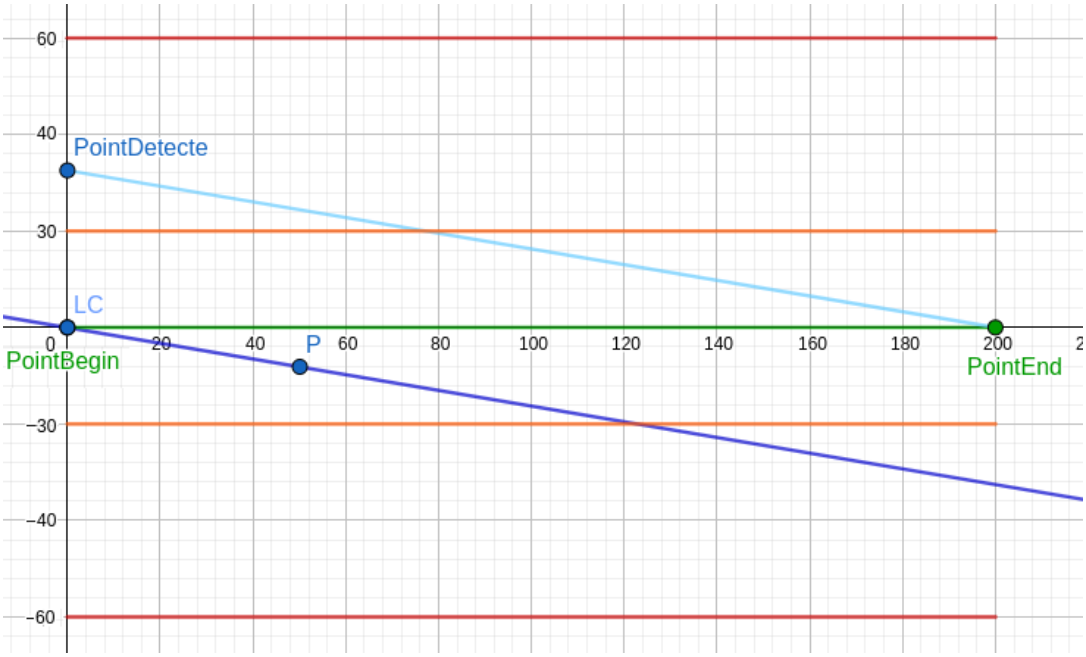


FIGURE 5.13 – Départ du vol.

`PointDetecte` correspond à la position estimée par les capteurs et corrigée par le filtre, qui prend en compte le bruit restant que nous avons vu précédemment dans la Figure 5.4c. Nous tirons au hasard un chiffre entre [-99, 100] pour la coordonnée (y) de `pointDetecte`. Avec les coordonnés $(0, PD_y)$ de ce point, nous utilisons la méthode vue précédemment (section 5.5) pour obtenir la position suivante du drone `P` = (P_x, P_y).

Cela revient à faire comme suit :

Nous traçons une ligne entre `PointDetecte` et `PointEnd (PD,PE)`: $y = k_{PDPE}x + b_{PDPE}$

$$\begin{cases} k_{PDPE} = \frac{PD_y - PE_y}{PD_x - PE_x} \\ b_{PDPE} = PE_y - k_{PDPE} * (PE_x) \end{cases}$$

Nous traçons ensuite une ligne parallèle à la ligne (`PD,PE`) qui passe par le point `LC`. Sa pente $k_{LCP} = k_{PDPE}$ est donc identique à celle de (`PD,PE`), avec la position de `LC` = (LC_x, LC_y) :

$$\begin{cases} k_{LCP} = k_{PDPE} \\ b_{LCP} = LC_y - k_{LCP} * (LC_x) \end{cases}$$

Comme nous ne prenons pas en compte le bruit pour l'axe x dans cette version, nous avons $P_x = (\texttt{times} + 1) * (\texttt{f/T}) * (\texttt{PE}_x - \texttt{PB}_x) = 50$ (voir Figure 5.13). Ainsi,

$$\begin{aligned} P_y &= k_{LCP} * P_x + LC_y - k_{LCP} * LC_x \\ &= \frac{PD_y - PE_y}{PD_x - PE_x} * P_x + LC_y - \frac{PD_y - PE_y}{PD_x - PE_x} * LC_x \end{aligned}$$

Dans notre exemple P n'est pas hors de la zone de sécurité, le drone continue donc son parcours. Si $P_y \in [60, +\infty) \cup (-\infty, -60]$, le trajet s'arrête immédiatement. Le calcul des probabilités donne la probabilité pour que le drone soit en dehors de la zone de sécurité.

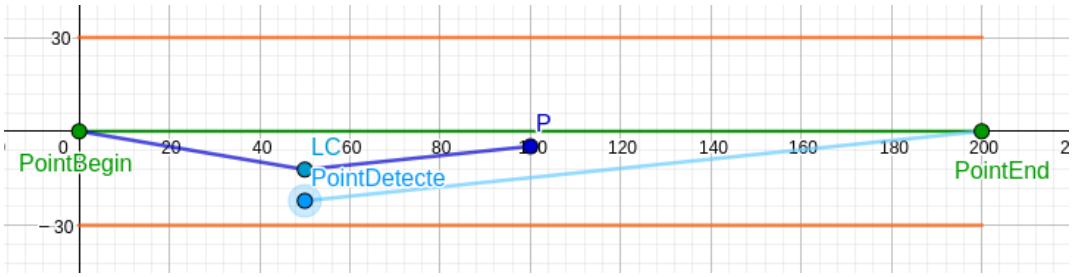


FIGURE 5.14 – Première correction de la trajectoire.

Le trajet continue comme dans la Figure 5.14, le point LC est maintenant égal à l'ancienne position P du drone. Avec un nouveau `PointDetecte`, nous avons la nouvelle position estimée du drone.

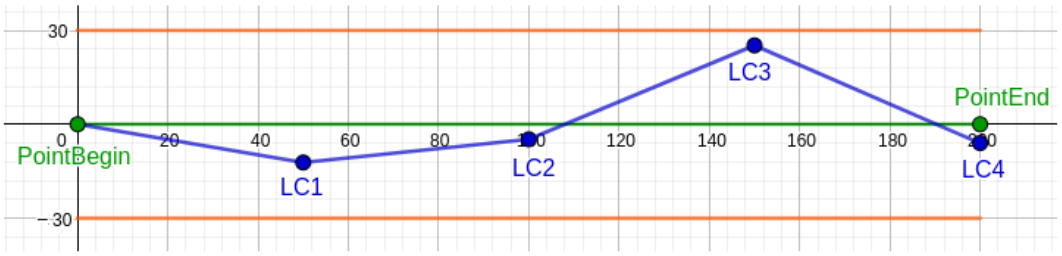


FIGURE 5.15 – Trajectoire complète.

À la fin, nous obtenons un trajet similaire à celui présenté dans la Figure 5.15. Nous utilisons finalement la méthode présentée dans la section 4.4 pour obtenir un polynôme final.

La version 3 de notre modèle a corrigé les limitations présentes dans la version 2 (L2.1 et L2.2) mais avec le modèle obtenu, nous ne considérons des perturbations que sur un seul axe ; ce sera la limitation L3 que nous aborderons dans la suite.

En effet, pour un trajet de A à B comme dans la Figure 5.6, le problème de position existe sur les deux axes x et y. Contrairement à l'axe y, qui correspond au problème des zones que nous avons présenté dans la section 5.5. L'axe x reflète le problème d'un drone en retard ou en avance sur le temps de son trajet. Normalement si un drone est en avance, il va juste attendre le temps prévu avant d'aller au prochain objectif. En revanche, si un drone est en retard, il peut ignorer un objectif intermédiaire pour aller directement à l'objectif suivant. La probabilité que l'objectif change et que le drone entre dans une zone de danger en coupant la trajectoire doit donc être ajoutée à la probabilité d'atteindre la zone de danger à cause des perturbations sur l'axe y.

5.9 Versions 4 à 7 avec la méthode pSMC

Le modèle $\mathcal{M}_{v4}[\mathcal{E}_{v4}, \mathcal{P}_{v4}]$ de la version 4 est encodé en Python similairement au modèle $\mathcal{M}_{v3Python}$ présenté en section 5.8.2. Dans ce nouveau modèle nous avons ajouté deux variables supplémentaires : `x` et `LengthX` (distance entre le dernier point de passage et le prochain).

Comme ce que nous avons fait pour la coordonnée (y), nous tirons au hasard dans l'intervalle [-99, 100] un chiffre pour la coordonnée (x) de `pointDetecte` (voir Figure 5.13). Nous pouvons aussi changer en `x = random.choice(range(-101, 100))`

pour l'intervalle $[-100, 100]$. Nous utilisons alors le calcul présenté dans la section 5.5. Les cas d'un drone en avance ou en retard par rapport à son temps de trajet sont également pris en compte mais sans changement d'objectif. Le modèle de la version 5 $\mathcal{M}_{v5}[\mathcal{E}_{v5}, \mathcal{P}_{v5}]$ est similaire au modèle $\mathcal{M}_{v4}$ ci-dessus, auquel nous avons ajouté des variables correspondant aux coordonnées permettant de calculer le changement d'objectif. Ce changement d'objectif a lieu lorsque le temps prévu pour une partie du trajet (entre deux points intermédiaires) est écoulé, même si le drone n'a pas encore atteint son objectif (intermédiaire).

En cas de retard, nous vérifions si le drone va sortir de la zone de sécurité en changeant d'objectif. Pour ce faire, nous vérifions si la trajectoire effectuée en changeant d'objectif passe hors de la zone de sécurité. Nous présentons notre méthode de développement avec GeoGebra [1] : l'importance est de trouver le point P (intersection de deux lignes de la zone de sécurité 4) et le point T (intersection du chemin raccourci pour rattraper le retard avec la nouvelle ligne de la zone de sécurité 4) (voir Figure 5.16). Si la coordonnée y de T est plus grande que la coordonnée y de P, alors le drone va traverser la zone de sécurité 4 ou 5 en coupant la trajectoire.

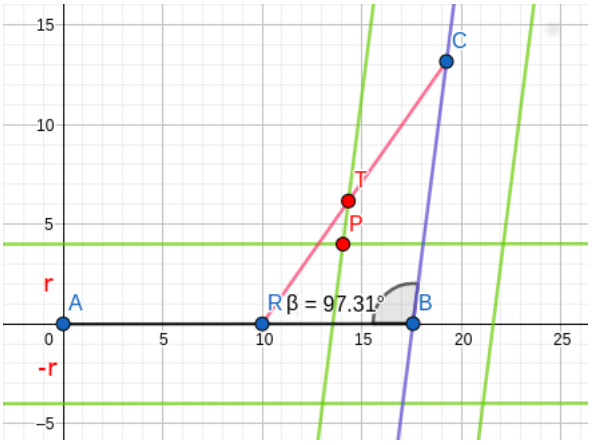


FIGURE 5.16 – Cas de changement de trajectoire à cause d'un retard.

Nous montrons ce cas avec l'outil *GeoGebra* pour comprendre visuellement ce problème. Considérons un trajet A-B-C comme dans la Figure 5.16. Soit $A = (A_x, A_y)$, $B = (B_x, B_y)$, $C = (C_x, C_y)$, la ligne AB : $y = k_{AB}x + b_{AB}$ et la ligne BC : $y = k_{BC}x + b_{BC}$. Les lignes vertes représentent les limites de la zone de sécurité. Avec les points B et C, nous avons la ligne AB : $y = k_{AB}x + b_{AB}$ et la ligne $BC : y = k_{BC}x + b_{BC}$ avec

1. Nous présentons uniquement l'un des quatre cas de figure possibles (en fonction de l'angle β représenté dans la Figure 5.16). Les autres s'obtiennent similairement.

$$\begin{cases} k_{AB} = \frac{A_y - B_y}{A_x - B_x} \\ b_{AB} = B_y - k_{AB} * B_x \end{cases} \qquad \begin{cases} k_{BC} = \frac{C_y - B_y}{C_x - B_x} \\ b_{BC} = B_y - k_{BC} * B_x \end{cases}$$

La valeur r (en rouge dans la Figure 5.16) permet de changer la taille de la zone de sécurité avec $r \geq 0$. Les lignes $y = k_{AB}x + b_{AB} \pm r$ représentent les limites de la zone de sécurité autour de la ligne AB. Les lignes $y = k_{BC}x + b_{BC} \pm r/\cos\beta$ représentent les limites de la zone de sécurité autour de la ligne BC, l'angle β représente l'angle entre le point A, le point B et le point C. Le point R est le point de retard, obtenu à cause des bruits des capteurs. En théorie, le point R pourrait ne pas se trouver sur la ligne AB. Nous supposons ici qu'il s'y trouve, pour l'exemple, mais notre modèle reflète le cas général. À cause du retard, le drone va oublier son objectif B, et voler directement vers l'objectif C. Ainsi, le drone va suivre la ligne rouge $RC : y = k_{RC}x + b_{RC}$. Avec les points R et C, nous pouvons avoir :

$$\begin{cases} k_{RC} = \dfrac{R_y - C_y}{R_x - C_x} \\ b_{RC} = C_y - k_{RC} * C_x \end{cases}$$

Nous avons besoin de savoir si le drone peut sortir de la zone de sécurité en suivant le trajet RC. Cela dépend de la distance entre le point B et le point C, de l'angle β, de la distance entre le point R et le point C et de la taille de la zone de sécurité r. Le point $T = (T_x, T_y)$ est l'intersection des lignes RC $y = k_{RC}x + b_{RC}$ et de la limite supérieure de la zone de sécurité entourant BC $y = k_{BC}x + b_{BC} - r/\cos\beta$ car $r \geq 0$ et $\cos\beta < 0$ dans notre exemple. Le point $P = (P_x, P_y)$ est l'intersection des limites supérieures des zones de sécurité entourant AB et BC $y = k_{AB}x + b_{AB} + r$ et $y = k_{BC}x + b_{BC} - r/\cos\beta$. En comparant les coordonnées du point T et du point P, nous pouvons conclure que le drone sort de la zone de sécurité si $T_y > P_y$, c'est-à-dire si :

$$C_y + \frac{-\frac{r}{\cos\beta} * k_{RC}}{k_{RC} - k_{BC}} > B_y + r, \ i.e.$$

$$C_y + \frac{-\frac{r}{\cos\beta}(C_x - B_x)(R_y - C_y)}{R_y(C_x - B_x) + C_y(B_x - R_x) + B_y(R_x - C_x)} > B_y + r.$$

Certains retards du drone peuvent aussi être rattrapés en augmentant sa vitesse

de déplacement (lorsque le temps n'est pas écoulé), mais cette vitesse ne peut pas dépasser une vitesse maximale.

Le drone est rarement en retard dans un environnement idéal mais cela devient beaucoup plus fréquent s'il y a du vent. En testant la version 5 de notre modèle, nous n'avons pas observé de changements importants dans les probabilités obtenues, car le retard ne s'est pas produit suffisamment souvent étant donné que le vent n'est pas pris en compte. Afin de vérifier que cette observation est réaliste, nous avons développé une version 6 qui prend en compte les perturbations liées au vent.
Le modèle $\mathcal{M}_{v6}[\mathcal{E}_{v6}, \mathcal{P}_{v6}]$ prend en compte la probabilité du vent comme constante, puis cette probabilité est considérée comme un paramètre dans la version 7 $\mathcal{M}_{v7}[\mathcal{E}_{v7}, \mathcal{P}_{v7}]$. Le modèle $\mathcal{M}_{v6}$ ressemble au modèle $\mathcal{M}_{v5}$ présenté ci-dessus. Dans ce modèle, nous avons ajouté des variables correspondant aux coordonnées des points A, B, C ainsi que des variables `ProbaWind1` à `ProbaWind5` qui correspondent aux probabilités des différentes forces du vent. Les perturbations liées au vent sont prises en compte dans ce modèle comme expliqué ci-dessous. Ces perturbations sont ajoutées à la position réelle du drone obtenue après le calcul de la déviation due au coefficient proportion α.

Calcul du l'effet du vent. La construction du modèle (version 6) se fait de la façon suivante : Nous prenons une valeur aléatoire `wind` dans $(0, 70]$ comme vitesse du vent (km/h). Plus précisément nous probabilisons la façon de tirer les valeurs de `wind` :

— nous avons une probabilité `ProbaWind1` pour `wind` $\in (0, 20]$,
— nous avons une probabilité `ProbaWind2` pour `wind` $\in [20, 30)$,
— nous avons une probabilité `ProbaWind3` pour `wind` $\in [30, 50)$,
— nous avons une probabilité `ProbaWind4` pour `wind` $\in [50, 70)$.

Dans la simulation, nous compensons `ProbaWind`i en la divisant par la probabilité d'avoir une valeur de `wind` au hasard (uniforme) afin de permettre à MCpMC de calculer la récompense r' en fin de simulation. Cela donne :

— pour `wind` $\in (0, 20]$: `ProbaWind1/(20/70)`,
— pour `wind` $\in [20, 30)$: `ProbaWind2/(10/70)`,
— pour `wind` $\in [30, 50)$: `ProbaWind3/(20/70)`,
— pour `wind` $\in [50, 70)$: `ProbaWind4/(20/70)`.

Nous avons également une variable `windAngle`, qui représente l'angle du vent par rapport à la trajectoire. Nous considérons que l'angle du vent est constant mais il serait facile d'autoriser sa modification en cours de simulation. Une fois l'angle fixé, nous

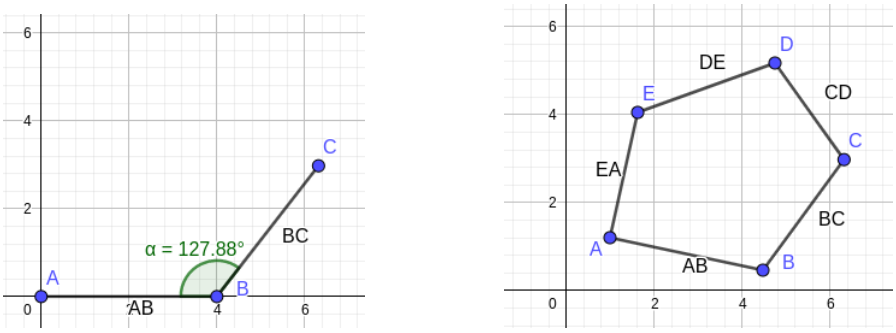
calculons $\cos$(`windAngle`) * `wind` pour les perturbations sur l'axe (x) et $\sin$(`windAngle`) * `wind` pour les perturbations sur l'axe (y).

Le modèle $\mathcal{M}_{v7}$ est identique au modèle $\mathcal{M}_{v6}$ mais les variables `ProbaWind1` à `ProbaWind5` sont des paramètres du polynôme résultat.

Ces deux versions ont encore une limitation (limitation L3.1) : pour un trajet de trois points, les versions 6 et 7 de notre modèle sont capables de calculer le polynôme, mais les versions 6 et 7 ne peuvent pas s'appliquer aux trajets réels contenant souvent plus de trois points.

5.10 Version 8 avec la méthode pSMC : prise en compte de longs trajets

Dans les versions 5 à 7 de notre modèle, nous commençons toujours avec le point $(0, 0)$ et un trajet AB suivant la ligne horizontale comme dans la Figure 5.17a. Cependant, un parcours réel ressemble plus à celui de la Figure 5.17b, donc plus complexe que celui de la Figure 5.17a. Il ne commence pas systématiquement à l'origine et avec une première ligne horizontale. Nous résolvons ce problème en changeant de repère à chaque point intermédiaire du trajet.



(a) (b)

FIGURE 5.17 – Trajet test (a) et trajet réel (b).

La version 8 de notre modèle $\mathcal{M}_{v8}[\mathcal{E}_{v8}, \mathcal{P}_{v8}]$ résout ainsi la limitation L3.1. Elle est similaire au modèle $\mathcal{M}_{v6}$ présenté à la fin de la section 5.9. Nous avons ajouté des variables correspondant aux coordonnées des points A', B', C', P', O, O', E, E', ainsi

que la liste des points du trajet et la liste des temps entre chaque point.

Notre point initial se trouve donc toujours à l'origine du repère, aux coordonnées $(0, 0)$. Le trajet suit initialement l'axe x. Nous calculons la distance D entre le premier point et le deuxième point. La fin de la première partie du trajet est donc aux coordonnées $(D, 0)$.

Quand le temps prévu du premier point au deuxième point est fini, nous calculons la probabilité d'aller en zone de sécurité 4 ou 5 sans changer la fin du trajet, puis nous changeons de repère pour que le deuxième point soit en coordonnées $(0, 0)$ et que le trajet du deuxième point au troisième point soit de nouveau sur l'axe x.

Dans ce modèle, nous incorporons aussi les perturbations sur l'axe x et la possibilité pour le drone de couper la trajectoire. Le repère est mis à jour au moment où le drone change d'objectif lorsque c'est le cas.

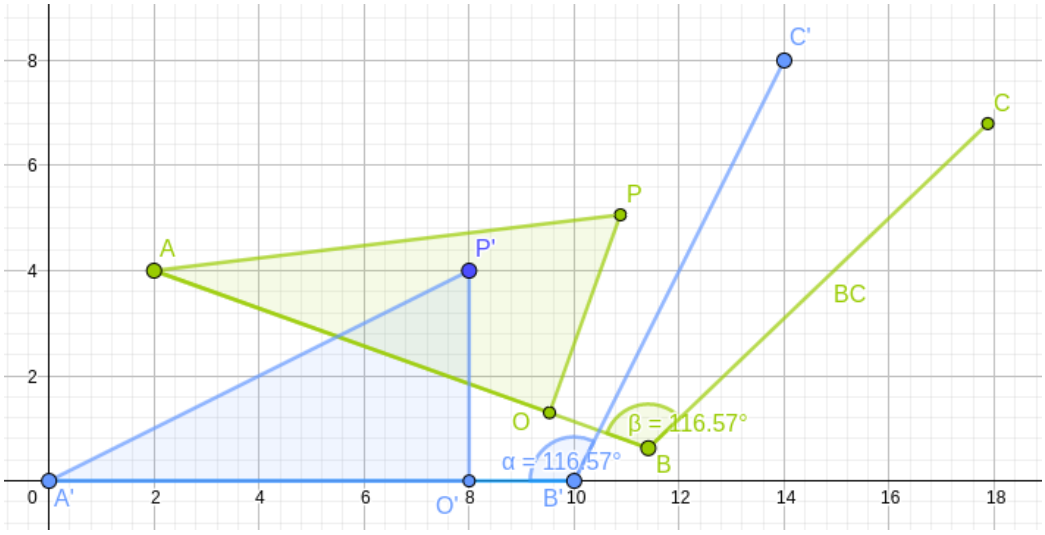


FIGURE 5.18 – Évolutions du drone selon différents axes.

Le trajet vert dans la Figure 5.18 est le trajet réel de A-B-C. Le trajet bleu A'-B'-C' est le trajet simulé dans notre modèle.

Examinons d'abord les trajets A-B et A'-B' : Soit $A = (A_x, A_y)$ et $B = (B_x, B_y)$.
La ligne AB a pour équation $y = k_{AB}x + b_{AB}$ avec

$$\begin{cases} k_{AB} = \frac{A_y - B_y}{A_x - B_x} \\ b_{AB} = A_y - k_{AB} * A_x \end{cases}$$

La distance entre le point A et le point B est $DistanceAB = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}$. La distance entre le point A' et le point B' est donc $DistanceA'B' = DistanceAB = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}$. Les coordonnées des points A' et B' sont donc les suivantes : $A' = (0, 0)$, $B' = (DistanceAB, 0)$ avec la ligne $A'B'$ d'équation $y = 0$.

Supposons que, lors d'une simulation, le drone arrive au point P' au lieu du point B'. Nous traçons une ligne verticale de P' à $A'B'$, qui intersecte l'axe x sur le point O'. Ainsi, les coordonnées de P' sont $P' = (P'_x, P'_y)$, avec P'_x la distance entre le point A' et le point O', et P'_y la distance entre P' et O'. Nous reportons ces informations sur le trajet original afin de placer les points O et P de la façon suivante : la ligne $A'P'$ a pour équation $y = k_{A'P'}x + b_{A'P'}$:

$$\begin{cases} k_{A'P'} = \frac{A'_y - P'_y}{A'_x - P'_x} = \frac{P'_y}{P'_x} \\ b_{A'P'} = A'_y - k_{A'P'} * A'_x = 0 \end{cases}$$

La ligne AP a donc pour équation $y = k_{AP}x + b_{AP}$, avec

$$\begin{cases} k_{AP} = k_{A'P'} + k_{AB} \\ b_{AP} = A_y - k_{AP} * A_x \end{cases}$$

Ainsi, $P_y = k_{AP}P_x + b_{AP}$.

La ligne PO est la ligne orthogonale à AB passant par P, donc

$$PO = \frac{|k_{AB}P_x - P_y + b_{AB}|}{\sqrt{k_{AB}^2 + 1}} = P'O' = |P'_y|$$

Avec les deux fonctions, nous obtenons les coordonnées du point P :

$$\begin{cases} P_x = A_x - P'_x\sqrt{\frac{A_y - B_y}{A_x - B_x}^2 + 1} \\ P_y = A_y - \left(P'_y + \frac{A_y - B_y}{A_x - B_x} \cdot P'_x\right)\sqrt{\frac{A_y - B_y}{A_x - B_x}^2 + 1} \end{cases}$$

En changeant de repère, le point B correspondra au point $(0, 0)$ et BC aura pour équation $y = 0$. Pour connaître les coordonnées du point P dans le nouveau repère, nous traçons la perpendiculaire à BC passant par P. Cette ligne intersecte BC au point E comme dans la Figure 5.19.

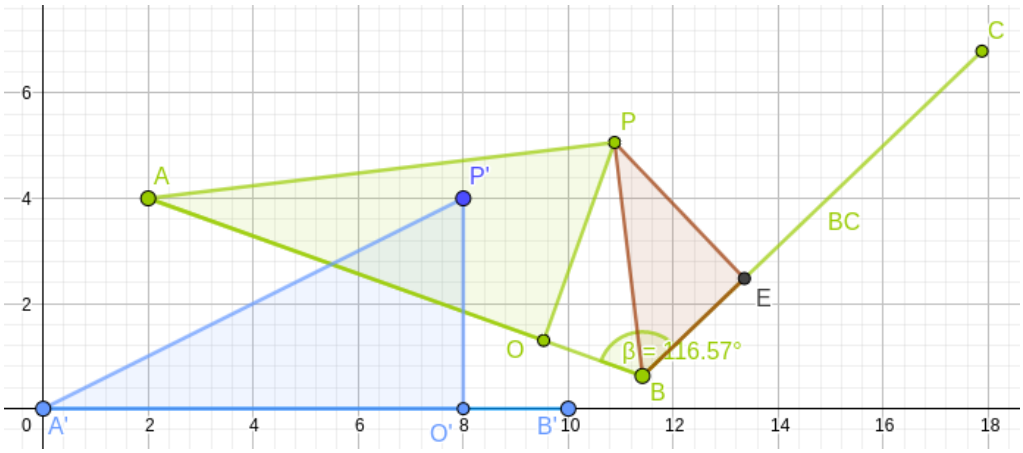


FIGURE 5.19 – Cas de changement de repère pendant un trajet.

Avec les points $B = (B_x, B_y)$ et $C = (C_x, C_y)$, la ligne BC a pour équation $y = k_{BC}x + b_{BC}$. PE est la ligne orthogonale à BC, nous avons donc :

$$PE = \frac{|k_{BC}P_x - P_y + b_{BC}|}{\sqrt{k_{BC}^2 + 1}}.$$

Nous avons aussi $PB^2 = PO^2 + BO^2 = PE^2 + BE^2$ donc

$$\begin{aligned} BE =& \sqrt{PO^2 + BO^2 - PE^2} \\ =& \sqrt{P'^2_y + (A'B' - P'_x)^2 - PE^2} \\ =& \sqrt{P'^2_y + (B'_x - P'_x)^2 - \frac{\left(\frac{B_y - C_y}{B_x - C_x}P_x - P_y + B_y - \frac{B_y - C_y}{B_x - C_x}B_x\right)^2}{(\frac{B_y - C_y}{B_x - C_x})^2 + 1}} \end{aligned}$$

Les valeurs de PE et BE sont les coordonnées de P dans le nouveau repère. Nous avons ainsi défini tous les éléments pour calculer systématiquement les changements de repère pendant la trajectoire du drone. La version 8 de notre modèle peut donc prendre comme paramètres une liste de points (une trajectoire) et une liste des temps pour parcourir les diverses portions de cette trajectoire. En sortie, le modèle de la version 8 nous permet d'obtenir un polynôme pour le calcul des probabilités d'aller en zone critique, en fonction des paramètres d'entrée.

A ce stade, nous avons pris en compte les paramètres principaux qui impactent notre modèle du drone ; nous arrêtons donc ici la construction des différentes versions du modèle du drone.

Les principales caractéristiques des modèles produits et les évolutions entre ces modèles sont résumées dans les Tables 5.4 et 5.5.

Ce modèle dans différentes versions va maintenant être utilisé comme entrée des outils d'analyse, afin d'étudier le comportement du drone par rapport aux zones de sécurité.

TABLE 5.4 – Caractéristiques des versions du modèle.

Version Modèle	**Outil ciblé**			**Probabilités du capteur**		**Probabilités du filtre**		**Probabilités globales de précision**		**Probabilités liées au vent**		**trajectoire # points**
	PRISM	PARAM	MCpMC	Const.	Param.	Const.	Param.	Const.	Param.	Const.	Param.	
1	x			x		x						2
2	x			x		x						2
3.1	x							x				2
3.2		x							x			2
3.3			x						x			2
4			x						x			2
5			x						x			3
6			x						x	x		3
7			x						x		x	3
8			x						x	x		> 3

TABLE 5.5 – Incréments des versions successives.

Version	**Incréments**
Version 1	Version de base.
Version 2	Prise en compte de la précision du filtre.
Version 3	Prise en compte de la position précédente du drone. Prise en compte des coordonnées exactes.
Version 4	Perturbations sur l'axe x.
Version 5	Changement possible d'objectif.
Version 6	Perturbations liées au vent (constantes).
Version 7	Perturbations liées au vent (paramètres).
Version 8	Changement de repère. Trajectoires longues (> 3 points).

CHAPITRE 6

EXPÉRIMENTATIONS ET INTERPRÉTATIONS

Nous avons présenté la méthode de construction ainsi que notre modèle formel complet dans le chapitre 5. Il s'agit jusque là d'un modèle qui sera utilisé comme entrée des outils d'analyse formelle. Ce chapitre porte sur la mise en œuvre de ce modèle dans divers *model checkers*. Nous interprétons aussi les résultats issus des expérimentations.

Tout d'abord, nous présentons les implémentations des différentes versions avec les *model checkers* utilisés. Ensuite nous présentons les résultats que nous avons obtenus et leur interprétation. Finalement, nous ferons des propositions pour établir un plan de vol diminuant la probabilité pour que le drone passe par la zone de danger.

6.1 Implémentation

Dans cette section, nous présentons les résultats des différentes versions et la manière dont nous utilisons les polynômes obtenus (à partir de la version 3). Dans la mise en œuvre, nous allons voir la relation entre le temps du vol et le nombre d'états. Nous présentons aussi l'analyse du polynôme avec la version 3 du modèle. Ensuite nous comparons les versions 4,6 et 7 pour savoir sur laquelle d'entre elles nous allons baser la version 8 de notre modèle. À la fin, nous présentons la simulation d'un trajet réel obtenue avec la version 8 de notre modèle.

6.1.1 Mise en œuvre de la version 2 avec le model checker PRISM

Nous avons implémenté notre modèle pour les versions 1, 2 et 3 avec l'outil PRISM. La première version est simple, elle engendre un automate qui ne comporte que 31

TABLE 6.1 – Nombres d'états et de transitions de la version 2 du modèle ($T_{reponse}$=1).

	Nombre d'états	**Nombre de transitions**
$T_{chemin} = 1$	106	130
$T_{chemin} = 2$	456	680
$T_{chemin} = 5$	1131	1955
$T_{chemin} = 10$	2006	3830
$T_{chemin} = 100$	15 506	35 330
$T_{chemin} = 400$	60 506	140 330
$T_{chemin} = 2400$	360 506	840 330

états et 55 transitions. La taille de la deuxième version dépend de T_{chemin} (le temps total du trajet en s) et $T_{reponse}$ (le temps de réponse en s). Nous fixons $T_{reponse} = 1$, l'outil doit donc simuler T_{chemin} fois le filtre pour un trajet donné.

Les nombres d'états et de transitions pour différentes instances de la version 2 sont présentés dans la Table 6.1. Notons que ces nombres sont multipliés par 4 lorsque $T_{chemin} = 2$. En réalité, notre IMU envoie la position toutes les 0,025s (ce qui correspond à une fréquence de $f = 40$Hz). Pour un trajet de 10s, il faut donc 400 boucles d'estimation de la position du drone, ce qui correspond à $T_{chemin} = 400$ si nous conservons $T_{reponse} = 1$. Pour un trajet d'une minute, 2400 boucles sont alors nécessaires.

La version 3 du modèle prend aussi en compte la dernière position corrigée du drone. La taille des zones et la position du drone ne sont donc plus des constantes dans le modèle mais des variables. PRISM n'est alors plus capable de faire la simulation car ces variables ne sont pas bornées.

Par exemple, supposons que `zone1` $\in [0, 20]$, `zone2` $\in (20, 40]$, `zone3` $\in (40, 60]$, `zone4` $\in (60, 80]$, `zone5` $\in (80, +\infty]$. Nous ne pouvons pas définir les variables avec PRISM car `zone5` n'est pas bornée. Nous choisissons donc `zone5` $\in (80, 100]$.

L'unité pour la taille des zones est le décimètre, car le mètre nécessite des nombres décimaux, ce qui ralentit le calcul. En centimètres les valeurs sont encore plus grandes, ce qui augmente le nombre d'états. Nous supposons un trajet de 50m (500dm) en 5min avec $(T_{reponse} = 1, T_{chemin} = 300s)$. Même en ne prenant pas en compte le temps de réponse de l'IMU $T_{reponse} = 0.025s$, le nombre d'états et de transitions est trop grand.

Ainsi, pour l'expérimentation avec un trajet de 50m, nous n'avons pas des nombres d'états et de transitions exacts à donner car nous n'avons pas pu finir ce test avec

TABLE 6.2 – Temps obtenus pour la version 3 (avec MCpMC).

Simulations	Temps (seconde)
10k	32.81
20k	56.37
50k	153.50
100k	390.22

PRISM. En effet, après deux heures d'attente pour un test avec $T_{chemin} = 5s$ et $T_{reponse} = 1s$, nous avons décidé de changer d'outil et donc de méthode.

6.1.2 Mise en œuvre de la version 3 avec l'outil MCpMC

La version 3 du modèle est présentée dans la section 5.8. Pour rappel, cette version 3 améliore les versions 1 et 2 en prenant en compte la position précédente du drone et en considérant les probabilités liées à la précision des estimations comme des paramètres et non plus comme des constantes. Ici nous présentons les expérimentations effectuées avec cette version 3 et commentons les résultats obtenus.

Nous développons notre modèle pour le prototype MCpMC depuis la version 3. Contrairement à PRISM, MCpMC n'indique pas le nombre d'états car l'outil ne construit jamais le modèle complet étant donné qu'il se contente de le simuler. Nous avons donc fait un test identique à celui fait avec PRISM sur la même machine *Intel(R) Core(TM) i5-7200U*, avec $T_{chemin} = 5s$ et $T_{reponse} = 1s$.

Avec les mêmes constantes `zone1` $\in [0, 20]$, `zone2` $\in (20, 40]$, `zone3` $\in (40, 60]$, `zone4` $\in (60, 80]$ et `zone5` $\in (80, +\infty]$, nous obtenons la Table 6.2 qui donne le temps d'exécution en fonction du nombre de simulations.

Le polynôme obtenu est beaucoup plus long (un nombre très important de termes) que prévu (nous l'avons mis dans l'annexe 8.2.1). Pour l'analyser, nous avons sélectionné les plus grands coefficients et les plus petits coefficients pour voir quel paramètre impacte plus ou moins le résultat. Le résultat est présenté dans la Table 6.3, où nous pouvons voir les degrés associés à chacun des paramètres pour chaque coefficient réel (`PF`i pour `ProbaFilter`i).

Par exemple, les deux premières lignes correspondent aux termes `49.21875` * `Proba-`

`Filter1` * `ProbaFilter2` * `ProbaFilter3` * `ProbaFilter4` * `ProbaFilter5` et `37.6875` * `ProbaFilter1` * `ProbaFilter2` * `ProbaFilter4`^{2} * `ProbaFilter5`.

Étant donné que $T_{chemin} = 5s$ et $T_{reponse} = 1s$, la position du drone est mise à jour au plus cinq fois avant la fin de la simulation. Notons que les simulations s'arrêtent dès lors que le drone se trouve dans les zones de sécurité 4 ou 5 puisque cela nous permet de conclure sur la satisfaction de la propriété à vérifier.

TABLE 6.3 – Analyse du polynôme résultat de la version 3.

	Occurrences des paramètres				
Coefficients	`PF1`	`PF2`	`PF3`	`PF4`	`PF5`
49.21875	1	1	1	1	1
37.6875	1	1	0	2	1
36.875	1	0	1	2	1
36.75	0	1	1	2	1
34.0625	1	0	1	1	2
33.71875	1	1	0	1	2
31.375	0	1	1	1	2
...	...	...	...	...	...
0.04375	0	2	1	0	1
0.03125	1	0	4	0	0
0.03125	1	3	1	0	0
0.03125	2	1	2	0	0
0.03125	2	2	1	0	0
0.025	0	0	0	4	0
0.00625	0	0	1	3	0

Dans la Table 6.3, les grands coefficients incluent souvent 2 fois `ProbaFilter4` ou `ProbaFilter5`. En effet `ProbaFilter4` et `ProbaFilter5` augmentent la probabilité pour le drone d'atteindre les zones de sécurité 4 ou 5.

Le coefficient d'un certain terme peut être grand pour deux raisons : soit c'est un terme qui apparait souvent dans les polynômes donnés par nos simulations (par exemple `ProbaFilter1` * `ProbaFilter2` * `ProbaFilter3` * `ProbaFilter4` * `ProbaFilter5`), soit l'apparition de ce terme est fortement liée à la satisfaction de la pro-

priété (le terme $\texttt{ProbaFilter5}^5$ implique des erreurs importantes à toutes les étapes du calcul). Malgré tout, la probabilité de $\texttt{ProbaFilter5}^5$ est une des probabilités les plus rares ($\frac{1}{5^5}$), mais le coefficient est 0.5. Même si ce cas est rare, lorsqu'il se produit, le drone a de très fortes probabilités d'atteindre les zones de sécurité 4 ou 5.

Certaines combinaisons des paramètres n'apparaissent pas dans le polynôme obtenu, par exemple $\texttt{ProbaFilter1}^5$ ou $\texttt{ProbaFilter3}^5$. Il y a deux raisons possibles : soit cette combinaison de paramètres ne permet pas (ou avec une très faible probabilité) d'atteindre les zones 4 et 5, soit cette combinaison n'a jamais eu lieu lors de nos 100k simulations.

Les versions 4 et 5 produisent des polynômes plus complexes, mais ils sont moins intéressants que pour le modèle complet (versions 6,7 ou 8).

6.1.3 Comparaison des versions 4, 6 et 7 en MCpMC

Les versions 4 à 7 du modèle sont présentées dans la section 5.9. Pour rappel, la version 4 améliore la version 3 en prenant en compte les erreurs d'estimation sur l'axe x en plus de celles sur l'axe y. Dans la version 5, nous avons de plus intégré les potentiels changements d'objectifs lorsque le drone est en retard sur sa trajectoire. Finalement, les versions 6 et 7 intègrent les perturbations liées au vent en tant que constantes dans la version 6 puis en tant que paramètres dans la version 7.

Comme vu précédemment, les polynômes obtenus en sortie sont trop longs pour être comparés ou analysés manuellement. Nous ne présentons donc pas les polynômes obtenus lors d'expérimentations réalistes, mais plutôt leur évaluation pour différentes valeurs des paramètres $\texttt{ProbaFilter1}$-$\texttt{ProbaFilter5}$ ($\texttt{PF1}$-$\texttt{PF5}$).

Une comparaison centre les versions 6 et 7 a été faite avant de développer la version 8 du modèle. Le but était de savoir si la version 8 de notre modèle devait se baser sur la version 7 ou la version 6. Les versions 6 et 7 sont deux versions développées en parallèle. La version 6 est plus rapide, mais dans la version 7 le vent est en paramètre dans le polynôme.

Nous définissons deux scénarios afin d'évaluer nos polynômes. Le scénario 1 est un scénario proche de la réalité. Le scénario 2 est pour voir plus clairement l'impact des paramètres. Pour le scénario 1 $\texttt{PF1} = 0.15$, $\texttt{PF2} = 0.3$, $\texttt{PF3} = 0.4$, $\texttt{PF4} = 0.1$ et $\texttt{PF5} = 0.05$. Pour le scénario 2 $\texttt{PF1} = 0.1$, $\texttt{PF2} = 0.25$, $\texttt{PF3} = 0.35$, $\texttt{PF4} = 0.2$ et $\texttt{PF5} = 0.1$.

Dans ces deux scénarios, la zone de sécurité 4 est située à 8m et la zone de

sécurité 5 est située à 50m de la zone de vol. `PF1-PF5` correspond à la probabilité que l'écart entre la position effective et la position estimée soit de $[0, 2]$ mètre pour `PF1`, de $(2, 4]$ mètres pour `PF2`, de $(4, 6]$ mètres pour `PF3`, de $(6, 8]$ mètres pour `PF4` et de $(8, 10]$ mètres pour `PF5`.

D'après l'expert de PIXIEL, le bruit des capteurs peut aller jusqu'à 10 mètres, mais grâce au filtre, la vitesse et la distance de correction réduisent l'erreur. Les bruits dépendent aussi de l'environnement de vol et de la qualité des composants du drone. Les éléments de l'environnement peuvent aussi perturber les drones. Par exemple lors d'un spectacle de drones, si des spectateurs oublient d'éteindre le WIFI ou le point d'accès de leur smartphone, les signaux interférent avec les drones.

Le scénario 1 est plus réaliste que le scénario 2. Pour des drones ayant des composants de qualité, la probabilité `PF1` devrait être plus grande et `PF3` plus petite, mais à cause de mauvais résultats des expérimentations sur la chaîne de production utilisée, nous ne pouvons pas fournir de valeurs exactes.

Dans les simulations des versions 6 et 7, les paramètres liés au vent correspondent à la probabilité d'avoir une force de vent de $[0, 20]$ km/h, $(20, 30]$ km/h, $(30, 50]$ km/h et $(50, 70]$ km/h respectivement ; nous les avons définis à 0.55, 0.43, 0.01 et 0.01 (ce qui correspond aux conditions typiques de la ville de Nantes). La direction du vent est toujours face au drone (direction identique dans tous les repères utilisés).

Dans la Table 6.4, nous avons synthétisé les résultats pour 10k, 20k et 50k simulations des versions 4, 6 et 7 de notre modèle en fixant $T_{chemin} = 5s$ et $T_{reponse} = 1s$. V1 et V2 représentent deux expériences avec les mêmes données afin de comparer les résultats obtenus. Dans chaque case, nous avons calculé la probabilité et l'intervalle de confiance pour les valeurs des paramètres de chacun des scénarios identifiés ci-dessus. Avec les valeurs des paramètres que nous avons choisies pour l'évaluation de nos polynômes, nous observons que les probabilités obtenues pour le scénario 1 sont plus faibles que celles obtenues pour le scénario 2. L'intervalle de confiance pour le scénario 1 est aussi plus faible. Cet intervalle est aussi un polynôme qui change avec les valeurs que nous avons données.

Nous observons aussi que les versions 4 et 6 ont un temps d'exécution beaucoup plus rapide comparé à la version 7 dans laquelle la force du vent est paramétrique. L'augmentation du nombre de paramètres ralenti en effet le temps de calcul de la version 7 qui doit de plus faire plus de simulations pour obtenir un niveau de précision

similaire. Pour cette raison, nous avons choisi de baser notre version 8 sur la version 6 (dans laquelle le vent est pris en compte de manière non paramétrique).

TABLE 6.4 – Analyse du polynôme résultat des versions 4, 6 et 7.

	Modèle	**10k**		**20k**		**50k**	
		V1	**V2**	**V1**	**V2**	**V1**	**V2**
T. d'exécution	version 4	28s		51-54s		142-143s	
Scénario 1	version 4	4.99%	5.09%	4.74%	5.10%	4.91%	4.98%
Interv. de conf.		±0.85%	±0.82%	±0.55%	±0.56%	±0.36%	±0.37%
Scénario 2	version 4	10.38%	10.04%	9.82%	10.05%	9.95%	9.81%
Interv. de conf.		±1.15%	±1.12%	±0.79%	±0.80%	±0.51%	±0.51%
T. d'exécution	version 6	28s		53-54s		149-155s	
Scénario 1	version 6	5.44%	5.31%	5.61%	5.21%	5.59%	5.47%
Interv. de conf.		±0.98%	±0.86%	±0.69%	±0.64%	±0.42%	±0.43%
Scénario 2	version 6	10.8%	10.9%	10.8%	10.8%	10.9%	10.7%
Interv. de conf.		±1.35%	±1.32%	±0.91%	±0.92%	±0.57%	±0.57%
T. d'exécution	version 7	185-190s		311-314s		612-621s	
Scénario 1	version 7	4.95%	5.97%	5.28%	6.62%	4.16%	5.61%
Interv. de conf.		±5.22%	±5.71%	±4.71%	±6.25%	±1.86%	±4.38%
Scénario 2	version 7	9.55%	9.87%	10.3%	11.3%	9.57%	10.7%
Interv. de conf.		±8.40%	±7.86%	±7.04%	±7.89%	±5.29%	±3.99%

Pour les probabilités et les intervalles de confiance, la version 6 est meilleure que la version 7. Cependant la version 7 est la seule à prendre en compte des paramètres liés au vent dans son polynôme.

Les figures 6.1 et 6.2 présentent les polynômes obtenus avec la version 7 pour $T_{chemin} = 1$ et $T_{reponse} = 1$ en 10k simulations et en 100k simulations respectivement. Nous pouvons voir que les coefficients sont très proches dans les deux cas. Pourtant, avec la Table 6.4 nous observons une différence sur l'intervalle de confiance avec 10k simulations et 50k simulations. Dans la Table 6.4, les paramètres sont $T_{chemin} = 5$ et $T_{reponse} = 1$ donc le modèle comporte plus que $5^{T_{chemin}}$ états et transitions. La différence est ainsi beaucoup plus marquée que pour $T_{chemin} = 1$ et $T_{reponse} = 1$, car l'intervalle de confiance est un polynôme qui dépend des valeurs des paramètres et de la longueur des simulations.

```
senario:  s1 Time= 1 , f= 1 , NbSim= 10000 , > 80 , Angle Wind= 0.0 , for(0, 1 )
0.0525*ProbaFilter3*ProbaWind1 + 0.0315*ProbaFilter3*ProbaWind2 + 0.05075*ProbaF
ilter3*ProbaWind3 + 0.0455*ProbaFilter3*ProbaWind4 + 0.94325*ProbaFilter4*ProbaW
ind1 + 0.9835*ProbaFilter4*ProbaWind2 + 0.98*ProbaFilter4*ProbaWind3 + 1.01325*P
robaFilter4*ProbaWind4
Computation duration (s):  7.496161699295044
 p:  0.0524195000000000  q:  0.943897500000000  icw:  0.00684357399285550  sum:
 0.996317000000000
```

FIGURE 6.1 – Le polynôme obtenu pour 10k simulations (version 7).

```
senario:  s1 Time= 1 , f= 1 , NbSim= 100000 , > 80 , Angle Wind= 0.0 , for(0, 1 )
0.047775*ProbaFilter3*ProbaWind1 + 0.0434*ProbaFilter3*ProbaWind2 + 0.05285*Proba
Filter3*ProbaWind3 + 0.0532*ProbaFilter3*ProbaWind4 + 0.991375*ProbaFilter4*Proba
Wind1 + 0.98595*ProbaFilter4*ProbaWind2 + 0.966525*ProbaFilter4*ProbaWind3 + 0.99
505*ProbaFilter4*ProbaWind4
Computation duration (s):  110.90503025054932
 p:  0.0540414000000000  q:  0.951688150000000  icw:  0.00220008291393564  sum:
1.00572955000000
```

FIGURE 6.2 – Le polynôme obtenu pour 100k simulations (version 7).

Le polynôme obtenu pour $T_{chemin} = 5$ et $T_{reponse} = 1$ est plus long, donc difficile a présenter et à analyser. Pour cette raison, nous présentons ici uniquement les polynômes pour $T_{chemin} = 1$ et $T_{reponse} = 1$ en figure 6.1 et figure 6.2. Les polynômes plus "réalistes" pourraient être analysés avec la même méthode que celle utilisée dans la section 6.1.2, ce qui permettrait de donner des informations sur les termes ayant le plus d'effet sur la probabilité de pénétrer dans les zones interdites.

6.1.4 Mise en œuvre de la version finale (version 8) avec MCpMC

La version 8, présentée en section 5.10, est la version la plus complète de notre modèle. Elle est basée sur la version 6 et y ajoute les changements de repère permettant de considérer des trajets longs (plus de 3 points), tout en prenant en compte les erreurs d'estimation sur les deux axes x et y, les perturbations liées au vent et les changements d'objectifs causés par les retards. Rappelons que, dans cette version, les probabilités des perturbations liées au vent sont des constantes et non des paramètres. Après avoir finalisé et testé toutes les versions de notre modèle, nous avons récupéré un plan de vol réel d'un drone pour faire un test complet. Ce plan de vol est un trajet de 139 secondes avec 18 points de passage. Toujours avec $T_{reponse} = 1$, l'automate correspondant théoriquement à notre modèle d'entrée contient alors au moins 5^{139} états. Le temps d'analyse pour la version 8 est très bon : 92 secondes pour 10k

simulations, 408 secondes pour 50k simulations et 899 secondes pour 100k simulations. En revanche, le nombre d'états étant supérieur à $5^{139} = 1,434929627 * 10^{97}$, 100k simulations reproduisent seulement une partie négligeable de tous les cas possibles. Cela n'est pas très satisfaisant.

Les valeurs des paramètres permettent de savoir quel cas a le plus de probabilité d'atteindre les zones de sécurités 4 ou 5. Nos polynômes nous aident à déterminer comment reproduire chaque cas. Dans la prochaine section nous présentons les manières de diminuer les probabilités d'aller dans les zones de sécurité interdites.

6.2 Réduire la probabilité d'entrer en zone interdite

Après de nombreux tests du modèle avec des trajectoires différentes, nous cherchons à comprendre la raison des différences entre les probabilités observées. En effet, le but du modèle est de trouver une trajectoire avec la plus faible probabilité d'arriver dans la zone danger tout en passant par les principaux points de la trajectoire initiale. Différentes raisons peuvent expliquer les différences de probabilité. Donc quand il y a un changement, nous essayons de reproduire les conditions dans lesquelles cela arrive pour en comprendre la raison. Lorsque nous observons 5 fois de suite le même changement de probabilité alors, nous modifions d'autres valeurs du contexte pour voir si cela impacte les changements de probabilité, nous faisons cela jusqu'à trouver les conditions dans lesquelles la probabilité change.

Dans cette section nous proposons des méthodes pour réduire la probabilité d'atteindre la zone de danger. Nous avons utilisé *GeoGebra* pour proposer des illustrations.

Nous avons étudié 2 pistes, la réduction du nombre de points sur un trajet et l'allongement du temps de trajet entre deux points du trajet.

6.2.1 Réduction du nombre de points sur une ligne droite

En effectuant plusieurs simulations, nous avons remarqué que la trajectoire entre un point A(0,0) et un point B(200,0) avec $T_{chemin} = 4$ a moins de probabilité d'entrer dans la zone interdite que la même trajectoire utilisant un point intermédiaire D(100,0) avec $T_{chemin} = 2$ entre A et D et $T_{chemin} = 2$ entre D et B. Il semble donc qu'ajouter un

point intermédiaire sans rien changer d'autre augmente la probabilité d'entrer dans la zone interdite.

Nous avons étudié ce cas avec GeoGebra. Afin d'accentuer les différences, nous avons diminué les distances parcourues.

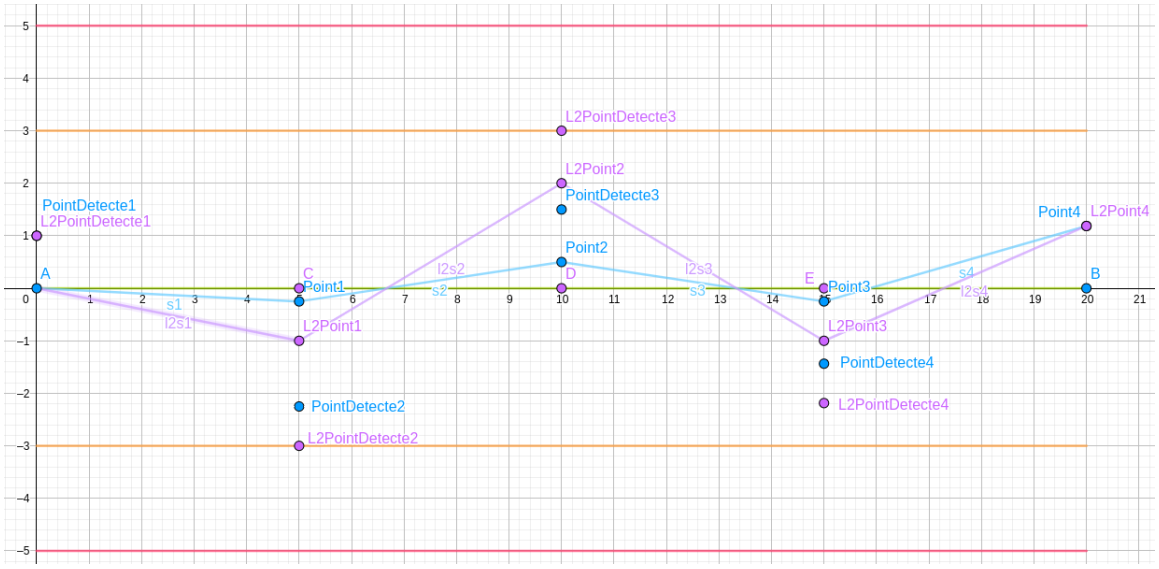


FIGURE 6.3 – Trajet A-B.

Dans la Figure 6.3, le drone démarre à la position (0,0). Deux trajets, bleu et violet, vont du point A(0,0) au point B(20,0). Dans le trajet violet, nous avons ajouté des points intermédiaires C(5,0), D(10,0) et E(15,0). Nous supposons que $T_{reponse}$ est identique pour les deux trajets et que la somme des temps des trajets intermédiaires du trajet violet est égale à T_{chemin} pour le trajet bleu. Pour simplifier la figure, nous avons choisi $T_{reponse} = 1$. Ainsi, les estimations des positions pour les deux trajets se font exactement lors des passages aux points A, C, D, E et B.

Les deux trajets commencent au même point, et nous supposons que la position estimée est identique (`PointDetecte1` et `L2PointDetecte1`). Dans la suite, les erreurs de mesure sont les mêmes sur les deux trajets (nous notons erreur$_i$ l'erreur à la ième mesure). Nous obtenons donc : `PointDetecte`$_{i+1}$ = erreur$_i$ + `Point`$_i$ et `L2PointDetecte`$_{i+1}$ = erreur$_i$ + `L2Point`$_i$.

Nous observons que les deux trajets terminent au même point ; mais pendant le parcours, le trajet violet est plus éloigné de son plan de vol que le trajet bleu. Cela s'explique par le fait que les corrections sur le trajet violet sont plus "brutales" que celles du trajet bleu. En effet, les distances restantes à parcourir avant le prochain ob-

jectif sont plus petites sur le trajet violet, donc son coefficient de correction "proportion α" est plus grand.

Cette étude rapide montre que lorsqu'un trajet en ligne droite comporte moins de points intermédiaires, le modèle établit que la probabilité de franchir la zone de danger diminue. La différence exacte dépend aussi du vent, des valeurs de *ProbaFiltre* et de la longueur du trajet.

6.2.2 Allongement du temps des trajets

Nous partons du constat que des temps d'attente sont utilisés pour recaler le drone sur ses temps de trajets. Ces temps d'attente qui sont "perdus" peuvent être récupérés pour la sécurité.

Lorsque le drone est en retard sur son temps de trajet, ajouter un temps d'attente dans le plan de vol pour qu'il rattrape son retard est une bonne idée. Cependant, prolonger le temps de trajet entre deux points en incluant le temps de retard peut néanmoins être une meilleure stratégie. Cette première situation est illustrée dans la Figure 6.4. Dans cet exemple, le drone doit faire le trajet A-B en 4 secondes avec une fréquence du filtre de 1Hz. La première stratégie en bleu consiste à donner au drone 4 secondes pour faire le trajet. Pour rattraper un éventuel retard, il doit attendre 4 secondes supplémentaires une fois arrivé au point B. La seconde stratégie en violet consiste à donner 8 secondes (4 secondes de trajet + 4 secondes d'attente) pour faire le trajet. Dans ce cas, le drone va atteindre le point B deux fois moins vite mais le filtre enverra plus de fois sa position. Ainsi, cela donnera au drone plus d'opportunités pour corriger sa trajectoire.

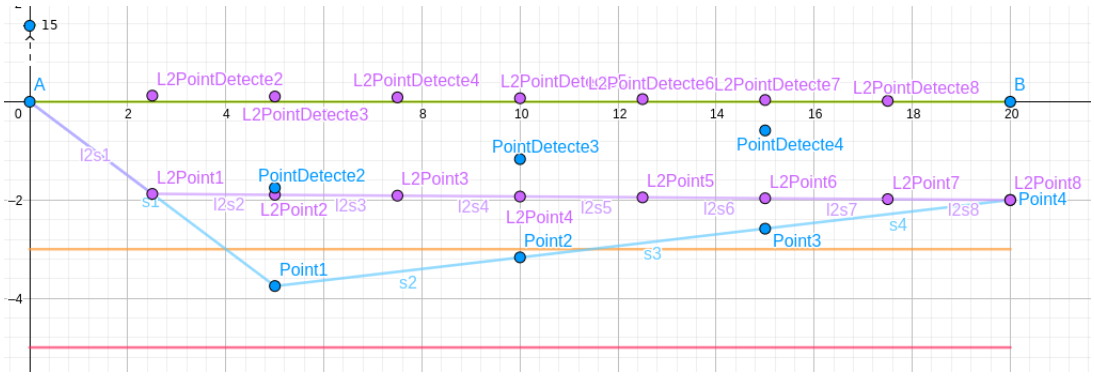


FIGURE 6.4 – Allongement des temps de trajet (situation 1).

Pour les deux trajets bleu et violet, au début avec $n = 1$, le bruit est très grand : $\texttt{PointDetecte}_1 = \texttt{L2PointDetecte}_1 = 15$. Ensuite, le bruit vaut 2 pour tout $n > 1$.

Nous pouvons voir que $\texttt{L2Point}_2$ est mieux positionné que $\texttt{Point}_1$ car le bruit a été corrigé plus tôt (dès $\texttt{L2Point}_1$). À la suite, les deux trajets convergent car les bruits sont constants. Cependant, nous remarquons que le trajet bleu entre dans la zone interdite au point $\texttt{Point}_1$ alors que le trajet violet n'y entre pas.

Normalement le drone ne revient pas sur une trajectoire quand il sait qu'il y a une déviation. Il recalcule une nouvelle trajectoire pour arriver au but en temps demandé. Avec cette nouvelle trajectoire, il corrige son erreur petit à petit. Pendant ce temps, il peut encore obtenir des bruits d'estimation. Mais dans l'algorithme de certains filtres, il y a de l'apprentissage sur les erreurs corrigées. Nous essayons de produire ce cas dans la deuxième situation. La deuxième situation (Figure 6.5) inclut plusieurs sortes de bruits. Au début, le premier bruit est identique à celui de la situation 1 : $\texttt{PointDetecte}_1 = \texttt{L2PointDetecte}_1 = 15$. Ensuite les bruits sont `erreur1`=2, `erreur2`=0 et `erreur3`=-2.

Pour $x \in \{5, 10, 15\}$, les points `PointDetecte` et `L2PointDetecte` subissent les mêmes bruits `erreur1`, `erreur2` et `erreur3`, et pour $i \in \{2, 4, 6, 8\}$,

$$\texttt{L2PointDetecte}_i = \texttt{L2Point}_{i-1} + \frac{\texttt{L2PointDetecte}_{i-1} - \texttt{L2Point}_{i-2}}{2}.$$

Nous observons que, dans cette situation, le trajet violet est presque toujours meilleur que le trajet bleu.

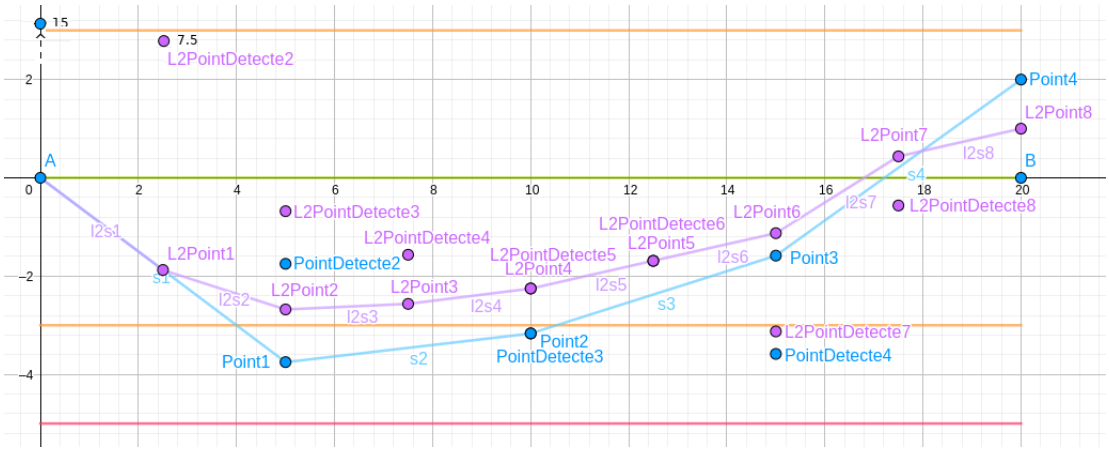


FIGURE 6.5 – Allongement des temps de trajet (situation 2).

En conclusion de cette étude, nous confirmons dans le cadre de notre modèle que, avec une précision identique, un filtre à haute fréquence est meilleur qu'un filtre à basse

fréquence. Étant donné que la vitesse des capteurs est souvent plus importante que celle des filtres, la fréquence des estimations dépend plus de la fréquence du filtre que de celle des capteurs. Un filtre à haute fréquence permet de vérifier plus souvent, et donc corrige plus souvent. Malheureusement, un filtre à haute fréquence est souvent moins précis qu'un filtre basse fréquence comme expliqué dans le chapitre 2. Nous n'avons pas obtenu la précision des filtres disponibles en conditions réelles, donc nous ne pouvons pas les comparer. Cela aurait pu permettre de savoir quel filtre utiliser en fonction de la longueur du trajet et de la force du vent, ce qui prouverait fermement notre conjecture (section 3.1).

Nous ne pouvons pas conclure sans les valeurs exactes des paramètres. Cependant avec les polynômes obtenus, nous pouvons facilement calculer la probabilité pour que le drone soit hors de la zone sécurité, une fois que les paramètres du filtre seront fournis.

Nous tirons plusieurs conclusions de nos expérimentations. Dans un premier temps, il est important de remarquer que les polynômes que nous obtenons ne sont pas facilement exploitables à cause de leur taille. Néanmoins, l'utilisation de simples tableaux permettant de repérer à quelles variables sont liés les plus grands coefficients (comme présenté dans la section 6.1.2) peut être utile pour identifier les paramètres ayant le plus d'impact sur la probabilité d'entrer dans la zone interdite. De plus, une fois le polynôme obtenu, il est facile de le "tester" avec des valeurs différentes des paramètres. Cela parait en effet beaucoup plus efficace que d'effectuer la vérification complète avec des modèles différents pour chaque scénario.

Dans un deuxième temps, nous avons remarqué grâce aux nombreuses simulations effectuées que l'ajout de points de passage intermédiaires lors d'un vol en ligne droite avait tendance à empirer la probabilité d'entrer en zone interdite. De même, l'utilisation de filtres à haute fréquence semble préférable à l'utilisation de filtres à basse fréquence, pourvu que la précision soit identique.

Finalement, un point qui n'a pas été pris en compte est la détérioration de la précision de certains filtres lors des changements de trajectoire aux points intermédiaires. Cette détérioration est due aux modifications "brutales" des mesures à cause des virages. Malheureusement, nous ne pouvons pas prendre en compte ce comportement dans notre modèle à moins d'établir un modèle plus précis du filtre lui-même.

CHAPITRE 7

CONCLUSION

Dans le domaine du drone multirotors automatique, les défaillances critiques peuvent provenir de défaillances des composants physiques, du système du drone ou de l'environnement. Le but de cette thèse effectuée dans un cadre R&D avec l'entreprise PIXIEL, était de trouver une méthode pour rendre le drone plus fiable. En considérant qu'il n'est pas possible de modifier l'environnement du vol, que l'étude des composants physiques est pertinente mais ne relève pas du domaine de l'informatique (logiciel) qui est notre champ de recherche, nous avons focalisé notre étude sur le système du drone. Nous avons particulièrement étudié les trois modules les plus importants dans le système du drone, que sont le filtre, le PID et la modulation. Nous avons ainsi proposé une méthode pour modéliser le drone, et utiliser le modèle pour détecter les erreurs de trajectoire lors du vol d'un drone.

Dans le cas idéal, le drone vole toujours normalement, ce sont les perturbations de ses composants ou de l'environnement qui provoquent des erreurs de trajectoire. Nous ne cherchons pas à détecter et corriger les inévitables perturbations, mais plutôt une façon de s'y adapter. Nous avons donc présenté une méthode de construction d'un modèle formel pour l'étude de la sûreté d'un drone en mode de vol automatique et avec un plan de vol prédéfini. Ce modèle cible est construit à l'aide de chaînes de Markov paramétrées. Il adapte le contrôleur de vol, prend en compte à l'aide de paramètres probabilistes, la précision des estimations de la position et de l'orientation en utilisant les capteurs, le filtre et aussi les perturbations liées au vent. Ainsi la construction du modèle a fait l'objet de plusieurs versions construites de façon incrémentale selon les caractéristiques ou les paramètres pris en compte. L'analyse du modèle nous donne des probabilités de vérifier les propriétés voulues ou des probabilités paramétriques. L'étude et l'optimisation de ces probabilités paramétriques permettront à l'entreprise PIXIEL d'affiner le compromis entre la fréquence et la précision du filtre afin de choisir judicieusement, en fonction des vols à effectuer, les bons composants à utiliser.

Ce choix de modélisation et d'analyse est justifié par le fait que le plan de vol est

un élément très facile à changer, et il permet aussi de sécuriser le spectacle du drone.

Notre méthode permet de construire de façon incrémentale un modèle formel du drone en prenant successivement en compte les différents composants du drone, et en utilisant comme paramètres du modèle certaines caractéristiques de ces composants. À l'issue de notre travail nous pouvons établir que le modèle minimal pour étudier convenablement le fonctionnement du drone doit correspondre à notre version 4. Il faut en effet y intégrer non seulement les comportements des capteurs et du filtre mais aussi les paramètres de la trajectoire du vol. Ce modèle minimal du drone comprend tous les principaux composants du contrôleur de vol, leur fonctionnement et leurs interactions. Il est donc utilisé pour étudier un aspect de la sécurité : éviter le vol du drone au dessus du public. Même avec des simplifications des paramètres le modèle est de grande taille.

Nous tirons la leçon après nos différentes expériences avec les outils de *Model Checking* que le *Model Checking statistique paramétré* est le plus adapté pour des modèles de complexité comparable à celle du drone. Nous avons fait plusieurs expériences avec différents outils tels que PRISM et PARAM, mais nous sommes allés au delà des capacités de ces outils avec la complexité relative du système de drone modélisé avec un modèle probabiliste paramétré.

Nous avons étudié la correction du modèle que nous construisons. Cette étude est focalisée sur le raisonnement sur les mouvements dans le plan. Nous nous sommes pour cela appuyés sur l'outil GeoGebra qui est une référence indiscutable pour la géométrie dans l'espace. Les équations modélisant les mouvements du drone sont ainsi vérifiées avant leur mise en œuvre dans notre modèle.

Grâce à la méthode *model checking statistique paramétrique* et au nouveau prototype MCpMC associé, nous pouvons analyser un très grand modèle paramétré, dans un temps raisonnable. En effet cette méthode présente l'avantage d'exploiter les transitions probabilisées paramétrées, les échantillons des exécutions symboliques et exprime le calcul de la probabilité d'une propriété par un polynôme en termes des paramètres probabilistes utilisés. Le prototype MCpMC a été mis à l'épreuve et a démontré son efficacité sur des modèles probabilistes de taille conséquente comme celui du drone.

Nous pouvons faire varier les valeurs des paramètres du modèle pour différentes

trajectoires de vol, et comparer les probabilités ou les polynômes de sortie, pour que le drone ne rentre pas dans les zones dangereuses pour le public. Lorsque les écarts sont importants entre les résultats, une recherche des causes est systématiquement effectuée pour expliquer les écarts et leur impact sur la sécurité : éviter certaines configurations des paramètres du drone ou bien confirmer certaines autres configurations garantissant une meilleure sécurité.

De manière générale, nous simulons plusieurs trajets différents et recherchons les très faibles probabilités d'entrer dans les zones interdites, et quelles configurations et quels trajets de vol du drone sont les plus sûrs.

En cela, le modèle élaboré est très utile pour l'entreprise PIXIEL ; elle peut s'en servir pour améliorer les trajets de vol des drones ainsi que leurs configurations de façon à garantir la sécurité du public ; c'était l'objectif des travaux initiés.

Une limitation de notre travail est que nous ne pouvons pas analyser correctement tous les polynômes obtenus. Nous avons analysé les polynômes courts comme ceux obtenus dans la section 6.1.2, mais nous ne pouvons pas bien étudier les polynômes plus longs comme ceux présentés en annexe. Ce sont des informations inexploitées pour le moment.

Perspectives. Nous avons pris en compte les perturbations du vent dans notre modèle, mais la façon dont le vent est intégré est perfectible. Dans notre modèle, nous considérons que le vent est toujours face au drone. Une première amélioration serait de considérer une direction constante dans l'espace. Il suffirait pour cela de mettre à jour la variable `windAngle` au cours des simulations. Dans un deuxième temps, nous pourrions aussi considérer une direction de vent variable pendant le trajet. Pour cela, nous pourrions utiliser les mesures prises par la station au sol *(GCS : Ground Control Station)*. Cependant, il faut faire attention à ces mesures car les saisons auxquelles les tests sont effectués et celles auxquelles le spectacle a lieu sont différentes, et peuvent avoir des directions de vent différentes. Utiliser des mesures non adaptées pourrait avoir un effet inverse à celui attendu.

Une idée pour réduire le temps de calcul de notre modèle est de ne pas tenir compte de la proportion α lorsqu'elle est négligeable. Si la fréquence du filtre est suffisamment importante, et si le drone est initialement positionné dans la zone 1, il est impossible que les corrections imposées par le filtre emmènent le drone dans les zones 4 et 5 dès

le début de la simulation. En effet dans ce cas, les déviations imposées seront limitées par la proportion α présentée dans la section 5.5.

Il est ainsi possible de calculer au bout de combien d'itérations nous pouvons, au plus vite, atteindre les zones 4 et 5. On pourrait alors "accélérer" nos simulations en utilisant une abstraction pour remplacer les préfixes sûrs et se concentrer sur les itérations plus risquées ayant lieu en fin de parcours.

Nous avons utilisé actuellement la propriété pour détecter si le drone arrive dans les zones interdites. C'est juste une manière de trouver la défaillance de drone ; nous pouvons aussi tester d'autres propriétés intéressantes. Par exemple : i) estimer le nombre de fois où le drone entre dans une zone cible (indépendamment de notre structuration en 5 zones) ; ii) estimer le temps pendant lequel le drone peut accumuler de faibles défaillances sans possibilité de sortir de la zone de sécurité ; en effet ce temps serait un seuil critique, car au delà, le drone pourrait sortir de la zone de sécurité avec les défaillances accumulées.

Une autre piste pour améliorer notre modèle serait de mettre la fréquence du filtre en tant que paramètre du modèle. Cela permettrait d'obtenir un polynôme dont l'une des variables est la fréquence. Ainsi, PIXIEL pourrait étudier le meilleur compromis entre fréquence et précision, et donc choisir le filtre le plus adapté au plan de vol prévu. Cependant, ajouter la fréquence en paramètre est un processus difficile. En effet, la fréquence n'est pas un paramètre probabiliste de même nature que la précision : elle ne pourrait pas être intégrée au modèle en tant que probabilité de transition, et cela rendrait donc plus complexe l'étape de *model checking statistique paramétré*. Nous pensons néanmoins que cette modification pourrait revenir à l'ajout dans notre modèle d'une étape préalable aux calculs, et dans laquelle la valeur de la fréquence serait fixée (lors de l'initialisation).

Le PID est aussi un composant qui peut avoir un effet sur la trajectoire. En effet, le PID mesure aussi le bruit résiduel de la position corrigée par le filtre en fonction des valeurs précédentes reçues. Il utilise ensuite cette mesure pour corriger marginalement l'estimation de la position. Afin d'obtenir un modèle plus complet et plus précis, une autre possibilité serait d'intégrer dans notre modèle un modèle du PID et de ses effets sur la trajectoire. Le PID dépend des paramètres d'équations différentielles ; compléter

notre modèle de cette façon ajouterait donc des paramètres supplémentaires. Comme pour la fréquence du filtre, ces paramètres ne pourront pas être directement ajoutés comme des probabilités des transitions, mais demandent une adaptation plus complexe du modèle pour les y intégrer. Une étude rigoureuse sur le plan théorique est nécessaire ici avant la mise en œuvre.

Comme cela a été précisé dans le chapitre 5, nous avons choisi d'abstraire le filtre et les capteurs dans notre modèle (par les paramètres de précision) car les différents filtres existants ont des fonctionnements variés. Sur un plan plus technique, une piste pour la suite serait de créer une base de modèles de filtres permettant de représenter les différents filtres présents sur le marché. Ces modèles pourraient ensuite être intégrés, au choix, dans le modèle représentant le drone et sa trajectoire afin d'obtenir une analyse plus fine une fois le filtre choisi. Vu les technologies que nous avons utilisées, il suffirait alors de développer les différents modèles de filtres comme des classes Python héritant d'une classe abstraite commune, qui serait utilisée dans le modèle du drone (encodé en Python pour MCpMC).

Finalement, nous nous sommes principalement intéressés à l'impact des erreurs de positionnement sur la trajectoire du drone. Il serait néanmoins intéressant de coupler cette analyse avec les analyses de défaillance des différents composants matériels du drone afin d'obtenir une analyse plus complète de la sûreté des drones.

Notre travail ouvre ainsi des perspectives de R&D pour l'entreprise et certaines d'entre elles demandent un travail conséquent en amont.

BIBLIOGRAPHIE

[1] B. Scott Andersen and George Romanski. Verification of safety-critical software. *Queue*, 9(8) :50 :50–50 :59, August 2011.

[2] Z.-Q Huang, B.-F Xu, S.-L Kan, J Hu, and Z Chen. Survey on embedded software safety analysis standards, methods and tools for airborne system. *Ruan Jian Xue Bao/Journal of Software*, 25 :200–218, 02 2014.

[3] A. Aljaafreh and L. Dong. Ground vehicle classification based on hierarchical hidden Markov model and Gaussian mixture model using wireless sensor networks. In *2010 IEEE International Conference on Electro/Information Technology*, pages 1–4, May 2010.

[4] R. Telford and S. Galloway. Fault classification and diagnostic system for unmanned aerial vehicle electrical networks based on hidden markov models. *IET Electrical Systems in Transportation*, 5(3) :103–111, 2015.

[5] Laura R. Humphrey. *Model checking for verification in UAV cooperative control applications*, pages 69–117. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[6] M. Kwiatkowska, G. Norman, and D. Parker. PRISM : probabilistic model checking for performance and reliability analysis. *ACM SIGMETRICS Performance Evaluation Review*, 36(4) :40–45, 2009.

[7] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0 : probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[8] Karl J Astrom and Tore Hagglund. *Advanced PID control*. ISA, Advanced PID control, 2006. The book can be consulted by contacting : BE-ABP : Sosin, Mateusz.

[9] S. Sabatelli, M. Galgani, L. Fanucci, and A. Rocchi. A double-stage Kalman filter for orientation tracking with an integrated processor in 9-D IMU. *IEEE Transactions on Instrumentation and Measurement*, 62(3) :590–598, March 2013.

[10] Sebastian O. H. Madgwick. An efficient orientation filter for inertial and inertial / magnetic sensor arrays. 2010.

[11] H. G. de Marina, F. J. Pereda, J. M. Giron-Sierra, and F. Espinosa. UAV attitude estimation using unscented Kalman filter and TRIAD. *IEEE Transactions on Industrial Electronics*, 59(11) :4465–4474, Nov 2012.

[12] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel. A complementary filter for attitude estimation of a fixed-wing UAV. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 340–345, Sep. 2008.

[13] M. St-Pierre and D. Gingras. Comparison between the unscented kalman filter and the extended kalman filter for the position estimation module of an integrated navigation information system. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 831–835, June 2004.

[14] Stanislaw Konatowski, Piotr Kaniewski, and Jan Matuszewski. Comparison of estimation accuracy of ekf, ukf and pf filters. *Annual of Navigation*, 23, 12 2016.

[15] Gabriella Gigante and Domenico Pascarella. Formal methods in avionic software certification : the DO-178C perspective. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies*, pages 205–215, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[16] JianLin Wang and Wei Chen. A uav flight-control system failure mode, effect and criticality analysis. *Ship Electronic Engineering*, 33(4) :49–51, 2013.

[17] Koppány Máthé and Lucian Busoniu. Vision and control for UAVs : A survey of general methods and of inexpensive platforms for infrastructure inspection. *Sensors*, 15(7) :14887–14916, 2015.

[18] Anaïs Finzi, Bastien Tauran, Nicolas Chemarin, and Fabrice Frances. Contrôle de drones et robots par reconnaissance de mouvements complexes. MSR 2013 - Modélisation des Systèmes Réactifs, 2013.

[19] Sarantis Kyristsis, Angelos Antonopoulos, Theofilos Chanialakis, Emmanouel Stefanakis, Christos Linardos, Achilles Tripolitsiotis, and Panagiotis Partsinevelos. Towards autonomous modular UAV missions : the detection, geo-location and landing paradigm. *Sensors*, 16(11) :1844, 2016.

[20] Luis Felipe Gonzalez, Glen A. Montes, Eduard Puig, Sandra Johnson, Kerrie L. Mengersen, and Kevin J. Gaston. Unmanned aerial vehicles (uavs) and artificial intelligence revolutionizing wildlife monitoring and conservation. *Sensors*, 16(1) :97, 2016.

[21] Andres Hernandez, Harold Murcia, Cosmin Copot, and Robin De Keyser. Towards the development of a smart flying sensor : illustration in the field of precision agriculture. *Sensors*, 15(7) :16688–16709, 2015.

[22] Ben G. Fitzpatrick. *Recent advances in research on unmanned aerial vehicles*, pages 31–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[23] C. Goerzen, Zhaodan Kong, and Berenice Mettler. A survey of motion planning algorithms from the perspective of autonomous uav guidance. *Journal of Intelligent and Robotic Systems*, 57 :65–100, 11 2010.

[24] Navid Dadkhah and Bérénice Mettler. Survey of motion planning literature in the presence of uncertainty : Considerations for uav guidance. *J. Intell. Robotics Syst.*, 65(1-4) :233–246, January 2012.

[25] Y. Lin and S. Saripalli. Path planning using 3d dubins curve for unmanned aerial vehicles. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 296–304, May 2014.

[26] J. Bellingham, A. Richards, and J. P. How. Receding horizon control of autonomous aerial vehicles. In *Proceedings of the 2002 American control conference (IEEE Cat. No.CH37301)*, volume 5, pages 3741–3746 vol.5, May 2002.

[27] A. Freddi, S. Longhi, and A. Monteriù. A model-based fault diagnosis system for unmanned aerial vehicles. *IFAC Proceedings Volumes*, 42(8) :71 – 76, 2009. 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes.

[28] Guillermo Heredia, Fernando Caballero, Iván Maza, Luis Merino, Antidio Viguria, and Aníbal Ollero. Multi-Unmanned Aerial Vehicle (UAV) cooperative fault detection employing Differential Global Positioning (DGPS), inertial and vision sensors. *Sensors*, 9(9) :7566–7579, 2009.

[29] R.J. Patton and J. Chen. Observer-based fault detection and isolation : Robustness and applications. *Control Engineering Practice*, 5(5) :671 – 682, 1997.

[30] Marcello R. Napolitano, Younghwan An, and Brad A. Seanor. A fault tolerant flight control system for sensor and actuator failures using neural networks. *Aircraft Design*, 3(2) :103 – 128, 2000.

[31] Guillermo Heredia and Anibal Ollero. Detection of sensor faults in small helicopter UAVs using observer/Kalman filter identification. *Mathematical Problems in Engineering*, 2011, 09 2011.

[32] P. J. Pingree, E. Mikk, G. J. Holzmann, M. H. Smith, and D. Dams. Validation of mission critical software design and implementation using model checking [spacecraft]. In *Proceedings. The 21st Digital Avionics Systems Conference*, volume 1, pages 6A4–6A4, Oct 2002.

[33] Stanislav V. Emel'yanov, Dmitry Makarov, Aleksandr I. Panov, and Konstantin Yakovlev. Multilayer cognitive architecture for UAV control. *Cognitive Systems Research*, 39 :58–72, 2016.

[34] John S. Lai, Jason J. Ford, Peter J. O'Shea, Rodney A. Walker, and Michael Bosse. A study of morphological pre-processing approaches for track-before-detect dim target detection. In Jonghyuk Kim and Robert Mahony, editors, *2008 Australasian Conference on Robotics & Automation*, Canberra, 2008. Australian Robotics & Automation Association. The contents of this proceeding can be freely accessed online via the conference's web page (see hypertext link).

[35] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[36] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 592–601, 1993.

[37] Koushik Sen, Mahesh Viswanathan, and Gul Agha. On statistical model checking of stochastic systems. In *International Conference on Computer Aided Verification*, pages 266–280. Springer, 2005.

[38] Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical model checking : an overview. In *Proc. Runtime Verification - First International Conference, RV 2010, St. Julians, Malta, November 1-4, 2010. Proceedings*, volume 6418 of *Lecture Notes in Computer Science*, pages 122–135. Springer, 2010.

[39] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*, volume 10. John Wiley & Sons, 2016.

[40] Benoît Barbot, Serge Haddad, and Claudine Picaronny. Coupling and importance sampling for statistical model checking. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 331–346. Springer, 2012.

[41] Cyrille Jegourel, Axel Legay, and Sean Sedwards. Cross-entropy optimisation of importance sampling parameters for statistical model checking. In *International Conference on Computer Aided Verification*, pages 327–342. Springer, 2012.

[42] Luca Bortolussi, Dimitrios Milios, and Guido Sanguinetti. Smoothed model checking for uncertain continuous-time markov chains. *Information and Computation*, 247 :235–253, 2016.

[43] Benoit Delahaye, Paulin Fournier, and Didier Lime. Statistical model checking for parameterized models. working paper or preprint, February 2019.

[44] M. Kwiatkowska, G. Norman, and D. Parker. Analysis of a gossip protocol in prism. *ACM SIGMETRICS Performance Evaluation Review*, 36(3) :17–22, 2008.

[45] D. Knuth and A. Yao. *Algorithms and complexity : new directions and recent results*, chapter The complexity of nonuniform random number generation. Academic Press, 1976.

[46] D. Chaum. The dining cryptographers problem : unconditional sender and recipient untraceability. *Journal of Cryptology*, 1 :65–75, 1988.

[47] N. Barkai and S. Leibler. Biological rhythms : circadian clocks limited by noise. *Nature*, 403 :267–268, 2000.

[48] H. Younes, M. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking : an empirical study. In K. Jensen and A. Podelski, editors, *Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, volume 2988 of *LNCS*, pages 46–60. Springer, 2004.

[49] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. Automatic verification of competitive stochastic systems. *Formal Methods in System Design*, 43(1) :61–92, 2013.

[50] Ernst Moritz Hahn, Holger Hermanns, Björn Wachter, and Lijun Zhang. PARAM : a model checker for parametric Markov models. In Tayssir Touili, Byron Cook, and Paul Jackson, editors, *Computer Aided Verification*, pages 660–664, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[51] Olivier Cardin. *Contribution of online simulation to production activity control decision support – Application to a flexible manufacturing system*. Theses, Université de Nantes, October 2007.

[52] M. Hohenwarter, M. Borcherds, G. Ancsin, B. Bencze, M. Blossier, A. Delobelle, C. Denizet, J. Éliás, Á Fekete, L. Gál, Z. Konečný, Z. Kovács, S. Lizelfelner, B. Parisse, and G. Sturr. GeoGebra 4.4. Master's thesis, December 2013. `http://www.geogebra.org`.

[53] Koushik Sen, Mahesh Viswanathan, and Gul Agha. Statistical model checking of black-box probabilistic systems. In *International Conference on Computer Aided Verification*, pages 202–215. Springer, 2004.

[54] Przemyslaw Gasior, Adam Bondyra, and Stanislaw Gardecki. Development of vertical movement controller for multirotor UAVs. In Szewczyk et al. [56], pages 339–348.

[55] Sheldon M. Ross. *A first course in probability*. 2009.

[56] Roman Szewczyk, Cezary Zielinski, and Malgorzata Kaliczynska, editors. *Automation 2017 - innovations in automation, robotics and measurement Techniques, Proceedings of AUTOMATION 2017, March 15-17, 2017, Warsaw, Poland*, volume 550 of *Advances in Intelligent Systems and Computing*. Springer, 2017.

[57] Mónica F. Bugallo, Shanshan Xu, and Petar M. Djuric. Performance comparison of EKF and particle filtering methods for maneuvering targets. *Digital Signal Processing*, 17(4) :774–786, 2007.

[58] Michal R. Nowicki, Jan Wietrzykowski, and Piotr Skrzypczynski. Simplicity or flexibility ? complementary filter vs. EKF for orientation estimation on mobile devices. In *2nd IEEE International Conference on Cybernetics, CYBCONF 2015, Gdynia, Poland, June 24-26, 2015*, pages 166–171, 2015.

[59] Chaoshu Jiang, Hongbin Li, and Muralidhar Rangaswamy. On the conjugate gradient matched filter. *IEEE Trans. Signal Processing*, 60(5) :2660–2666, 2012.

[60] Young Hwan Chang, Qie Hu, and Claire J. Tomlin. Secure estimation based Kalman Filter for cyber-physical systems against sensor attacks. *Automatica*, 95 :399–412, 2018.

[61] Zhengyuan Zhou, Jerry Ding, Haomiao Huang, Ryo Takei, and Claire Tomlin. Efficient path planning algorithms in reach-avoid problems. *Automatica*, 89 :28–36, 2018.

CHAPITRE 8

ANNEXE

8.1 Correspondance entre versions du modèle et fichiers sources

TABLE 8.1 – Présentation synthétique des différentes versions du modèle

Version	**Nom du fichier**	**Incréments des version successives**
Version 1	version1simple.pm	Version de base.
Version 2	version2complet.pm version2.props	Prise en compte de la précision du filtre.
Version 3 en PRISM en MCpMC	 modelAB6.pm drone2.py	Prise en compte de la position précédente du drone. Prise en compte des coordonnées exactes.
Version 4	drone4.py	Perturbations sur l'axe x.
Version 5 Version 6 Version 7	 drone4use.py drone4vent.py	Changement possible d'objectif. Perturbations liées au vent (constantes). Perturbations liées au vent (paramètres).
Version 8	drone8CompletX.py	Changement de repère. Trajectoires longues (> 3 points).

8.2 Polynômes résultant de l'analyse du modèle (version 3) avec MCpMC

8.2.1 Polynôme résultat pour 100k simulations (pour $T_{chemin} = 5$ et $T_{repondre} = 1$)

$1.3125 * ProbaFilter1^4 * ProbaFilter4$
$+ 1.125 * ProbaFilter1^4 * ProbaFilter5$
$+ 4.5 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter4$
$+ 3.75 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter5$
$+ 0.0625 * ProbaFilter1^3 * ProbaFilter3^2$
$+ 4.3125 * ProbaFilter1^3 * ProbaFilter3 * ProbaFilter4$
$+ 3.875 * ProbaFilter1^3 * ProbaFilter3 * ProbaFilter5$
$+ 4.40625 * ProbaFilter1^3 * ProbaFilter4^2$
$+ 7.9375 * ProbaFilter1^3 * ProbaFilter4 * ProbaFilter5$
$+ 4.09375 * ProbaFilter1^3 * ProbaFilter5^2$
$+ 0.03125 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter3$
$+ 6.34375 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter4$
$+ 6.21875 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter5$
$+ 0.03125 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3^2$
$+ 13.09375 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4$
$+ 12.4375 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5$
$+ 13.625 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter4^2$
$+ 23.6875 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5$
$+ 11.8125 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter5^2$
$+ 6.625 * ProbaFilter1^2 * ProbaFilter3^2 * ProbaFilter4$
$+ 6.84375 * ProbaFilter1^2 * ProbaFilter3^2 * ProbaFilter5$
$+ 12.21875 * ProbaFilter1^2 * ProbaFilter3 * ProbaFilter4^2$
$+ 23.625 * ProbaFilter1^2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5$
$+ 12.125 * ProbaFilter1^2 * ProbaFilter3 * ProbaFilter5^2$
$+ 5.5625 * ProbaFilter1^2 * ProbaFilter4^3$
$+ 18.15625 * ProbaFilter1^2 * ProbaFilter4^2 * ProbaFilter5$
$+ 17.40625 * ProbaFilter1^2 * ProbaFilter4 * ProbaFilter5^2$
$+ 0.08125 * ProbaFilter1^2 * ProbaFilter4 * ProbaFilter5$

$$
\begin{aligned}
&+ 5.8125 * ProbaFilter1^2 * ProbaFilter5^3 \\
&+ 0.2625 * ProbaFilter1^2 * ProbaFilter5^2 \\
&+ 0.03125 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter3 \\
&+ 4.15625 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter4 \\
&+ 4.125 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter5 \\
&+ 0.09375 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3^2 \\
&+ 12.0625 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4 \\
&+ 11.46875 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter5 \\
&+ 12.09375 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter4^2 \\
&+ 22.0625 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter4 * ProbaFilter5 \\
&+ 12.15625 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter5^2 \\
&+ 12.34375 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4 \\
&+ 12.875 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter5 \\
&+ 23.9375 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^2 \\
&+ 49.21875 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5 \\
&+ 23.375 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5^2 \\
&+ 0.0625 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5 \\
&+ 12.0 * ProbaFilter1 * ProbaFilter2 * ProbaFilter4^3 \\
&+ 37.6875 * ProbaFilter1 * ProbaFilter2 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 33.71875 * ProbaFilter1 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 0.20625 * ProbaFilter1 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5 \\
&+ 10.21875 * ProbaFilter1 * ProbaFilter2 * ProbaFilter5^3 \\
&+ 0.5875 * ProbaFilter1 * ProbaFilter2 * ProbaFilter5^2 \\
&+ 0.03125 * ProbaFilter1 * ProbaFilter3^4 \\
&+ 4.0625 * ProbaFilter1 * ProbaFilter3^3 * ProbaFilter4 \\
&+ 4.34375 * ProbaFilter1 * ProbaFilter3^3 * ProbaFilter5 \\
&+ 12.34375 * ProbaFilter1 * ProbaFilter3^2 * ProbaFilter4^2 \\
&+ 24.5625 * ProbaFilter1 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5 \\
&+ 11.53125 * ProbaFilter1 * ProbaFilter3^2 * ProbaFilter5^2 \\
&+ 0.10625 * ProbaFilter1 * ProbaFilter3^2 * ProbaFilter5 \\
&+ 12.25 * ProbaFilter1 * ProbaFilter3 * ProbaFilter4^3 \\
&+ 36.875 * ProbaFilter1 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 34.0625 * ProbaFilter1 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 0.46875 * ProbaFilter1 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5
\end{aligned}
$$

$$+ 10.125 * ProbaFilter1 * ProbaFilter3 * ProbaFilter5^3$$
$$+ 0.78125 * ProbaFilter1 * ProbaFilter3 * ProbaFilter5^2$$
$$+ 4.5625 * ProbaFilter1 * ProbaFilter4^4$$
$$+ 15.8125 * ProbaFilter1 * ProbaFilter4^3 * ProbaFilter5$$
$$+ 23.09375 * ProbaFilter1 * ProbaFilter4^2 * ProbaFilter5^2$$
$$+ 0.35625 * ProbaFilter1 * ProbaFilter4^2 * ProbaFilter5$$
$$+ 13.5 * ProbaFilter1 * ProbaFilter4 * ProbaFilter5^3$$
$$+ 0.9375 * ProbaFilter1 * ProbaFilter4 * ProbaFilter5^2$$
$$+ 3.09375 * ProbaFilter1 * ProbaFilter5^4$$
$$+ 0.46875 * ProbaFilter1 * ProbaFilter5^3$$
$$+ 1.09375 * ProbaFilter2^4 * ProbaFilter4$$
$$+ 0.90625 * ProbaFilter2^4 * ProbaFilter5$$
$$+ 4.21875 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter4$$
$$+ 4.15625 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter5$$
$$+ 4.84375 * ProbaFilter2^3 * ProbaFilter4^2$$
$$+ 7.625 * ProbaFilter2^3 * ProbaFilter4 * ProbaFilter5$$
$$+ 3.90625 * ProbaFilter2^3 * ProbaFilter5^2$$
$$+ 5.53125 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter4$$
$$+ 5.375 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter5$$
$$+ 11.84375 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4^2$$
$$+ 22.78125 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5$$
$$+ 10.96875 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter5^2$$
$$+ 0.04375 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter5$$
$$+ 5.84375 * ProbaFilter2^2 * ProbaFilter4^3$$
$$+ 16.46875 * ProbaFilter2^2 * ProbaFilter4^2 * ProbaFilter5$$
$$+ 17.4375 * ProbaFilter2^2 * ProbaFilter4 * ProbaFilter5^2$$
$$+ 0.14375 * ProbaFilter2^2 * ProbaFilter4 * ProbaFilter5$$
$$+ 5.5 * ProbaFilter2^2 * ProbaFilter5^3$$
$$+ 0.31875 * ProbaFilter2^2 * ProbaFilter5^2$$
$$+ 4.03125 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter4$$
$$+ 4.1875 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter5$$
$$+ 12.125 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4^2$$
$$+ 22.59375 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5$$
$$+ 11.59375 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter5^2$$

$$
\begin{aligned}
&+ 0.09375 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter5 \\
&+ 11.75 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^3 \\
&+ 36.75 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 31.375 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 0.5375 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5 \\
&+ 10.40625 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5^3 \\
&+ 0.81875 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5^2 \\
&+ 4.4375 * ProbaFilter2 * ProbaFilter4^4 \\
&+ 16.40625 * ProbaFilter2 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 21.375 * ProbaFilter2 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 0.55625 * ProbaFilter2 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 14.0625 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5^3 \\
&+ 1.05 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 3.4375 * ProbaFilter2 * ProbaFilter5^4 \\
&+ 0.4875 * ProbaFilter2 * ProbaFilter5^3 \\
&+ 0.75 * ProbaFilter3^4 * ProbaFilter4 \\
&+ 1.09375 * ProbaFilter3^4 * ProbaFilter5 \\
&+ 4.3125 * ProbaFilter3^3 * ProbaFilter4^2 \\
&+ 6.9375 * ProbaFilter3^3 * ProbaFilter4 * ProbaFilter5 \\
&+ 3.71875 * ProbaFilter3^3 * ProbaFilter5^2 \\
&+ 0.06875 * ProbaFilter3^3 * ProbaFilter5 \\
&+ 5.78125 * ProbaFilter3^2 * ProbaFilter4^3 \\
&+ 18.0 * ProbaFilter3^2 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 15.53125 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 0.49375 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5 \\
&+ 5.21875 * ProbaFilter3^2 * ProbaFilter5^3 \\
&+ 0.575 * ProbaFilter3^2 * ProbaFilter5^2 \\
&+ 3.84375 * ProbaFilter3 * ProbaFilter4^4 \\
&+ 14.59375 * ProbaFilter3 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 0.00625 * ProbaFilter3 * ProbaFilter4^3 \\
&+ 20.5625 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 0.49375 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 12.875 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^3 \\
&+ 1.225 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^2
\end{aligned}
$$

$+ 2.875 * ProbaFilter3 * ProbaFilter5^4$
$+ 0.69375 * ProbaFilter3 * ProbaFilter5^3$
$+ 1.03125 * ProbaFilter4^5$
$+ 5.28125 * ProbaFilter4^4 * ProbaFilter5$
$+ 0.025 * ProbaFilter4^4$
$+ 10.46875 * ProbaFilter4^3 * ProbaFilter5^2$
$+ 0.30625 * ProbaFilter4^3 * ProbaFilter5$
$+ 7.1875 * ProbaFilter4^2 * ProbaFilter5^3$
$+ 0.70625 * ProbaFilter4^2 * ProbaFilter5^2$
$+ 4.21875 * ProbaFilter4 * ProbaFilter5^4$
$+ 0.775 * ProbaFilter4 * ProbaFilter5^3$
$+ 0.5 * ProbaFilter5^5$
$+ 0.225 * ProbaFilter5^4$

8.2.2 Polynôme résultat pour 100k simulations (pour $T_{chemin} = 7$ et $T_{repondre} = 1$)

$8.59375 * ProbaFilter1^5 * ProbaFilter2 * ProbaFilter4$
$+ 4.6875 * ProbaFilter1^5 * ProbaFilter2 * ProbaFilter5$
$+ 9.375 * ProbaFilter1^5 * ProbaFilter3 * ProbaFilter4$
$+ 5.46875 * ProbaFilter1^5 * ProbaFilter3 * ProbaFilter5$
$+ 9.375 * ProbaFilter1^5 * ProbaFilter4^2$
$+ 10.15625 * ProbaFilter1^5 * ProbaFilter4 * ProbaFilter5$
$+ 5.46875 * ProbaFilter1^5 * ProbaFilter5^2$
$+ 17.1875 * ProbaFilter1^4 * ProbaFilter2^2 * ProbaFilter4$
$+ 14.84375 * ProbaFilter1^4 * ProbaFilter2^2 * ProbaFilter5$
$+ 31.25 * ProbaFilter1^4 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4$
$+ 22.65625 * ProbaFilter1^4 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5$
$+ 32.03125 * ProbaFilter1^4 * ProbaFilter2 * ProbaFilter4^2$
$+ 71.875 * ProbaFilter1^4 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5$
$+ 31.25 * ProbaFilter1^4 * ProbaFilter2 * ProbaFilter5^2$
$+ 11.71875 * ProbaFilter1^4 * ProbaFilter3^2 * ProbaFilter4$
$+ 18.75 * ProbaFilter1^4 * ProbaFilter3^2 * ProbaFilter5$
$+ 37.5 * ProbaFilter1^4 * ProbaFilter3 * ProbaFilter4^2$

$$
\begin{aligned}
&+ 64.0625 * ProbaFilter1^4 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5 \\
&+ 19.53125 * ProbaFilter1^4 * ProbaFilter3 * ProbaFilter5^2 \\
&+ 16.40625 * ProbaFilter1^4 * ProbaFilter4^3 \\
&+ 45.3125 * ProbaFilter1^4 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 38.28125 * ProbaFilter1^4 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 14.84375 * ProbaFilter1^4 * ProbaFilter5^3 \\
&+ 0.15625 * ProbaFilter1^4 * ProbaFilter5^2 \\
&+ 28.90625 * ProbaFilter1^3 * ProbaFilter2^3 * ProbaFilter4 \\
&+ 20.3125 * ProbaFilter1^3 * ProbaFilter2^3 * ProbaFilter5 \\
&+ 61.71875 * ProbaFilter1^3 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4 \\
&+ 57.8125 * ProbaFilter1^3 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter5 \\
&+ 54.6875 * ProbaFilter1^3 * ProbaFilter2^2 * ProbaFilter4^2 \\
&+ 114.84375 * ProbaFilter1^3 * ProbaFilter2^2 * ProbaFilter4 * ProbaFilter5 \\
&+ 73.4375 * ProbaFilter1^3 * ProbaFilter2^2 * ProbaFilter5^2 \\
&+ 68.75 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4 \\
&+ 64.0625 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter5 \\
&+ 122.65625 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^2 \\
&+ 239.0625 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5 \\
&+ 135.9375 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5^2 \\
&+ 0.15625 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5 \\
&+ 62.5 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter4^3 \\
&+ 202.34375 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 165.625 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 0.46875 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5 \\
&+ 70.3125 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter5^3 \\
&+ 0.46875 * ProbaFilter1^3 * ProbaFilter2 * ProbaFilter5^2 \\
&+ 18.75 * ProbaFilter1^3 * ProbaFilter3^3 * ProbaFilter4 \\
&+ 23.4375 * ProbaFilter1^3 * ProbaFilter3^3 * ProbaFilter5 \\
&+ 58.59375 * ProbaFilter1^3 * ProbaFilter3^2 * ProbaFilter4^2 \\
&+ 125.0 * ProbaFilter1^3 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5 \\
&+ 66.40625 * ProbaFilter1^3 * ProbaFilter3^2 * ProbaFilter5^2 \\
&+ 71.875 * ProbaFilter1^3 * ProbaFilter3 * ProbaFilter4^3 \\
&+ 205.46875 * ProbaFilter1^3 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 196.09375 * ProbaFilter1^3 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^2
\end{aligned}
$$

$$
\begin{aligned}
&+ 0.625 * ProbaFilter1^3 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5 \\
&+ 52.34375 * ProbaFilter1^3 * ProbaFilter3 * ProbaFilter5^3 \\
&+ 1.5625 * ProbaFilter1^3 * ProbaFilter3 * ProbaFilter5^2 \\
&+ 18.75 * ProbaFilter1^3 * ProbaFilter4^4 \\
&+ 80.46875 * ProbaFilter1^3 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 125.78125 * ProbaFilter1^3 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 0.46875 * ProbaFilter1^3 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 71.09375 * ProbaFilter1^3 * ProbaFilter4 * ProbaFilter5^3 \\
&+ 1.875 * ProbaFilter1^3 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 17.1875 * ProbaFilter1^3 * ProbaFilter5^4 \\
&+ 0.3125 * ProbaFilter1^3 * ProbaFilter5^3 \\
&+ 16.40625 * ProbaFilter1^2 * ProbaFilter2^4 * ProbaFilter4 \\
&+ 9.375 * ProbaFilter1^2 * ProbaFilter2^4 * ProbaFilter5 \\
&+ 1.5625 * ProbaFilter1^2 * ProbaFilter2^3 * ProbaFilter3^2 \\
&+ 59.375 * ProbaFilter1^2 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter4 \\
&+ 44.53125 * ProbaFilter1^2 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter5 \\
&+ 60.15625 * ProbaFilter1^2 * ProbaFilter2^3 * ProbaFilter4^2 \\
&+ 121.875 * ProbaFilter1^2 * ProbaFilter2^3 * ProbaFilter4 * ProbaFilter5 \\
&+ 60.9375 * ProbaFilter1^2 * ProbaFilter2^3 * ProbaFilter5^2 \\
&+ 84.375 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter4 \\
&+ 100.78125 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter5 \\
&+ 200.0 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4^2 \\
&+ 392.96875 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5 \\
&+ 182.8125 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter5^2 \\
&+ 113.28125 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter4^3 \\
&+ 285.9375 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 258.59375 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 1.71875 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter4 * ProbaFilter5 \\
&+ 82.8125 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter5^3 \\
&+ 1.875 * ProbaFilter1^2 * ProbaFilter2^2 * ProbaFilter5^2 \\
&+ 0.78125 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3^4 \\
&+ 66.40625 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter4 \\
&+ 74.21875 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter5 \\
&+ 188.28125 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4^2
\end{aligned}
$$

$$
\begin{aligned}
&+ 370.3125 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5 \\
&+ 172.65625 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter5^2 \\
&+ 0.46875 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter5 \\
&+ 185.15625 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^3 \\
&+ 554.6875 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 508.59375 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 2.34375 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5 \\
&+ 169.53125 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5^3 \\
&+ 4.375 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5^2 \\
&+ 56.25 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter4^4 \\
&+ 218.75 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 342.1875 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 2.96875 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 223.4375 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5^3 \\
&+ 5.46875 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 53.90625 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter5^4 \\
&+ 4.6875 * ProbaFilter1^2 * ProbaFilter2 * ProbaFilter5^3 \\
&+ 13.28125 * ProbaFilter1^2 * ProbaFilter3^4 * ProbaFilter4 \\
&+ 8.59375 * ProbaFilter1^2 * ProbaFilter3^4 * ProbaFilter5 \\
&+ 59.375 * ProbaFilter1^2 * ProbaFilter3^3 * ProbaFilter4^2 \\
&+ 115.625 * ProbaFilter1^2 * ProbaFilter3^3 * ProbaFilter4 * ProbaFilter5 \\
&+ 45.3125 * ProbaFilter1^2 * ProbaFilter3^3 * ProbaFilter5^2 \\
&+ 0.3125 * ProbaFilter1^2 * ProbaFilter3^3 * ProbaFilter5 \\
&+ 100.78125 * ProbaFilter1^2 * ProbaFilter3^2 * ProbaFilter4^3 \\
&+ 285.15625 * ProbaFilter1^2 * ProbaFilter3^2 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 255.46875 * ProbaFilter1^2 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 2.03125 * ProbaFilter1^2 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5 \\
&+ 100.78125 * ProbaFilter1^2 * ProbaFilter3^2 * ProbaFilter5^3 \\
&+ 2.65625 * ProbaFilter1^2 * ProbaFilter3^2 * ProbaFilter5^2 \\
&+ 64.84375 * ProbaFilter1^2 * ProbaFilter3 * ProbaFilter4^4 \\
&+ 242.1875 * ProbaFilter1^2 * ProbaFilter3 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 335.9375 * ProbaFilter1^2 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 2.5 * ProbaFilter1^2 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 250.0 * ProbaFilter1^2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^3
\end{aligned}
$$

$$+ 9.375 * ProbaFilter1^2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^2$$
$$+ 59.375 * ProbaFilter1^2 * ProbaFilter3 * ProbaFilter5^4$$
$$+ 5.15625 * ProbaFilter1^2 * ProbaFilter3 * ProbaFilter5^3$$
$$+ 11.71875 * ProbaFilter1^2 * ProbaFilter4^5$$
$$+ 74.21875 * ProbaFilter1^2 * ProbaFilter4^4 * ProbaFilter5$$
$$+ 153.90625 * ProbaFilter1^2 * ProbaFilter4^3 * ProbaFilter5^2$$
$$+ 1.09375 * ProbaFilter1^2 * ProbaFilter4^3 * ProbaFilter5$$
$$+ 139.0625 * ProbaFilter1^2 * ProbaFilter4^2 * ProbaFilter5^3$$
$$+ 4.53125 * ProbaFilter1^2 * ProbaFilter4^2 * ProbaFilter5^2$$
$$+ 76.5625 * ProbaFilter1^2 * ProbaFilter4 * ProbaFilter5^4$$
$$+ 4.84375 * ProbaFilter1^2 * ProbaFilter4 * ProbaFilter5^3$$
$$+ 11.71875 * ProbaFilter1^2 * ProbaFilter5^5$$
$$+ 2.65625 * ProbaFilter1^2 * ProbaFilter5^4$$
$$+ 9.375 * ProbaFilter1 * ProbaFilter2^5 * ProbaFilter4$$
$$+ 4.6875 * ProbaFilter1 * ProbaFilter2^5 * ProbaFilter5$$
$$+ 32.03125 * ProbaFilter1 * ProbaFilter2^4 * ProbaFilter3 * ProbaFilter4$$
$$+ 32.8125 * ProbaFilter1 * ProbaFilter2^4 * ProbaFilter3 * ProbaFilter5$$
$$+ 28.125 * ProbaFilter1 * ProbaFilter2^4 * ProbaFilter4^2$$
$$+ 65.625 * ProbaFilter1 * ProbaFilter2^4 * ProbaFilter4 * ProbaFilter5$$
$$+ 36.71875 * ProbaFilter1 * ProbaFilter2^4 * ProbaFilter5^2$$
$$+ 63.28125 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter3^2 * ProbaFilter4$$
$$+ 64.0625 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter3^2 * ProbaFilter5$$
$$+ 143.75 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter4^2$$
$$+ 242.96875 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5$$
$$+ 123.4375 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter5^2$$
$$+ 0.15625 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter5$$
$$+ 63.28125 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter4^3$$
$$+ 174.21875 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter4^2 * ProbaFilter5$$
$$+ 157.03125 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter4 * ProbaFilter5^2$$
$$+ 0.3125 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter4 * ProbaFilter5$$
$$+ 52.34375 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter5^3$$
$$+ 1.09375 * ProbaFilter1 * ProbaFilter2^3 * ProbaFilter5^2$$
$$+ 64.0625 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3^3 * ProbaFilter4$$
$$+ 75.78125 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3^3 * ProbaFilter5$$

$$
\begin{aligned}
&+ 167.96875 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter4^2 \\
&+ 387.5 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5 \\
&+ 164.0625 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter5^2 \\
&+ 1.25 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter5 \\
&+ 155.46875 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4^3 \\
&+ 561.71875 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 546.09375 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 2.65625 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5 \\
&+ 175.0 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter5^3 \\
&+ 4.6875 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter5^2 \\
&+ 58.59375 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter4^4 \\
&+ 237.5 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 341.40625 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 2.03125 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 225.0 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter4 * ProbaFilter5^3 \\
&+ 5.9375 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 52.34375 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter5^4 \\
&+ 3.75 * ProbaFilter1 * ProbaFilter2^2 * ProbaFilter5^3 \\
&+ 31.25 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^4 * ProbaFilter4 \\
&+ 37.5 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^4 * ProbaFilter5 \\
&+ 123.4375 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter4^2 \\
&+ 248.4375 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter4 * ProbaFilter5 \\
&+ 99.21875 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter5^2 \\
&+ 0.3125 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter5 \\
&+ 178.90625 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4^3 \\
&+ 565.625 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 518.75 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 4.0625 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5 \\
&+ 154.6875 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter5^3 \\
&+ 5.9375 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter5^2 \\
&+ 118.75 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^4 \\
&+ 495.3125 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 650.78125 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 6.40625 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5
\end{aligned}
$$

$$+ 424.21875 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^3$$
$$+ 15.0 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^2$$
$$+ 97.65625 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5^4$$
$$+ 9.375 * ProbaFilter1 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5^3$$
$$+ 26.5625 * ProbaFilter1 * ProbaFilter2 * ProbaFilter4^5$$
$$+ 149.21875 * ProbaFilter1 * ProbaFilter2 * ProbaFilter4^4 * ProbaFilter5$$
$$+ 288.28125 * ProbaFilter1 * ProbaFilter2 * ProbaFilter4^3 * ProbaFilter5^2$$
$$+ 2.8125 * ProbaFilter1 * ProbaFilter2 * ProbaFilter4^3 * ProbaFilter5$$
$$+ 255.46875 * ProbaFilter1 * ProbaFilter2 * ProbaFilter4^2 * ProbaFilter5^3$$
$$+ 10.3125 * ProbaFilter1 * ProbaFilter2 * ProbaFilter4^2 * ProbaFilter5^2$$
$$+ 126.5625 * ProbaFilter1 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5^4$$
$$+ 9.375 * ProbaFilter1 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5^3$$
$$+ 25.0 * ProbaFilter1 * ProbaFilter2 * ProbaFilter5^5$$
$$+ 4.0625 * ProbaFilter1 * ProbaFilter2 * ProbaFilter5^4$$
$$+ 0.03125 * ProbaFilter1 * ProbaFilter2 * ProbaFilter5^3$$
$$+ 5.46875 * ProbaFilter1 * ProbaFilter3^5 * ProbaFilter4$$
$$+ 6.25 * ProbaFilter1 * ProbaFilter3^5 * ProbaFilter5$$
$$+ 33.59375 * ProbaFilter1 * ProbaFilter3^4 * ProbaFilter4^2$$
$$+ 60.15625 * ProbaFilter1 * ProbaFilter3^4 * ProbaFilter4 * ProbaFilter5$$
$$+ 30.46875 * ProbaFilter1 * ProbaFilter3^4 * ProbaFilter5^2$$
$$+ 70.3125 * ProbaFilter1 * ProbaFilter3^3 * ProbaFilter4^3$$
$$+ 187.5 * ProbaFilter1 * ProbaFilter3^3 * ProbaFilter4^2 * ProbaFilter5$$
$$+ 158.59375 * ProbaFilter1 * ProbaFilter3^3 * ProbaFilter4 * ProbaFilter5^2$$
$$+ 1.71875 * ProbaFilter1 * ProbaFilter3^3 * ProbaFilter4 * ProbaFilter5$$
$$+ 51.5625 * ProbaFilter1 * ProbaFilter3^3 * ProbaFilter5^3$$
$$+ 2.5 * ProbaFilter1 * ProbaFilter3^3 * ProbaFilter5^2$$
$$+ 54.6875 * ProbaFilter1 * ProbaFilter3^2 * ProbaFilter4^4$$
$$+ 243.75 * ProbaFilter1 * ProbaFilter3^2 * ProbaFilter4^3 * ProbaFilter5$$
$$+ 351.5625 * ProbaFilter1 * ProbaFilter3^2 * ProbaFilter4^2 * ProbaFilter5^2$$
$$+ 4.21875 * ProbaFilter1 * ProbaFilter3^2 * ProbaFilter4^2 * ProbaFilter5$$
$$+ 220.3125 * ProbaFilter1 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5^3$$
$$+ 8.4375 * ProbaFilter1 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5^2$$
$$+ 50.0 * ProbaFilter1 * ProbaFilter3^2 * ProbaFilter5^4$$
$$+ 4.6875 * ProbaFilter1 * ProbaFilter3^2 * ProbaFilter5^3$$

$$
\begin{aligned}
&+ 22.65625 * ProbaFilter1 * ProbaFilter3 * ProbaFilter4^5 \\
&+ 148.4375 * ProbaFilter1 * ProbaFilter3 * ProbaFilter4^4 * ProbaFilter5 \\
&+ 0.15625 * ProbaFilter1 * ProbaFilter3 * ProbaFilter4^4 \\
&+ 273.4375 * ProbaFilter1 * ProbaFilter3 * ProbaFilter4^3 * ProbaFilter5^2 \\
&+ 3.59375 * ProbaFilter1 * ProbaFilter3 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 270.3125 * ProbaFilter1 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5^3 \\
&+ 12.03125 * ProbaFilter1 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 121.09375 * ProbaFilter1 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^4 \\
&+ 10.78125 * ProbaFilter1 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^3 \\
&+ 18.75 * ProbaFilter1 * ProbaFilter3 * ProbaFilter5^5 \\
&+ 4.84375 * ProbaFilter1 * ProbaFilter3 * ProbaFilter5^4 \\
&+ 2.34375 * ProbaFilter1 * ProbaFilter4^6 \\
&+ 31.25 * ProbaFilter1 * ProbaFilter4^5 * ProbaFilter5 \\
&+ 89.84375 * ProbaFilter1 * ProbaFilter4^4 * ProbaFilter5^2 \\
&+ 1.09375 * ProbaFilter1 * ProbaFilter4^4 * ProbaFilter5 \\
&+ 101.5625 * ProbaFilter1 * ProbaFilter4^3 * ProbaFilter5^3 \\
&+ 5.78125 * ProbaFilter1 * ProbaFilter4^3 * ProbaFilter5^2 \\
&+ 69.53125 * ProbaFilter1 * ProbaFilter4^2 * ProbaFilter5^4 \\
&+ 6.40625 * ProbaFilter1 * ProbaFilter4^2 * ProbaFilter5^3 \\
&+ 25.0 * ProbaFilter1 * ProbaFilter4 * ProbaFilter5^5 \\
&+ 3.75 * ProbaFilter1 * ProbaFilter4 * ProbaFilter5^4 \\
&+ 0.25 * ProbaFilter1 * ProbaFilter4 * ProbaFilter5^3 \\
&+ 3.125 * ProbaFilter1 * ProbaFilter5^6 \\
&+ 0.15625 * ProbaFilter1 * ProbaFilter5^5 \\
&+ 0.0625 * ProbaFilter1 * ProbaFilter5^4 \\
&+ 0.78125 * ProbaFilter2^6 * ProbaFilter4 \\
&+ 1.5625 * ProbaFilter2^6 * ProbaFilter5 \\
&+ 2.34375 * ProbaFilter2^5 * ProbaFilter3 * ProbaFilter4 \\
&+ 5.46875 * ProbaFilter2^5 * ProbaFilter3 * ProbaFilter5 \\
&+ 4.6875 * ProbaFilter2^5 * ProbaFilter4^2 \\
&+ 17.1875 * ProbaFilter2^5 * ProbaFilter4 * ProbaFilter5 \\
&+ 10.9375 * ProbaFilter2^5 * ProbaFilter5^2 \\
&+ 14.84375 * ProbaFilter2^4 * ProbaFilter3^2 * ProbaFilter4 \\
&+ 12.5 * ProbaFilter2^4 * ProbaFilter3^2 * ProbaFilter5
\end{aligned}
$$

$$+ 32.8125 * ProbaFilter2^4 * ProbaFilter3 * ProbaFilter4^2$$
$$+ 61.71875 * ProbaFilter2^4 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5$$
$$+ 32.03125 * ProbaFilter2^4 * ProbaFilter3 * ProbaFilter5^2$$
$$+ 12.5 * ProbaFilter2^4 * ProbaFilter4^3$$
$$+ 42.1875 * ProbaFilter2^4 * ProbaFilter4^2 * ProbaFilter5$$
$$+ 42.96875 * ProbaFilter2^4 * ProbaFilter4 * ProbaFilter5^2$$
$$+ 8.59375 * ProbaFilter2^4 * ProbaFilter5^3$$
$$+ 0.46875 * ProbaFilter2^4 * ProbaFilter5^2$$
$$+ 20.3125 * ProbaFilter2^3 * ProbaFilter3^3 * ProbaFilter4$$
$$+ 18.75 * ProbaFilter2^3 * ProbaFilter3^3 * ProbaFilter5$$
$$+ 67.1875 * ProbaFilter2^3 * ProbaFilter3^2 * ProbaFilter4^2$$
$$+ 123.4375 * ProbaFilter2^3 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5$$
$$+ 50.78125 * ProbaFilter2^3 * ProbaFilter3^2 * ProbaFilter5^2$$
$$+ 0.3125 * ProbaFilter2^3 * ProbaFilter3^2 * ProbaFilter5$$
$$+ 69.53125 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter4^3$$
$$+ 183.59375 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5$$
$$+ 170.3125 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^2$$
$$+ 0.78125 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5$$
$$+ 66.40625 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter5^3$$
$$+ 2.8125 * ProbaFilter2^3 * ProbaFilter3 * ProbaFilter5^2$$
$$+ 14.84375 * ProbaFilter2^3 * ProbaFilter4^4$$
$$+ 79.6875 * ProbaFilter2^3 * ProbaFilter4^3 * ProbaFilter5$$
$$+ 0.15625 * ProbaFilter2^3 * ProbaFilter4^3$$
$$+ 125.0 * ProbaFilter2^3 * ProbaFilter4^2 * ProbaFilter5^2$$
$$+ 0.9375 * ProbaFilter2^3 * ProbaFilter4^2 * ProbaFilter5$$
$$+ 66.40625 * ProbaFilter2^3 * ProbaFilter4 * ProbaFilter5^3$$
$$+ 1.40625 * ProbaFilter2^3 * ProbaFilter4 * ProbaFilter5^2$$
$$+ 16.40625 * ProbaFilter2^3 * ProbaFilter5^4$$
$$+ 1.09375 * ProbaFilter2^3 * ProbaFilter5^3$$
$$+ 18.75 * ProbaFilter2^2 * ProbaFilter3^4 * ProbaFilter4$$
$$+ 11.71875 * ProbaFilter2^2 * ProbaFilter3^4 * ProbaFilter5$$
$$+ 53.90625 * ProbaFilter2^2 * ProbaFilter3^3 * ProbaFilter4^2$$
$$+ 116.40625 * ProbaFilter2^2 * ProbaFilter3^3 * ProbaFilter4 * ProbaFilter5$$
$$+ 57.8125 * ProbaFilter2^2 * ProbaFilter3^3 * ProbaFilter5^2$$

$$
\begin{aligned}
&+ 0.3125 * ProbaFilter2^2 * ProbaFilter3^3 * ProbaFilter5 \\
&+ 81.25 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter4^3 \\
&+ 275.78125 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 278.90625 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 2.03125 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5 \\
&+ 70.3125 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter5^3 \\
&+ 2.65625 * ProbaFilter2^2 * ProbaFilter3^2 * ProbaFilter5^2 \\
&+ 64.84375 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4^4 \\
&+ 260.15625 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 321.875 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 3.90625 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 225.0 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^3 \\
&+ 7.96875 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 53.125 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter5^4 \\
&+ 2.8125 * ProbaFilter2^2 * ProbaFilter3 * ProbaFilter5^3 \\
&+ 13.28125 * ProbaFilter2^2 * ProbaFilter4^5 \\
&+ 68.75 * ProbaFilter2^2 * ProbaFilter4^4 * ProbaFilter5 \\
&+ 138.28125 * ProbaFilter2^2 * ProbaFilter4^3 * ProbaFilter5^2 \\
&+ 2.34375 * ProbaFilter2^2 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 128.90625 * ProbaFilter2^2 * ProbaFilter4^2 * ProbaFilter5^3 \\
&+ 4.6875 * ProbaFilter2^2 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 68.75 * ProbaFilter2^2 * ProbaFilter4 * ProbaFilter5^4 \\
&+ 5.15625 * ProbaFilter2^2 * ProbaFilter4 * ProbaFilter5^3 \\
&+ 13.28125 * ProbaFilter2^2 * ProbaFilter5^5 \\
&+ 1.71875 * ProbaFilter2^2 * ProbaFilter5^4 \\
&+ 5.46875 * ProbaFilter2 * ProbaFilter3^5 * ProbaFilter4 \\
&+ 4.6875 * ProbaFilter2 * ProbaFilter3^5 * ProbaFilter5 \\
&+ 32.8125 * ProbaFilter2 * ProbaFilter3^4 * ProbaFilter4^2 \\
&+ 67.1875 * ProbaFilter2 * ProbaFilter3^4 * ProbaFilter4 * ProbaFilter5 \\
&+ 29.6875 * ProbaFilter2 * ProbaFilter3^4 * ProbaFilter5^2 \\
&+ 0.3125 * ProbaFilter2 * ProbaFilter3^4 * ProbaFilter5 \\
&+ 55.46875 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter4^3 \\
&+ 173.4375 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 195.3125 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter4 * ProbaFilter5^2
\end{aligned}
$$

$$
\begin{aligned}
&+ 0.9375 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter4 * ProbaFilter5 \\
&+ 55.46875 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter5^3 \\
&+ 3.125 * ProbaFilter2 * ProbaFilter3^3 * ProbaFilter5^2 \\
&+ 60.15625 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4^4 \\
&+ 233.59375 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 0.3125 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4^3 \\
&+ 314.84375 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 3.125 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 205.46875 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5^3 \\
&+ 7.8125 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 50.78125 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter5^4 \\
&+ 4.6875 * ProbaFilter2 * ProbaFilter3^2 * ProbaFilter5^3 \\
&+ 18.75 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^5 \\
&+ 149.21875 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^4 * ProbaFilter5 \\
&+ 275.0 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^3 * ProbaFilter5^2 \\
&+ 3.90625 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 274.21875 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5^3 \\
&+ 9.84375 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 124.21875 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^4 \\
&+ 10.78125 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^3 \\
&+ 0.0625 * ProbaFilter2 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 17.96875 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5^5 \\
&+ 3.59375 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5^4 \\
&+ 0.0625 * ProbaFilter2 * ProbaFilter3 * ProbaFilter5^3 \\
&+ 5.46875 * ProbaFilter2 * ProbaFilter4^6 \\
&+ 39.84375 * ProbaFilter2 * ProbaFilter4^5 * ProbaFilter5 \\
&+ 76.5625 * ProbaFilter2 * ProbaFilter4^4 * ProbaFilter5^2 \\
&+ 1.40625 * ProbaFilter2 * ProbaFilter4^4 * ProbaFilter5 \\
&+ 87.5 * ProbaFilter2 * ProbaFilter4^3 * ProbaFilter5^3 \\
&+ 4.84375 * ProbaFilter2 * ProbaFilter4^3 * ProbaFilter5^2 \\
&+ 57.03125 * ProbaFilter2 * ProbaFilter4^2 * ProbaFilter5^4 \\
&+ 5.78125 * ProbaFilter2 * ProbaFilter4^2 * ProbaFilter5^3 \\
&+ 0.0625 * ProbaFilter2 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 25.78125 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5^5
\end{aligned}
$$

$$
\begin{aligned}
&+ 2.03125 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5^4 \\
&+ 0.15625 * ProbaFilter2 * ProbaFilter4 * ProbaFilter5^3 \\
&+ 3.90625 * ProbaFilter2 * ProbaFilter5^6 \\
&+ 1.25 * ProbaFilter2 * ProbaFilter5^5 \\
&+ 0.09375 * ProbaFilter2 * ProbaFilter5^4 \\
&+ 2.34375 * ProbaFilter3^6 * ProbaFilter4 \\
&+ 0.78125 * ProbaFilter3^6 * ProbaFilter5 \\
&+ 7.03125 * ProbaFilter3^5 * ProbaFilter4^2 \\
&+ 12.5 * ProbaFilter3^5 * ProbaFilter4 * ProbaFilter5 \\
&+ 3.125 * ProbaFilter3^5 * ProbaFilter5^2 \\
&+ 0.15625 * ProbaFilter3^5 * ProbaFilter5 \\
&+ 11.71875 * ProbaFilter3^4 * ProbaFilter4^3 \\
&+ 56.25 * ProbaFilter3^4 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 43.75 * ProbaFilter3^4 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 0.3125 * ProbaFilter3^4 * ProbaFilter4 * ProbaFilter5 \\
&+ 19.53125 * ProbaFilter3^4 * ProbaFilter5^3 \\
&+ 0.625 * ProbaFilter3^4 * ProbaFilter5^2 \\
&+ 21.09375 * ProbaFilter3^3 * ProbaFilter4^4 \\
&+ 72.65625 * ProbaFilter3^3 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 120.3125 * ProbaFilter3^3 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 1.25 * ProbaFilter3^3 * ProbaFilter4^2 * ProbaFilter5 \\
&+ 72.65625 * ProbaFilter3^3 * ProbaFilter4 * ProbaFilter5^3 \\
&+ 3.59375 * ProbaFilter3^3 * ProbaFilter4 * ProbaFilter5^2 \\
&+ 14.0625 * ProbaFilter3^3 * ProbaFilter5^4 \\
&+ 1.5625 * ProbaFilter3^3 * ProbaFilter5^3 \\
&+ 9.375 * ProbaFilter3^2 * ProbaFilter4^5 \\
&+ 55.46875 * ProbaFilter3^2 * ProbaFilter4^4 * ProbaFilter5 \\
&+ 0.15625 * ProbaFilter3^2 * ProbaFilter4^4 \\
&+ 132.8125 * ProbaFilter3^2 * ProbaFilter4^3 * ProbaFilter5^2 \\
&+ 2.1875 * ProbaFilter3^2 * ProbaFilter4^3 * ProbaFilter5 \\
&+ 128.125 * ProbaFilter3^2 * ProbaFilter4^2 * ProbaFilter5^3 \\
&+ 5.46875 * ProbaFilter3^2 * ProbaFilter4^2 * ProbaFilter5^2 \\
&+ 60.9375 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5^4 \\
&+ 4.0625 * ProbaFilter3^2 * ProbaFilter4 * ProbaFilter5^3
\end{aligned}
$$

$$+ 8.59375 * ProbaFilter3^2 * ProbaFilter5^5$$
$$+ 1.40625 * ProbaFilter3^2 * ProbaFilter5^4$$
$$+ 0.03125 * ProbaFilter3^2 * ProbaFilter5^3$$
$$+ 9.375 * ProbaFilter3 * ProbaFilter4^6$$
$$+ 36.71875 * ProbaFilter3 * ProbaFilter4^5 * ProbaFilter5$$
$$+ 0.15625 * ProbaFilter3 * ProbaFilter4^5$$
$$+ 72.65625 * ProbaFilter3 * ProbaFilter4^4 * ProbaFilter5^2$$
$$+ 1.25 * ProbaFilter3 * ProbaFilter4^4 * ProbaFilter5$$
$$+ 99.21875 * ProbaFilter3 * ProbaFilter4^3 * ProbaFilter5^3$$
$$+ 4.84375 * ProbaFilter3 * ProbaFilter4^3 * ProbaFilter5^2$$
$$+ 60.9375 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5^4$$
$$+ 7.03125 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5^3$$
$$+ 0.03125 * ProbaFilter3 * ProbaFilter4^2 * ProbaFilter5^2$$
$$+ 29.6875 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^5$$
$$+ 2.96875 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^4$$
$$+ 0.15625 * ProbaFilter3 * ProbaFilter4 * ProbaFilter5^3$$
$$+ 4.6875 * ProbaFilter3 * ProbaFilter5^6$$
$$+ 0.625 * ProbaFilter3 * ProbaFilter5^5$$
$$+ 0.09375 * ProbaFilter3 * ProbaFilter5^4$$
$$+ 0.78125 * ProbaFilter4^7$$
$$+ 3.125 * ProbaFilter4^6 * ProbaFilter5$$
$$+ 20.3125 * ProbaFilter4^5 * ProbaFilter5^2$$
$$+ 0.3125 * ProbaFilter4^5 * ProbaFilter5$$
$$+ 26.5625 * ProbaFilter4^4 * ProbaFilter5^3$$
$$+ 1.875 * ProbaFilter4^4 * ProbaFilter5^2$$
$$+ 26.5625 * ProbaFilter4^3 * ProbaFilter5^4$$
$$+ 2.96875 * ProbaFilter4^3 * ProbaFilter5^3$$
$$+ 0.09375 * ProbaFilter4^3 * ProbaFilter5^2$$
$$+ 10.9375 * ProbaFilter4^2 * ProbaFilter5^5$$
$$+ 1.875 * ProbaFilter4^2 * ProbaFilter5^4$$
$$+ 0.3125 * ProbaFilter4^2 * ProbaFilter5^3$$
$$+ 1.5625 * ProbaFilter4 * ProbaFilter5^6$$
$$+ 1.09375 * ProbaFilter4 * ProbaFilter5^5$$
$$+ 0.25 * ProbaFilter4 * ProbaFilter5^4$$

$+ 1.5625 * ProbaFilter5^{7}$

$+ 0.3125 * ProbaFilter5^{6}$

$+ 0.03125 * ProbaFilter5^{5}$

Titre : Modélisation formelle de systèmes de drones civils à l'aide de méthodes probabilistes paramétrées

Mot clés : Drone, Modèle formel, Chaîne de Markov, *Model checking statistique paramétrique*

Résumé : Les drones sont maintenant très répandus dans la société et sont souvent utilisés dans des situations dangereuses pour le public environnant. Il est alors nécessaire d'étudier leur fiabilité, en particulier dans le contexte de vols au-dessus d'un public. Dans cette thèse, nous étudions la modélisation et l'analyse de drones dans le contexte de leur plan de vol. Pour cela, nous construisons plusieurs modèles probabilistes du drone et les utilisons ainsi que le plan de vol pour modéliser la trajectoire du drone. Le modèle le plus détaillé obtenu prend en compte des paramètres comme la précision de l'estimation de la position par les différents capteurs, ainsi que la force et la direction du vent. Le modèle est analysé afin de mesurer la probabilité que le drone entre dans une zone dangereuse. Du fait de la nature et de la complexité des modèles successifs obtenus, leur vérification avec les outils classiques, tels que PRISM ou PARAM, est impossible. Nous utilisons donc une nouvelle méthode d'approximation, appelée *Model Checking Statistique Paramétrique*. Cette méthode a été implémentée dans un prototype, que nous avons mis à l'épreuve sur ce cas d'étude complexe. Nous avons pour finir utilisé les résultats fournis par ce prototype afin de proposer des pistes permettant d'améliorer la sécurité du public dans le contexte considéré.

Title: Parametric Statistical model checking of UAV flight plan

Keywords: UAV, Formal Model, Markov Chain, Parametric statistical model checking

Abstract: Unmanned Aerial Vehicles (UAV) are now widespread in our society and are often used in a context where they can put people at risk. Studying their reliability, in particular in the context of flight above a crowd, thus becomes a necessity. In this thesis, we study the modeling and analysis of UAV in the context of their flight plan. To this purpose, we build several parametrics probabilistic models of the UAV and use them, as well as a given flight plan, in order to model its trajectory. Our most advanced model takes into account the precision of position estimation by embedded sensors, as well as wind force and direction. The model is analyzed in order to measure the probability that the drone enters an unsafe zone. Because of the nature and complexity of the obtained models, their exact verification using classical tools such as PRISM or PARAM is impossible. We therefore develop a new approximation method, called Parametric Statistical Model Checking. This method has been implemented in a prototype tool, which we tested on this complex case study. Finally, we use the result to propose some ways to improve the safety of the public in our context.