

Planification de trajectoire et contrôle pour la course autonome par optimisation et apprentissage par renforcement

par

Victor DANG

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE DES TECHNOLOGIES DE L'INFORMATION
M. Sc. A.

MONTREAL, LE 8 OCTOBRE 2026

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC



Victor Dang, 2026



Cette licence Creative Commons signifie qu'il est permis de diffuser, d'imprimer ou de sauvegarder sur un autre support une partie ou la totalité de cette oeuvre à condition de mentionner l'auteur, que ces utilisations soient faites à des fins non commerciales et que le contenu de l'oeuvre n'ait pas été modifié.

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Christian BELLEAU, directeur de mémoire
Département de génie mécanique à l'École de technologie supérieure

M. Jean-Phillipe ROBERGE, codirecteur
Département de génie des systèmes à l'École de technologie supérieure

M. Simon JONCAS, président du jury
Département de génie des systèmes à l'École de technologie supérieure

M. Georges GHAZI, membre du jury
Département de génie des systèmes à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 23 SEPTEMBRE 2026

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je souhaite tout d'abord remercier mes directeurs de recherche Christian Belleau et Jean-Philippe Roberge. Votre encadrement et votre soutien m'ont été précieux. J'ai été octroyé une grande liberté pour me permettre d'explorer différentes directions et de satisfaire ma curiosité de maintes différentes façons.

À mes amis dispersés à travers le monde : merci de faire partie de ma vie.

Un grand merci à Maman, Papa, Théophile, Frédéric et François. Votre soutien a été essentiel pour me permettre d'aller au bout de ce parcours et de voir le fruit de mon travail arriver à maturité.

Planification de trajectoire et contrôle pour la course autonome par optimisation et apprentissage par renforcement

Victor DANG

RÉSUMÉ

La course autonome constitue un point de référence exigeant pour les algorithmes de perception, de planification et de contrôle : un véhicule de course doit fonctionner à la limite de son enveloppe de performance physique, où la dynamique non linéaire du pneu impose des exigences extrêmes à chacun des sous-systèmes. Pour résoudre ce problème, il faut construire une carte de l'environnement, estimer l'état du véhicule, générer une trajectoire optimale et la suivre sous des contraintes dynamiques strictes, mais l'intégration systématique de ces sous-systèmes sur une plateforme unifiée reste un défi ouvert.

Ce mémoire se concentre sur la planification et le suivi de trajectoire pour la plateforme de course autonome F1TENTH. Les expériences sont menées entièrement en simulation via F1TENTH Gym.

Il est présenté un planificateur de trajectoire hors ligne qui fournit à la fois une ligne de course avec une courbure minimale et un profil de vitesse. La trajectoire géométrique est calculée en résolvant un problème d'optimisation sous la forme d'un programme quadratique. Le profil de vitesse est obtenu avec un schéma d'intégration avant-arrière qui respecte les contraintes du cercle de friction.

Deux types de contrôleurs sont explorés. Un premier basé sur un modèle est implémenté par commande prédictive (MPC) sous la forme d'un programme non linéaire contraint résolu sur un horizon glissant fini. Le second est basé sur les données. L'algorithme *Soft Actor-Critic* (SAC) est utilisé pour entraîner des agents par apprentissage par renforcement en tant que politiques de contrôle. Ces politiques sont entraînées sans dépendre d'une trajectoire de référence.

Mots-clés: course autonome, F1TENTH, optimisation sous contraintes, programmation quadratique, programmation non linéaire, processus de décision markovien partiellement observable, apprentissage par renforcement

Trajectory planning and control for autonomous racing using optimization and reinforcement learning

Victor DANG

ABSTRACT

Autonomous racing provides a demanding benchmark for perception, planning and control algorithms : a race car must operate at the edge of its physical performance envelope, where nonlinear tire dynamics impose extreme requirements on each sub-system. Solving this problem requires constructing a map of the environment, estimating the vehicle state, generating an optimal trajectory, and tracking it under strict dynamic constraints, but the systematic integration of these sub-systems on a unified platform remains an open challenge.

This thesis focuses on trajectory planning and tracking for the F1TENTH autonomous racing platform. Experimentation is done entirely in simulation through F1TENTH Gym.

An offline trajectory planner is introduced which provides both a racing line with a minimum curvature and a velocity profile. The geometric trajectory is computed through solving an optimization problem under the form of a quadratic program. The velocity profile is obtained with a forward-backward integration scheme that respects the friction circle constraints.

Two types of controllers are explored. The first one is a model-based implemented as a Model Predictive Controller (MPC) where a constrained non-linear program is solved over a receding finite horizon. The second one is a data-driven. The Soft Actor-Critic (SAC) algorithm is used to train reinforcement learning agents as control policies. These policies are trained without depending on a reference trajectory.

Keywords: autonomous racing, F1TENTH, constrained optimization, quadratic programming, nonlinear programming, partially observable Markov decision process, reinforcement learning

TABLE DES MATIÈRES

	Page
INTRODUCTION	1
CHAPITRE 1 CONTEXTE THÉORIQUE	5
1.1 Dynamique du véhicule	5
1.1.1 Modèle de bicyclette cinématique	5
1.1.2 Modèle de bicyclette dynamique	6
1.1.3 Modèles de pneus	7
1.1.3.1 Formule magique de Pacejka	8
1.1.3.2 Modèle de pneus linéaires	9
1.1.4 Cercle de friction	9
1.2 Description et localisation sur un circuit	10
1.2.1 Représentation d'un circuit et d'une trajectoire de course	10
1.2.2 Localisation sur un circuit	11
CHAPITRE 2 INFRASTRUCTURE LOGICIELLE	13
2.1 Ressources informatiques	13
2.2 Bibliothèques Python utilisées	13
2.3 F1TENTH Gym	13
2.3.1 Gestion des dépendances	14
2.3.2 Modèles de véhicule	14
2.3.3 Circuits	15
2.3.4 Espace d'observation	15
2.3.5 Espace d'action	16
2.3.6 Exécution d'un épisode	17
2.3.6.1 Initialisation d'épisode	17
2.3.6.2 Exécution d'un pas de temps	18
2.3.6.3 Fin d'épisode	18
2.4 Suivi des expériences	19
CHAPITRE 3 PLANIFICATION DE TRAJECTOIRE	21
3.1 Travaux connexes	21
3.1.1 Optimisation géométrique de la ligne de course	21
3.1.2 Approches découplées trajectoire-vitesse	22
3.1.3 Formulations couplées avec minimisation du temps	22
3.1.4 Approches alternatives	23
3.1.5 Contexte en F1TENTH	23
3.2 Fondements théoriques	23
3.2.1 D'une représentation discrète à une trajectoire continue	24
3.2.2 Objectif d'optimisation à courbure minimale	26
3.2.3 Formulation du programme quadratique	27

3.2.3.1	Système des coefficients de spline M	27
3.2.3.2	Second membre et dépendance en α	31
3.2.3.3	Matrices d'extraction et dérivées affines	33
3.2.3.4	Forme quadratique de la courbure	34
3.2.3.5	Formulation du programme quadratique sans contrainte	35
3.2.3.6	Contraintes au programme quadratique	36
3.2.4	Affinement itératif	38
3.2.5	Génération du profil de vitesse	38
3.2.5.1	Vitesse maximale selon la courbure	38
3.2.5.2	Passe avant	39
3.2.5.3	Passe en arrière	39
3.2.5.4	Profil de vitesse final	39
3.3	Expérimentation	40
3.3.1	Résultats	40
3.3.1.1	Trajectoires générées	41
3.3.1.2	Profils de vitesse	43
3.4	Conclusions	45
CHAPITRE 4 CONTRÔLE PAR COMMANDE PRÉDICTIVE		47
4.1	Travaux connexes	47
4.1.1	MPC pour la course autonome	48
4.1.2	Apprentissage en ligne	49
4.1.3	Commande prédictive par échantillonnage	50
4.2	Fondements théoriques	51
4.2.1	Formulation générale du MPC	51
4.2.2	NMPC pour la course autonome	51
4.2.3	Modèle de prédiction	52
4.2.4	Contraintes du problème d'optimisation	53
4.2.4.1	Contraintes liées au circuit	53
4.2.4.2	Contraintes sur les commandes	54
4.2.4.3	Contraintes sur les variations des commandes	54
4.3	Mise en œuvre	54
4.3.1	Discrétisation et horizon de prédiction	54
4.3.2	Réglage des paramètres	55
4.3.2.1	Poids de la fonction de coût	55
4.3.2.2	Marges et bornes	55
4.3.2.3	Initialisation à chaud	56
4.3.3	Implémentation logicielle	56
4.4	Résultats	57
4.4.1	Suivi de trajectoire	58
4.4.2	Suivi du profil de vitesse	59
4.4.3	Fluidité des actions	61
4.5	Conclusions	62

CHAPITRE 5	CONTRÔLE PAR APPRENTISSAGE PROFOND PAR RENFORCEMENT	65
5.1	Travaux connexes	65
5.1.1	Premières études en simulation	65
5.1.2	Performance en simulation face à des pilotes humains	66
5.1.3	Plateformes miniatures et transfert simulation-réalité	66
5.1.4	Apprentissage hors ligne et généralisation	67
5.2	Fondements théoriques	67
5.2.1	Réseaux de neurones	67
5.2.2	Entraînement, rétropropagation et optimisation	71
5.2.3	Processus de décision de Markov	72
5.2.4	Méthodes de gradient de politique	73
5.2.5	Soft Actor-Critic	73
5.3	Mise en œuvre	76
5.3.1	Architecture des réseaux de neurones	76
5.3.2	Espace d'état et espace d'action	76
5.3.3	Conception de la récompense	77
5.3.4	Structure des épisodes	78
5.3.5	Hyperparamètres	78
5.4	Résultats	79
5.4.1	Courbes d'apprentissage	79
5.4.2	Efficacité d'échantillonnage	80
5.4.3	Analyse de trajectoire	81
5.4.4	Profil de vitesse	82
5.4.5	Fluidité des actions	83
5.4.6	Comparaison avec le MPC	85
5.5	Conclusions	90
	CONCLUSION ET RECOMMANDATIONS	93
6.1	Synthèse	93
6.2	Limitations et hypothèses	94
6.2.1	Suppositions simplificatrices	94
6.2.2	Limites méthodologiques	94
6.3	Travaux futurs	95
6.3.1	Déploiement futur	95
6.3.2	Course multi-agents	95
6.3.3	Rapprochement entre optimisation et apprentissage	95
ANNEXE I	DÉMONSTRATIONS MATHÉMATIQUES	97
	BIBLIOGRAPHIE	115

LISTE DES TABLEAUX

	Page
Tableau 2.1	Librairies Python utilisées 13
Tableau 3.1	Statistiques sur les trajectoires générées sur le circuit de Montréal 44
Tableau 3.2	Statistiques sur le profil de vitesse généré sur la trajectoire à courbure minimale sur le circuit de Montréal 44
Tableau 4.1	Poids de la fonction de coût 55
Tableau 5.1	Hyperparamètres d'entraînement 78
Tableau 5.2	Comparaison des performances du MPC et de l'agent RL 90

LISTE DES FIGURES

	Page
Figure 0.1	Plateforme type F1TENTH Tiré de https://roboracer.ai/build 2
Figure 1.1	Modèle de bicyclette cinématique Adaptée de Kong, Pfeiffer, Schildbach & Borrelli (2015) 6
Figure 1.2	Modèle de bicyclette dynamique Adaptée de Kloeser <i>et al.</i> (2020) 7
Figure 1.3	Modélisation de la force latérale par la Formule magique de Pacejka Tirée de Bakker, Pacejka & Lidner (1989) 9
Figure 1.4	Représentation d'un circuit 10
Figure 1.5	Représentation d'une trajectoire de course 11
Figure 1.6	Modèle de bicyclette dans un repère de Serret-Frenet Tirée de Raji <i>et al.</i> (2022) 12
Figure 2.1	Circuit Gilles Villeneuve à Montréal réadapté pour F1TENTH Gym 15
Figure 2.2	<i>Look ahead</i> observable 16
Figure 2.3	Expériences journalisées avec MLflow 19
Figure 2.4	Métriques d'une expérience suivie avec MLflow 20
Figure 3.1	Interpolation ouverte avec trois splines de degrés un à trois entre cinq points 24
Figure 3.2	Interpolation fermée par splines cubiques entre 5 points 25
Figure 3.3	Fonctions $x_i(t)$ et $y_i(t)$ des cinq splines cubiques de la Figure 3.2 25
Figure 3.4	Fonctions $x(t)$, $x'(t)$ et $x''(t)$ 28
Figure 3.5	Trajectoire avec une itération sur le circuit de Montréal 41
Figure 3.6	Trajectoire avec quatre itérations sur le circuit de Montréal 42
Figure 3.7	Superposition des trajectoires avec une itération et quatre itérations sur le premier virage du circuit de Montréal 42
Figure 3.8	Superposition des trajectoires avec une itération et quatre itérations sur le treizième et quatorzième virages du circuit de Montréal 43

Figure 3.9	Profils de courbure des trajectoires calculées sur le circuit de Montréal ..	43
Figure 3.10	Profil de vitesse généré projeté sur la trajectoire IQP du circuit de Montréal	44
Figure 3.11	Profils de vitesse générés sur la trajectoire à courbure minimale et la ligne de référence	45
Figure 4.1	Trajectoire de référence et trajectoire suivie par le MPC sur le circuit de Montréal	58
Figure 4.2	Erreur latérale par rapport à la trajectoire de référence du MPC	59
Figure 4.3	Déviations latérales par rapport à la ligne centrale de la trajectoire de référence et de la trajectoire suivie par le MPC	60
Figure 4.4	Vitesse suivie par le MPC et profil de vitesse de référence	60
Figure 4.5	Commandes de direction et d'accélération du MPC	61
Figure 5.1	Fonction sigmoïde et sa dérivée	68
Figure 5.2	Fonction tangente hyperbolique et sa dérivée	69
Figure 5.3	Fonction unité linéaire rectifiée et sa dérivée	69
Figure 5.4	Vue d'ensemble de l'algorithme <i>Soft Actor-Critic</i>	75
Figure 5.5	Courbes d'apprentissage sur le circuit de Montréal	79
Figure 5.6	Récompense et progression d'évaluation normalisées par leur maximum, en fonction du pas de temps d'entraînement	81
Figure 5.7	Trajectoire fermée de l'agent SAC sur le circuit de Montréal	82
Figure 5.8	Déviations latérales par rapport à la ligne centrale de la trajectoire de référence et de la trajectoire suivie par l'agent RL	82
Figure 5.9	Superposition de la trajectoire IQP et de la trajectoire suivie par l'agent RL sur le sixième et septième virages du circuit de Montréal	83
Figure 5.10	Profil de vitesse de l'agent SAC	84
Figure 5.11	Angle de glissement de l'agent SAC	84
Figure 5.12	Commandes de l'agent SAC	85

Figure 5.13	Trajectoires de référence, du MPC et de l'agent RL sur le circuit de Montréal	86
Figure 5.14	Déviations latérales du MPC et de l'agent RL	86
Figure 5.15	Superposition des trajectoires de référence, du MPC et de l'agent RL sur le sixième et septième virages du circuit de Montréal	87
Figure 5.16	Profils de vitesse de référence, du MPC et de l'agent RL. La courbure absolue $ \kappa $ localise les virages du circuit	88
Figure 5.17	Angle de glissement du MPC et de l'agent RL	89
Figure 5.18	Commandes de direction et d'accélération du MPC et de l'agent RL	89

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACADO	Automatic Control And Dynamical Optimization
CasADi	Computer Algebra System with Automatic Differentiation
DARPA	Defense Advanced Research Projects Agency
DL	Deep Learning
DRL	Deep Reinforcement Learning
IPOPT	Interior Point OPTimizer
LMPC	Learning-based Model Predictive Control
LPV-MPC	Linear Parameter Varying Model Predictive Control
MDP	Markov Decision Process
MPC	Model Predictive Control
MPCC	Model Predictive Control Contouring
MPPI	Model Predictive Path Integral
NMPC	Nonlinear Model Predictive Control
OSQP	Operator Splitting Quadratic Program
PANOC	Proximal Average Newton-type method for Optimal Control
pip	Pip Installs Packages
QP	Quadratic Program
TORCS	The Open Racing Car Simulator
IQP	Iterative Quadratic Program
RL	Reinforcement Learning
SAC	Soft Actor-Critic

INTRODUCTION

Le sport automobile a toujours été l'un des accélérateurs les plus efficaces pour la technologie automobile. Des innovations telles que les freins à disque, la turbocompression, les motorisations hybrides et les conceptions aérodynamiques avancées ont d'abord été développées et validées sur la piste avant d'être transférées sur des véhicules de production. Au cours des dernières années, ce modèle s'est étendu à la conduite autonome : des compétitions telles que le DARPA Grand Challenge, l'Indy Autonomous Challenge (Wischnewski *et al.*, 2022), et la série Roborace ont positionné la course autonome comme une frontière à repousser pour faire progresser l'état de l'art dans la robotique et le transport intelligent (Betz *et al.*, 2022).

L'attrait de la course en tant que plateforme de recherche réside dans les conditions opérationnelles extrêmes qu'elle impose. Un véhicule de course fonctionne à la limite de son enveloppe de performance, où la dynamique non linéaire des pneus et le couplage entre les forces longitudinales et latérales deviennent des préoccupations de premier ordre (Guiggiani, 2023). Ces conditions exposent des modes de défaillance qui restent cachés à des vitesses modérées et exigent des algorithmes capables de prendre des décisions en temps réel avec des marges d'erreur minimales.

La course autonome présente des défis algorithmiques distincts par rapport à la conduite autonome urbaine. Alors que la conduite urbaine met l'accent sur la sécurité, le respect des règles et l'interaction dans un trafic avec des participants hétérogènes, la course optimise un seul objectif : minimiser le temps au tour sous de strictes contraintes physiques. Cette simplification de l'objectif, combinée à l'importance des effets de la dynamique du véhicule, crée un environnement où les algorithmes de perception, de planification et de contrôle sont testés à leurs limites fondamentales.

Pour résoudre le problème de course autonome, il faut traiter un ensemble de sous-problèmes interdépendants : le véhicule doit d'abord construire une carte de son environnement et s'y localiser, puis planifier une trajectoire qui minimise le temps au tour, et enfin suivre cette trajectoire

précisément dans des conditions dynamiques exigeantes. Ces interactions interdépendantes impliquent que la performance du système global est limitée par son maillon faible.

La plateforme RoboRacer, anciennement connue sous le nom de F1TENTH (O’Kelly, Zheng, Karthik & Mangharam, 2020), est apparue comme une plateforme de recherche largement adoptée pour la course autonome. Par souci de simplicité, tout le long de ce mémoire on désignera la plateforme sous le nom F1TENTH. À l’échelle réduite de 1/10, les véhicules F1TENTH reproduisent la dynamique essentielle que rencontre une voiture de course à une fraction du coût et du risque. La communauté F1TENTH couvre plus de 70 universités dans le monde, organise des concours annuels et maintient une forte culture de reproductibilité et de développement open-source (Evans *et al.*, 2024).



Figure 0.1 Plateforme type F1TENTH
Tiré de <https://roboracer.ai/build>

Tous les travaux expérimentaux présentés dans ce mémoire sont menés en simulation, sans accès au matériel physique. Le simulateur utilisé est F1TENTH Gym (Babu & Behl, 2020). Ce dernier fournit un simulateur dans un environnement de type Gymnasium (Towers *et al.*, 2023). Cette

interface standardisée permet notamment d’entraîner par apprentissage par renforcement des politiques de contrôle.

Le chapitre 1 présente le contexte théorique permettant une compréhension des chapitres suivants. Puis, dans le chapitre 2, l’infrastructure logicielle mise en place est décrite. En particulier, l’environnement de simulation F1TENTH Gym est présenté ainsi que les modifications qui ont été apportées. Ensuite, un planificateur de trajectoire est détaillé dans le chapitre 3. Ce planificateur fournit une trajectoire géométrique avec une courbure minimale et un profil de vitesse s’appuyant sur les contraintes du cercle de friction. Il s’ensuit le chapitre 4 qui couvre un contrôleur par commande prédictive aussi appelé *Model Predictive Control* (MPC). Enfin, le chapitre 5 décrit comment des agents entraînés par apprentissage par renforcement, aussi appelés agents RL (*Reinforcement Learning*), sont utilisés pour contrôler le véhicule sans s’appuyer sur des trajectoires pré-calculées.

CHAPITRE 1

CONTEXTE THÉORIQUE

Ce chapitre établit les fondements théoriques nécessaires pour la compréhension des chapitres suivants.

1.1 Dynamique du véhicule

La course automobile nécessite d'opérer aux limites physiques des performances du véhicule, où de petites différences dans la précision des modèles dynamiques se traduisent directement en temps au tour et en stabilité. Pour comprendre comment un véhicule réagit aux commandes de direction et aux entrées au niveau de la pédale d'accélérateur, il faut des modèles qui capturent la physique pertinente à chaque régime de vitesse. Cette section présente la hiérarchie des modèles de véhicules utilisés dans la recherche en course autonome, des approximations cinématiques simples valables à basse vitesse aux formulations dynamiques complètes avec des modèles de pneus non linéaires requis près de la limite d'adhérence.

1.1.1 Modèle de bicyclette cinématique

Le modèle de bicyclette cinématique réduit les quatre roues à deux roues virtuelles le long de l'axe longitudinal du véhicule et néglige le glissement des pneus (Rajamani, 2012). Le vecteur d'état est $\mathbf{x} = [x, y, \psi, v]^\top$, représentant la position cartésienne, l'angle de cap et la vitesse longitudinale. La dynamique est :

$$\dot{x} = v \cos(\psi + \beta) \quad (1.1)$$

$$\dot{y} = v \sin(\psi + \beta) \quad (1.2)$$

$$\dot{\psi} = \frac{v}{l_r} \sin(\beta) \quad (1.3)$$

$$\dot{v} = a \quad (1.4)$$

où $\beta = \arctan\left(\frac{l_r}{l_f+l_r} \tan(\delta)\right)$ est l'angle de glisse latéral au centre de gravité, δ est l'angle de direction avant, l_f et l_r sont les distances entre le centre de gravité et les essieux avant et arrière, et a est l'accélération longitudinale. La figure 1.1 présente le modèle de bicyclette cinématique.

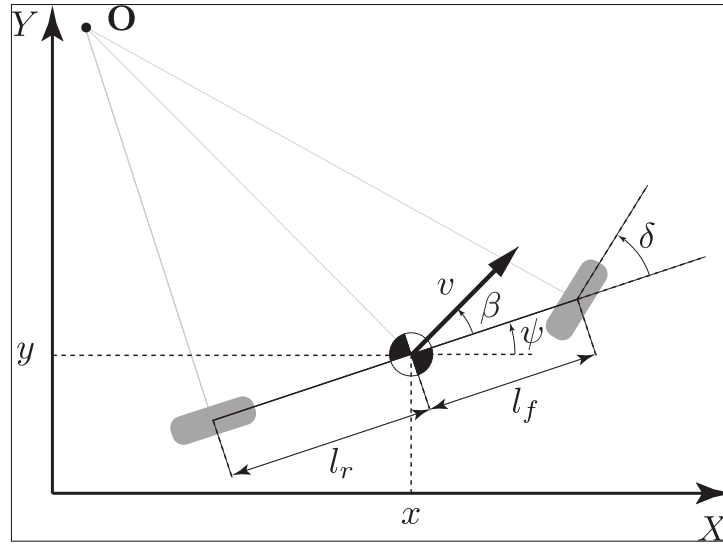


Figure 1.1 Modèle de bicyclette cinématique
Adaptée de Kong *et al.* (2015)

Le modèle cinématique est valable à basse vitesse où les angles de glissement des pneus sont négligeables. Il fournit une approximation utile dans les virages à basse vitesse et forme la base de nombreux algorithmes de planification, mais il devient inexact lorsque les forces latérales approchent les limites d'adhérence des pneus (Guiggiani, 2023).

1.1.2 Modèle de bicyclette dynamique

À des vitesses plus élevées, les forces latérales du pneu introduisent un glissement que le modèle cinématique ne peut capturer. Le modèle de bicyclette dynamique étend la représentation de la bicyclette en intégrant des équations d'équilibre des forces et des modèles de pneus explicites (Rajamani, 2012).

Le vecteur d'état devient $\mathbf{x} = [x, y, \psi, v_x, v_y, \dot{\psi}]^T$, où v_x et v_y sont les vitesses longitudinales et latérales dans le référentiel lié au véhicule et $\dot{\psi}$ est le taux de lacet. Les équations de mouvement

1.1.3.1 Formule magique de Pacejka

Bakker *et al.* (1989) ont introduit une formule capable de caractériser l'état stationnaire des forces s'appliquant à un pneu dans toute la gamme des conditions de glissement. Ayant été obtenue empiriquement, elle est connue sous le nom de «Formule magique de Pacejka» et a été ultérieurement améliorée par Pacejka & Bakker (1992). Dans le cas d'un glissement latéral pur, la force latérale est fonction de l'angle de glisse α :

$$F_y = D \sin(C \arctan(B\alpha - E(B\alpha - \arctan(B\alpha)))) \quad (1.8)$$

où les coefficients sont :

- B : facteur de rigidité
- C : facteur de forme
- D : amplitude maximale
- E : facteur de courbure proche du pic

Les coefficients B , C , et E sont sans dimension tandis que D a la dimension d'une force (MLT^{-2}).

L'angle de glisse α est l'angle entre la direction dans laquelle la roue pointe et la direction dans laquelle elle se déplace.

La figure 1.3 peut être décrite comme une fonction passant par l'origine, atteignant un maximum puis tendant vers une asymptote horizontale. Elle capture trois régimes :

- le régime linéaire pour des angles de glissement petits,
- le régime de force maximale pour l'angle de glissement optimal,
- le régime de force réduite pour des angles de glissement grands où le pneu passe au dérapage.

Pour un angle de glisse nul, le produit BCD équivaut à la rigidité de dérive C_α et à la pente initiale dans la Figure 1.3 qui correspond au régime linéaire pour des angles de glisses petits.

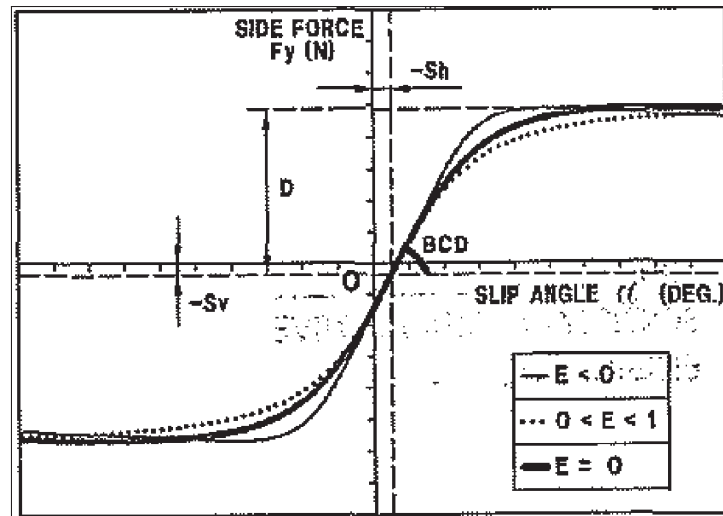


Figure 1.3 Modélisation de la force latérale par la Formule magique de Pacejka
Tirée de Bakker *et al.* (1989)

1.1.3.2 Modèle de pneus linéaires

Pour de petits angles de glisse α , on approxime la force latérale du pneu par la relation linéaire :

$$F_y = -C_\alpha \alpha \quad (1.9)$$

Cette approximation surestime les forces près de la limite d'adhérence, où le pneu sature.

1.1.4 Cercle de friction

La capacité de force latérale et longitudinale combinée appliquée à un pneu peut être représentée par un cercle de friction :

$$\sqrt{F_x^2 + F_y^2} \leq \mu F_z \quad (1.10)$$

où μ est le coefficient de frottement et F_z est la force normale appliquée au pneu.

Cette contrainte implique que l'utilisation d'une force longitudinale (freinage ou accélération) réduit la force latérale disponible (la prise de virage), et vice versa. Le contexte de course

automobile nécessite de prendre des trajectoires où le véhicule se retrouve près de la limite de frottement (Milliken & Milliken, 1995; Guiggiani, 2023) afin d'exploiter l'entièreté de son enveloppe de performance.

1.2 Description et localisation sur un circuit

1.2.1 Représentation d'un circuit et d'une trajectoire de course

Un circuit est représenté en premier lieu par une ligne centrale de référence. La ligne centrale est discrétisée en N points $\{(x_i, y_i)\}_{i=1}^N = \{\vec{p}_i\}_{i=1}^N$. Les limites latérales du circuit sont obtenues en annotant pour chaque point $\vec{p}_i$ la distance de part et d'autre de la ligne centrale w_i^{gauche} et w_i^{droite} .

On obtient les coordonnées des limites droite et gauche par les vecteurs normaux :

$\forall i \in \{1, \dots, N\}$, $\vec{n}_i$, le vecteur unitaire normal à la ligne centrale au voisinage du point $\vec{p}_i$

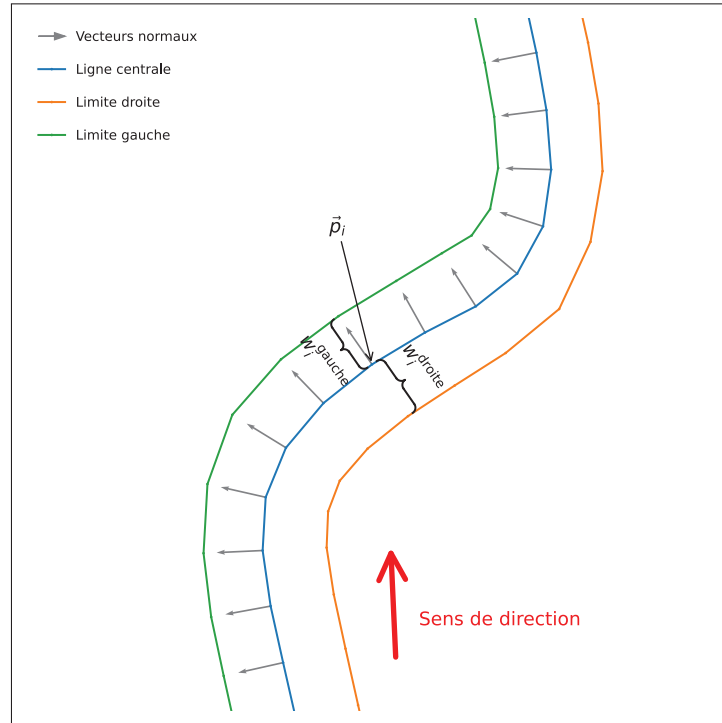


Figure 1.4 Représentation d'un circuit

Considérons une trajectoire de course quelconque décrite par un ensemble de points $\{\vec{r}_i\}_{i=1}^N$. Celle-ci est paramétrée comme un ensemble de déviations latérales $\{\alpha_i\}_{i=1}^N$ par rapport à la ligne centrale avec les contraintes limites suivantes :

$$-w_i^{\text{gauche}} \leq \alpha_i \leq w_i^{\text{droite}}, \quad i = 1, \dots, N \quad (1.11)$$

Ainsi, on obtient les égalités suivantes :

$$\forall i \in \{1, \dots, N\}, \vec{r}_i = \vec{p}_i + \alpha_i \vec{n}_i \quad (1.12)$$

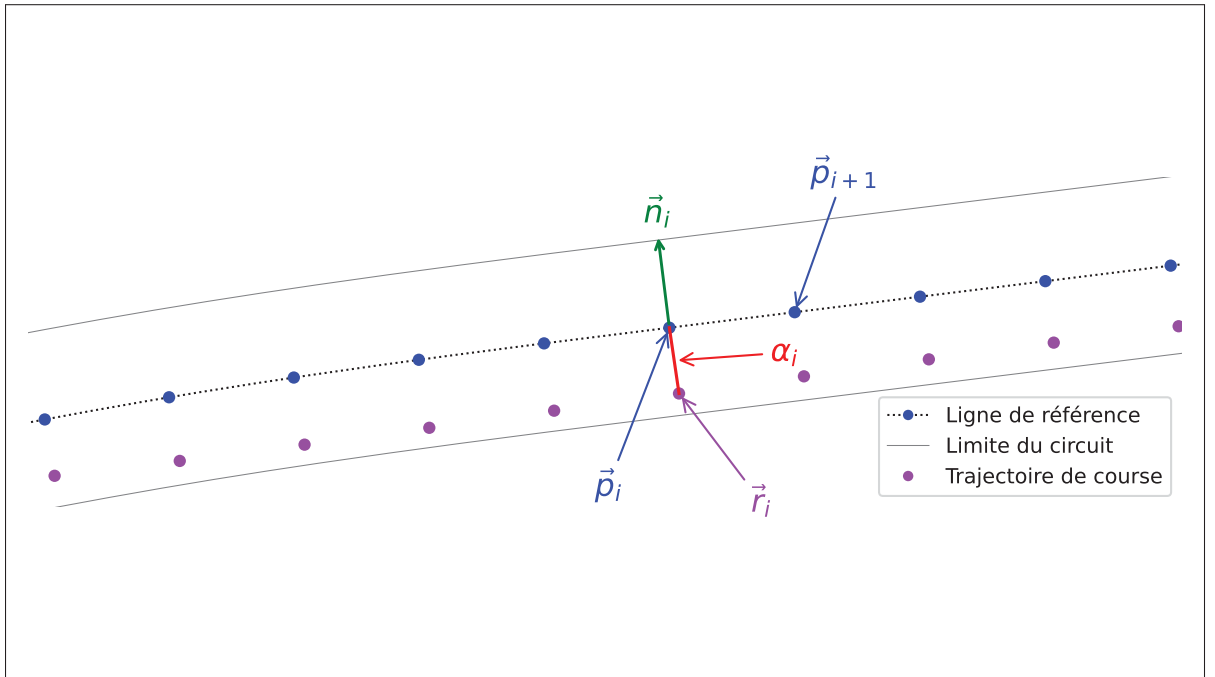


Figure 1.5 Représentation d'une trajectoire de course

1.2.2 Localisation sur un circuit

Afin de repérer le véhicule sur le circuit le système de coordonnées cartésien est remplacé par un système de coordonnées curvilignes avec un repère de Serret-Frenet.

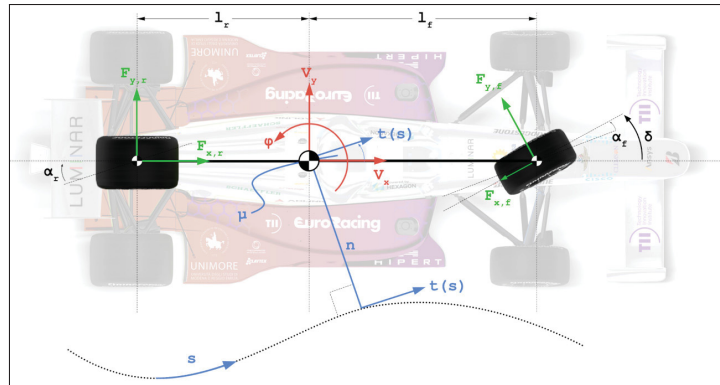


Figure 1.6 Modèle de bicyclette dans un repère de Serret-Frenet
Tirée de Raji *et al.* (2022)

Plutôt que de travailler avec la position cartésienne (x, y) du véhicule, on utilise les coordonnées curvilignes, en accord avec la Figure 1.6, (s, n, μ) où :

- s est la distance le long de la ligne centrale,
- n est la déviation latérale par rapport à la ligne centrale,
- μ est l'angle de cap par rapport à $t(s)$ la tangente de la ligne centrale en s et également l'angle entre v_x et $t(s)$.

CHAPITRE 2

INFRASTRUCTURE LOGICIELLE

Ce chapitre décrit l’architecture logicielle utilisée pour ce mémoire.

2.1 Ressources informatiques

Toutes les expériences ont été mises en œuvre sur un ordinateur ayant les caractéristiques suivantes :

- Système d’exploitation : Ubuntu 22.04.5 LTS
- Processeur : Intel Core i7-12700K
- GPU : NVIDIA GeForce RTX 5080 (16 Go de mémoire vidéo)
- Mémoire RAM : 64 Go DDR4

2.2 Bibliothèques Python utilisées

Pour le projet, Python 3.12.12 a été utilisé. Le tableau 2.1 présente les bibliothèques Python utilisées et leur version.

Tableau 2.1 Bibliothèques Python utilisées

Librairie	Version
numpy	2.3.5
gymnasium	1.2.3
CasADi	3.7.2
OSQP	1.1.0
IPOPT	3.14.11
MLflow	3.8.1
CleanRL	1.0.0

2.3 F1TENTH Gym

F1TENTH Gym (O’Kelly *et al.*, 2020; Babu & Behl, 2020) est un simulateur physique en 2D qui modélise le véhicule F1TENTH. Le simulateur propose plusieurs modèles de véhicule, y

compris les modèles de bicyclette cinématique et dynamique décrits à la section 1.1. Également, il simule des scans de LiDAR 2D et fournit des informations sur l'état dynamique du véhicule. L'environnement de simulation est fonctionnel mais limité. Plusieurs modifications ont été nécessaires, c'est pourquoi le dépôt GitHub `f1tenth_gym` (O'Kelly *et al.*, 2020; Babu & Behl, 2020) a été dupliqué (*fork*) à partir du commit `bba67a8` datant du 14 octobre 2025. Les modifications décrites dans les sous-sections suivantes ont été versionnées dans un dépôt GitHub privé.

2.3.1 Gestion des dépendances

Le gestionnaire de paquets Python standard `pip` a été remplacé par `uv`. Ce choix est motivé par l'installation d'un environnement virtuel Python et une résolution des dépendances plus rapide. Afin d'assurer une compatibilité avec les bibliothèques d'apprentissage profond et par renforcement, la version minimale de Python requise est passée de 3.9 à 3.12.

2.3.2 Modèles de véhicule

F1TENTH Gym prend en charge quatre modèles dynamiques de véhicules de fidélité croissante :

1. un modèle de bicyclette cinématique,
2. un modèle de bicyclette dynamique avec un modèle de pneu linéaire,
3. un modèle de bicyclette dynamique avec la formule magique de Pacejka,
4. un modèle multi-corps.

Le modèle multi-corps est paramétré avec les dimensions d'une voiture de taille réelle et ne simule donc pas un F1TENTH. L'implémentation par les développeurs originels du modèle de bicyclette dynamique avec la formule magique de Pacejka entraîne des interruptions inattendues de la simulation et n'a donc pas été retenue. Ainsi le modèle de bicyclette dynamique avec un modèle de pneus linéaire a été choisi. Par ailleurs, c'est le modèle par défaut.

2.3.3 Circuits

Un circuit est représenté comme décrit à la section 1.2.1. F1TENTH Gym propose différents circuits de Formule 1 qui ont été réadaptés en réduisant leur taille pour être utilisés avec le modèle 1/10.

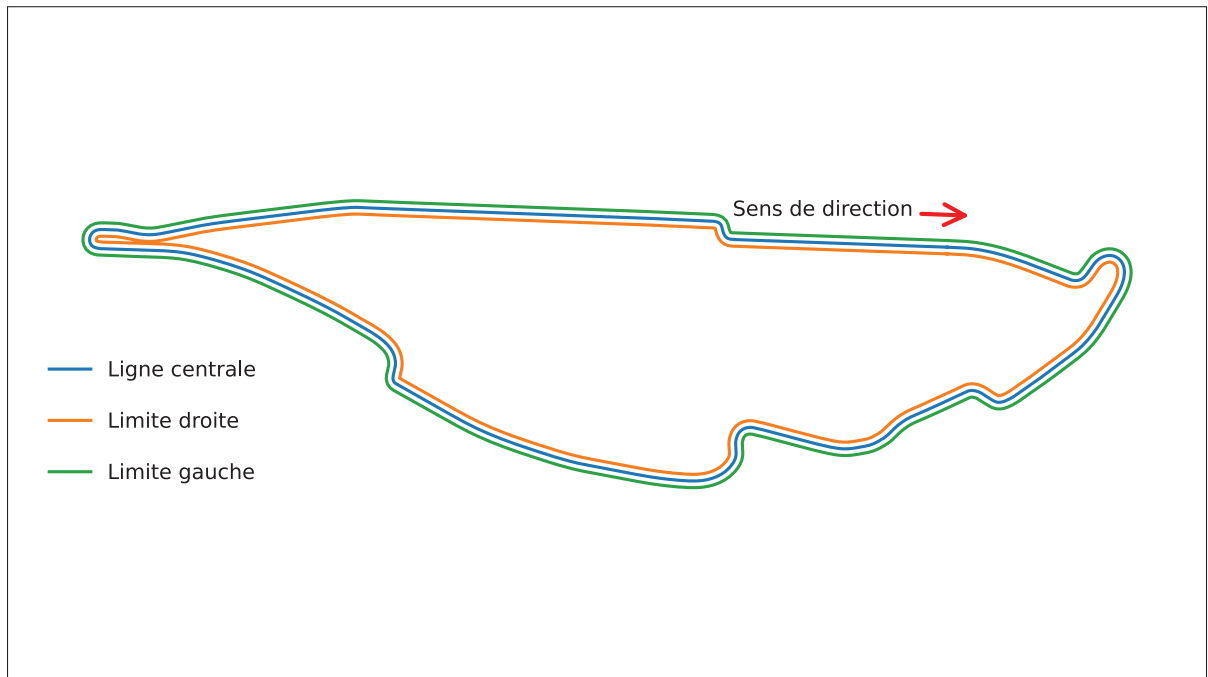


Figure 2.1 Circuit Gilles Villeneuve à Montréal réadapté pour F1TENTH Gym

Par exemple, le circuit de Montréal réadapté pour F1TENTH Gym a une longueur de 280 mètres.

2.3.4 Espace d'observation

L'espace d'observation définit les informations disponibles à chaque pas de temps. Il est possible de configurer un LiDAR 2D avec un nombre de faisceaux et une plage angulaire configurables. Les scans LiDAR résultants sont disponibles comme observations sous forme d'un vecteur dont la longueur est le nombre de faisceaux et dont les valeurs sont les distances mesurées par chaque faisceau. Les états dynamiques du véhicule décrits dans la section 1.1.2 sont également disponibles comme observations.

F1TENTH Gym a été modifiée pour permettre d’obtenir la position curviligne du véhicule (Section 1.2.2). On peut obtenir la distance parcourue le long de la ligne centrale de référence entre deux pas de temps consécutifs, c’est-à-dire : $\Delta s = s_{t+1} - s_t$.

Une vision simulée du circuit qu’on appellera *look ahead* a été implémentée. Ce concept a déjà été utilisé dans d’autres simulateurs de course par Wurman *et al.* (2022) et Subosits, Lee, Manuel, Tylkin & Balachandran (2024). Le *look ahead* consiste à obtenir dans le référentiel de la voiture en coordonnées cartésiennes des points régulièrement espacés de la ligne centrale et des bordures du circuit comme représentés dans la Figure 2.2.

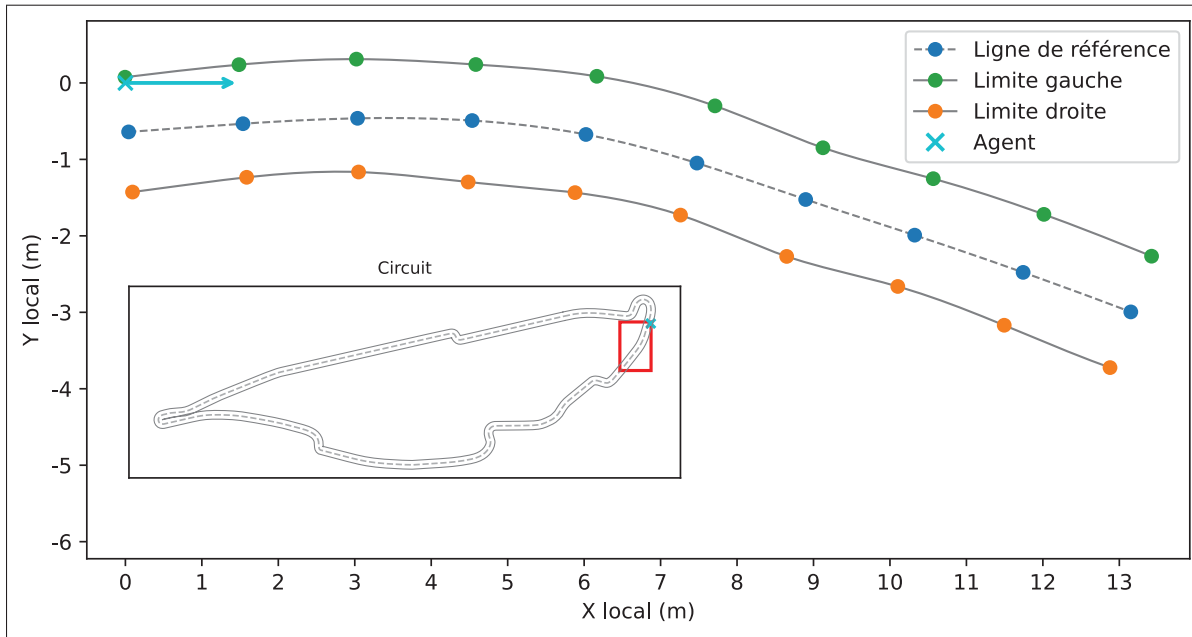


Figure 2.2 *Look ahead* observable

2.3.5 Espace d’action

Il existe deux catégories d’actions, les actions de direction et les actions longitudinales qu’il faut configurer à l’initialisation de la simulation. Il faut choisir à la fois un type d’action de direction et un type d’action longitudinale.

Les actions de direction peuvent être :

- l'angle de direction avant $\delta \in [-\delta_{\max}, \delta_{\max}]$,
- la vitesse angulaire de changement de direction avant $\dot{\delta} \in [-\dot{\delta}_{\max}, \dot{\delta}_{\max}]$.

Les actions longitudinales peuvent être :

- la vitesse longitudinale $v_{\text{cmd}} \in [v_{\min}, v_{\max}]$,
- l'accélération longitudinale $a \in [a_{\min}, a_{\max}]$.

Il est à noter que $a_{\min}$ et $v_{\min}$ peuvent être négatifs.

2.3.6 Exécution d'un épisode

On appelle un épisode une instance de simulation débutant par l'initialisation de la simulation suivie d'une exécution d'un nombre de pas de temps jusqu'à ce qu'une condition de terminaison ou de troncation entraîne la fin de l'épisode.

2.3.6.1 Initialisation d'épisode

En notant r_0 la position initiale du véhicule, on peut initialiser l'épisode en plaçant le véhicule dans différentes positions.

Soit de manière aléatoire sur le circuit : $\exists i \in \{0, \dots, N\}, r_0 = n_i$. Soit sur la ligne de départ : $r_0 = n_0$.

On peut aussi décider d'ajouter un déplacement latéral aléatoire à la position initiale $\alpha_0 \in [-w_i^{\text{gauche}}, w_i^{\text{droite}}]$.

L'orientation initiale est choisie de façon à correspondre à la tangente de la ligne centrale en r_0 .

Une fonctionnalité permettant de démarrer la voiture avec une vitesse non nulle a été implémentée en échantillonnant uniformément une plage de vitesse initiale $v_0 \in [v_{0,\min}, v_{0,\max}]$.

2.3.6.2 Exécution d'un pas de temps

Selon la configuration de la simulation, certaines des observations décrites dans la section 2.3.4 sont fournies à un contrôleur externe comme ceux présentés dans les chapitres 4 et 5. Une fois que le contrôleur a choisi les actions de direction et longitudinale, le simulateur effectue un pas de temps Δt en mettant à jour l'état du véhicule par un schéma d'intégration numérique de Runge-Kutta d'ordre 4 (Hairer & Wanner, 2015). Le simulateur a été modifié pour découpler l'horloge de la simulation de l'horloge du véhicule. La simulation peut être configurée pour interagir avec un contrôleur à une fréquence de 10 Hz tandis que le modèle du véhicule est intégré à une fréquence de 100 Hz. Ceci permet d'assurer une stabilité numérique de la simulation lorsque le contrôleur interagit à une faible fréquence.

2.3.6.3 Fin d'épisode

On définit deux modes de fin d'épisode, une terminaison et une troncation.

Une terminaison est définie lorsque le véhicule se retrouve dans une situation hors course et est généralement indésirable. Les conditions de terminaison sont décrites ci-dessous.

L'angle de la voiture dans le repère de Frenet est supérieur à un seuil, on considère que la voiture a fait un tête-à-queue.

Initialement, lorsque la voiture fait une sortie de piste l'épisode est terminé. Cependant, la simulation a été modifiée pour permettre d'activer la terminaison seulement si un nombre de pas de temps consécutifs hors piste est atteint.

On a également rajouté un nombre maximum de pas de temps consécutifs où le progrès observé est en dessous d'un seuil, ceci permet d'éviter que la voiture aille en arrière ou reste immobile.

Une troncation est définie lorsque le véhicule a effectué un nombre maximum de pas de temps. Cette condition est utilisée pour éviter que la voiture roule indéfiniment.

2.4 Suivi des expériences

MLflow (Zaharia *et al.*, 2018) est un framework open-source pour le suivi des expériences d'apprentissage automatique et le versionnement des modèles.

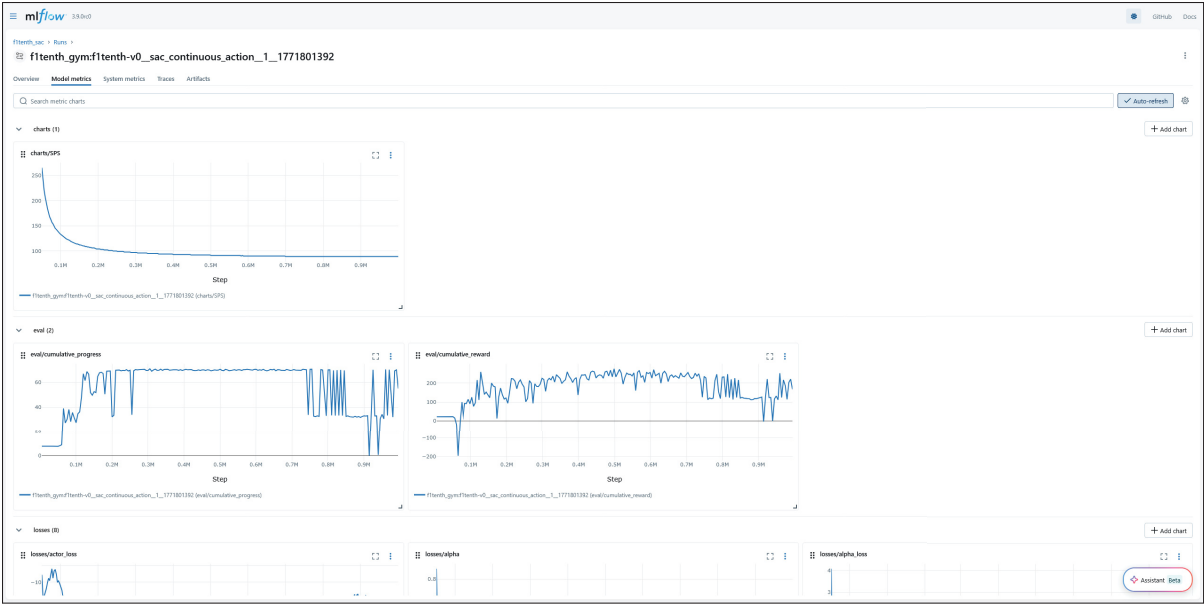


Figure 2.4 Métriques d’une expérience suivie avec MLflow

CHAPITRE 3

PLANIFICATION DE TRAJECTOIRE

La planification de trajectoire pour la course vise à trouver la trajectoire qui minimise le temps au tour tout en respectant les contraintes de la dynamique du véhicule et les limites de la piste. Contrairement à la planification de trajectoire pour les véhicules routiers, la planification pour la course cible explicitement les limites de performance du véhicule en traitant le cercle de friction (équation 1.10) comme une contrainte à atteindre plutôt que comme une marge de sécurité.

Ce chapitre présente un planificateur qui découple la trajectoire géométrique du profil de vitesse, d'après la formulation de Heilmeier *et al.* (2020). On calcule d'abord une trajectoire de course à courbure minimale, puis on lui associe un profil de vitesse respectant les limites dynamiques du véhicule. Ce chapitre situe d'abord cette approche par rapport à d'autres travaux connexes (section 3.1), introduit ensuite ses fondements théoriques (section 3.2), puis présente sa mise en œuvre et les résultats obtenus sur des circuits de course (section 3.3).

3.1 Travaux connexes

La planification de trajectoire dans un contexte de course automobile a fait l'objet d'études approfondies, couvrant un spectre allant des formulations géométriques découplées aux problèmes de commande optimale entièrement couplés (Betz *et al.*, 2022).

3.1.1 Optimisation géométrique de la ligne de course

Une première famille de méthodes optimise la géométrie de la trajectoire indépendamment du profil de vitesse. Heilmeier *et al.* (2020) minimisent l'intégrale de la courbure au carré le long de la trajectoire de course au moyen d'un programme quadratique. Cette formulation fournit l'optimum du problème quadratique linéarisé, tout en restant peu coûteuse en calcul, mais ne constitue pas une solution globale du problème à temps minimal. Leur méthode a été validée sur un Roborace DevBot à des vitesses supérieures à 150 km/h, atteignant des temps au tour à moins de 0,1 s de ceux d'un pilote professionnel. Xue, Yue & Dolan (2023) proposent une

formulation du problème à courbure minimale sur des B-splines, en ne prenant comme variables de décision que les points de contrôle de la spline. Ils atteignent une optimalité comparable avec une réduction de 90 % de la dimension du problème, ramenant le temps de calcul de quelques secondes à quelques millisecondes et ouvrant la voie à une adaptation en ligne.

3.1.2 Approches découplées trajectoire-vitesse

D'autres travaux formalisent explicitement le découplage entre le tracé et la vitesse. Kapania, Subosits & Christian Gerdes (2016) proposent un algorithme séquentiel en deux étapes : un schéma d'intégration aller-retour détermine d'abord le profil de vitesse minimisant le temps sous contrainte du cercle de friction, puis un problème d'optimisation convexe met à jour le tracé en minimisant la courbure résultante. Les deux étapes sont itérées jusqu'à convergence du temps au tour. Cette structure découplée, qui sépare la résolution d'un problème de tracé et d'un problème de vitesse plus simples qu'un unique problème de commande optimale, s'inscrit dans la même logique de découplage que celle adoptée dans ce mémoire. La différence principale est que le présent travail suit la méthodologie de Heilmeier *et al.* (2020). Le tracé géométrique à courbure minimale est d'abord calculé hors ligne, puis le profil de vitesse est construit sur la trajectoire fixée.

3.1.3 Formulations couplées avec minimisation du temps

À l'opposé, les approches par commande optimale optimisent conjointement la trajectoire et les commandes. Perantoni & Limebeer (2014) résolvent le problème de commande optimale à temps minimal pour une voiture de Formule 1 par transcription directe et programmation non linéaire, en optimisant simultanément la trajectoire, les commandes du pilote et les paramètres de réglage du véhicule. Scheffe, Henneken, Kloock & Alrifaae (2023) proposent une programmation convexe séquentielle pour la planification en temps réel, équilibrant optimalité et coût de calcul, tandis que Novi, Liniger, Capitani & Annicchiarico (2020) abordent la commande en temps réel à partir d'une trajectoire optimale prédéfinie. Ces formulations couplées produisent des

solutions quasi optimales, mais à un coût de calcul élevé, ce qui motive le recours à une approche découplée pour une planification globale et hors ligne.

3.1.4 Approches alternatives

Des méthodes inspirées par les techniques de pilotes humains ou fondées sur une recherche heuristique ont également été explorées. Theodosis & Gerdes (2011) construisent la trajectoire de course à partir d'une famille de courbes simples (droites, clothoïdes et arcs de cercle) inspirées des manœuvres des pilotes professionnels, alors que Cardamone, Loiacono, Lanzi & Bardelli (2010) recherchent la trajectoire optimale au moyen d'algorithmes génétiques.

3.1.5 Contexte en F1TENTH

Sur la plateforme F1TENTH, Evans *et al.* (2024) comparent des méthodes classiques et dirigées par la donnée dans un cadre de référence commun. Leur évaluation montre que la séquence d'optimisation de trajectoire puis de suivi de trajectoire obtient les meilleurs temps au tour parmi les méthodes testées, devant les approches de planification locale et les méthodes réactives comme *Follow-the-Gap* (Sezer & Gokasan, 2012). Ce constat motive le choix d'une trajectoire de course optimisée hors ligne, servant de référence globale pour les contrôleurs en aval.

3.2 Fondements théoriques

L'approche de planification à courbure minimale minimise l'intégrale de la courbure au carré le long de la trajectoire de course (Heilmeier *et al.*, 2020). Intuitivement, une trajectoire de plus faible courbure nécessite moins d'accélération latérale pour être suivie, ce qui autorise des vitesses plus élevées. Cette section détaille le passage d'une représentation discrète du circuit à une trajectoire continue par interpolation, l'objectif d'optimisation, sa formulation en programme quadratique, l'affinement itératif, puis la génération du profil de vitesse.

3.2.1 D'une représentation discrète à une trajectoire continue

Le chapitre 1 (section 1.2.1) a introduit la représentation d'un circuit : la ligne centrale est décrite par une séquence discrète de points $\{\vec{p}_i\}_{i=1}^N$, annotés des largeurs de piste w_i^{gauche} et w_i^{droite} et des vecteurs normaux $\vec{n}_i$, et une trajectoire de course y est paramétrée par des déviations latérales α_i résultants en $\vec{r}_i = \vec{p}_i + \alpha_i \vec{n}_i$ (équation 1.12).

Cette représentation est toutefois discrète, alors qu'une trajectoire de course est une courbe continue. Or l'objectif d'optimisation porte sur la courbure, qui fait intervenir les dérivées première et seconde de la trajectoire. L'évaluer de façon fiable à partir de points isolés exige donc de reconstruire une courbe continue et dérivable passant par ces points. On interpole pour cela la ligne centrale par des splines cubiques, c'est-à-dire des polynômes de degré trois $\mathbb{R}_3[X]$, définis par morceaux entre les points consécutifs et raccordés de manière à assurer la continuité jusqu'à la dérivée seconde. Cette continuité C^2 garantit en particulier une courbure continue le long du parcours, contrairement à une interpolation linéaire qui produirait une courbure discontinue et nulle par morceaux comme le montre la Figure 3.1.

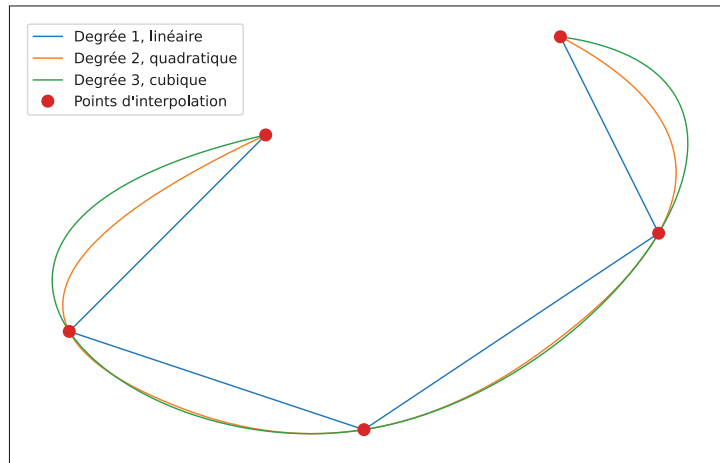


Figure 3.1 Interpolation ouverte avec trois splines de degrés un à trois entre cinq points

L'intuition est assez claire : pour toute trajectoire donnée, il faut pouvoir y associer un profil de vitesse et d'accélération. Or la vitesse et l'accélération sont respectivement les dérivées première

et seconde de la position d'où la nécessité d'une continuité C^2 ce qui motive le choix d'une interpolation par splines cubiques.

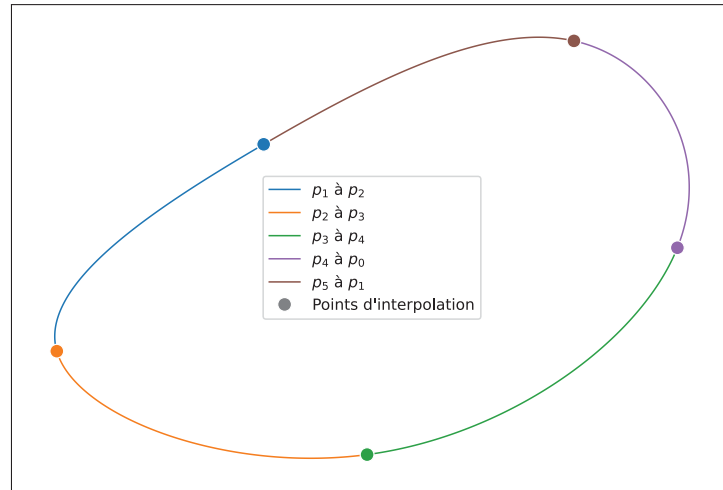


Figure 3.2 Interpolation fermée par splines cubiques entre 5 points

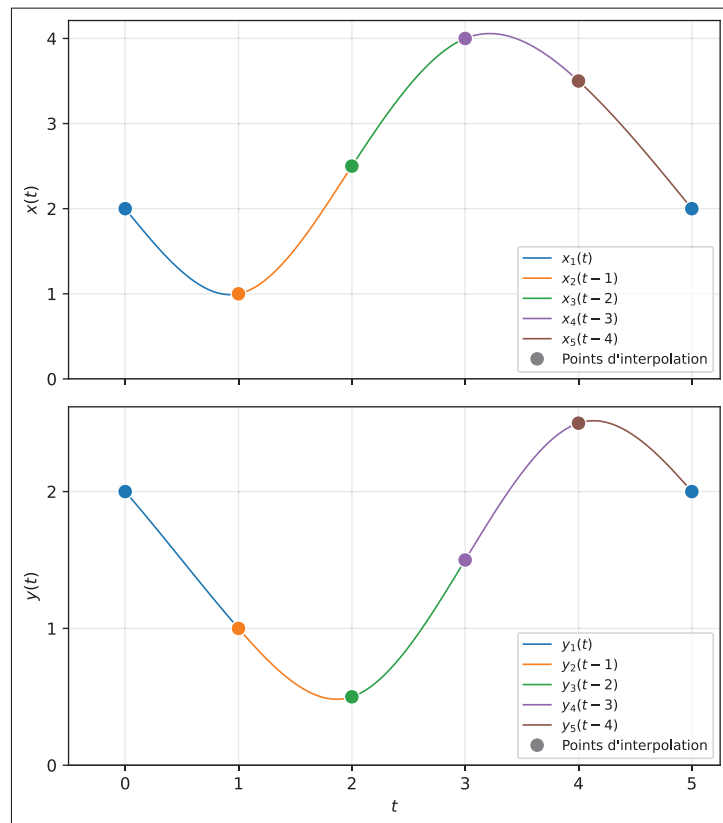


Figure 3.3 Fonctions $x_i(t)$ et $y_i(t)$ des cinq splines cubiques de la Figure 3.2

Pour N points, on cherche à déterminer N segments cubiques qui forment une boucle fermée comme illustré dans la Figure 3.2. On paramétrise un segment i par une longueur d'arc normalisée $t \in [0, 1]$. Cette spline est décrite par deux fonctions : $x_i(t), y_i(t) \in C^2([0, 1], \mathbb{R})$ (Figure 3.3).

Puis on construit les fonctions $x(t), y(t) \in C^2([0, N], \mathbb{R})$ en utilisant les fonctions $x_i(t), y_i(t)$:

$$\forall (i, t) \in \llbracket 1, N \rrbracket \times [0, 1], \begin{cases} x(i - 1 + t) &= x_i(t) \\ y(i - 1 + t) &= y_i(t) \end{cases} \quad (3.1)$$

Notons que $\forall i \in \llbracket 1, N - 1 \rrbracket, x(i) = x_i(1) = x_{i+1}(0)$ et $y(i) = y_i(1) = y_{i+1}(0)$.

En tout point l'expression de l'angle de cap $\theta(t)$ et la courbure de référence $\kappa(t)$ sont :

$$\begin{aligned} \theta(t) &= \text{atan2}(y'(t), x'(t)), \\ \kappa(t) &= \frac{x'(t) y''(t) - y'(t) x''(t)}{(x'(t)^2 + y'(t)^2)^{3/2}} \end{aligned} \quad (3.2)$$

où ' désigne la dérivation par rapport à t .

Les vecteurs normaux $\vec{n}_i$ servent à déplacer latéralement la trajectoire découlant de la tangente $(x'(t), y'(t))$ de la spline, et les largeurs de piste sont interpolées le long de t afin de rester cohérentes avec cette paramétrisation. La courbe interpolée est ré-échantillonnée à un pas de longueur d'arc choisi, ce qui forme la grille uniforme de points sur laquelle porte l'optimisation (3.2.3.5). Une marge de sécurité est soustraite des largeurs de piste, w_i^{gauche} et w_i^{droite} , pour tenir compte des dimensions physiques du véhicule et garder le véhicule à l'intérieur de la piste (3.2.3.6). Cette interpolation par splines est réappliquée à chaque itération d'affinement (3.2.4) pour recalculer la courbure de la trajectoire générée (Heilmeier *et al.*, 2020).

3.2.2 Objectif d'optimisation à courbure minimale

Les variables de décision sont les déviations latérales $\alpha = [\alpha_1, \dots, \alpha_N]^\top$ introduites au chapitre 1, qui positionnent les points de la trajectoire de course $\vec{r}_i = \vec{p}_i + \alpha_i \vec{n}_i$ (équation 1.12). L'objectif

est de minimiser la somme des courbures au carré le long des N points de la trajectoire :

$$\min_{\alpha} \sum_{i=1}^N \kappa_i^2(\alpha) \quad (3.3)$$

La difficulté vient du fait que la courbure κ_i (équation 3.2) est une fonction non linéaire des positions des points, donc de α , ce qui empêche de formuler directement l'équation 3.3 comme un programme quadratique. Une formulation en programme quadratique convexe est néanmoins recherchée pour sa rapidité et sa robustesse de résolution : sa convexité garantit la convergence vers un optimum global contrairement à un programme non linéaire général, dont la résolution est plus lente et sujette à des optima locaux (Heilmeier *et al.*, 2020). Cette propriété est d'autant plus importante que le raffinement itératif présenté plus loin (3.2.4) requiert la résolution répétée du problème. La suite montre comment l'interpolation par splines rend les dérivées secondes affines en α . En gelant les dérivées premières autour d'une ligne de référence, on obtient alors une approximation quadratique de l'équation 3.3, résolue sous forme de programme.

3.2.3 Formulation du programme quadratique

L'enjeu est d'exprimer une approximation de la courbure sous une forme quadratique en α . Les calculs en x et en y étant identiques, on détaille la construction pour la composante x , la composante y s'en déduit à l'identique. On indexe les N points du circuit fermé de manière cyclique. Dans l'implémentation, la ligne de référence est d'abord ré-échantillonnée avec un pas de longueur d'arc uniforme, ce qui permet d'utiliser un paramètre local normalisé $t \in [0, 1]$ sur chaque segment de spline.

3.2.3.1 Système des coefficients de spline M

Le segment i qui relie deux points s'écrit pour un paramètre local $t \in [0, 1]$:

$$x_i(t) = a_{i,0} + a_{i,1}t + a_{i,2}t^2 + a_{i,3}t^3, \quad (3.4)$$

Notons que :

$$\begin{aligned}
 x'_i(t) &= a_{i,1} + 2a_{i,2}t + 3a_{i,3}t^2 \\
 x'_i(0) &= a_{i,1} \\
 x'_i(1) &= a_{i,1} + 2a_{i,2} + 3a_{i,3} \\
 x''_i(t) &= 2a_{i,2} + 6a_{i,3}t \\
 x''_i(0) &= 2a_{i,2} \\
 x''_i(1) &= 2a_{i,2} + 6a_{i,3}
 \end{aligned} \tag{3.5}$$

Soit $i \in \llbracket 1, N \rrbracket$. Le segment i relie les nœuds $p_i = (p_{i,x}, p_{i,y})$ et $p_{i+1} = (p_{i+1,x}, p_{i+1,y})$.

Pour le cas $i = N$, il suffit de considérer qu'on étend la liste de points avec $p_{N+1} = p_1$.

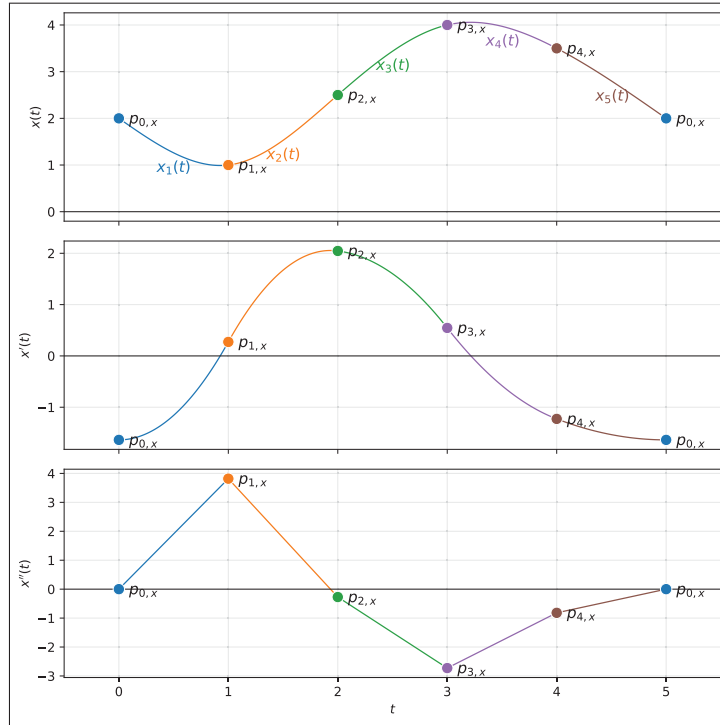


Figure 3.4 Fonctions $x(t)$, $x'(t)$ et $x''(t)$

Les quatre conditions imposées au segment i sont :

$$\underbrace{a_{i,0}}_{x_i(0)} = p_{i,x} \quad (3.6)$$

$$\underbrace{a_{i,0} + a_{i,1} + a_{i,2} + a_{i,3}}_{x_i(1)} = p_{i+1,x} \quad (3.7)$$

$$\underbrace{a_{i,1} + 2a_{i,2} + 3a_{i,3}}_{x'_i(1)} = \underbrace{a_{i+1,1}}_{x'_{i+1}(0)} \quad (3.8)$$

$$\underbrace{2a_{i,2} + 6a_{i,3}}_{x''_i(1)} = \underbrace{2a_{i+1,2}}_{x''_{i+1}(0)} \quad (3.9)$$

Ces conditions sont illustrées dans la Figure 3.4. Les conditions 3.6 et 3.7 correspondent aux deux nœuds que le segment i relie. Les conditions 3.8 et 3.9 permettent d'assurer la continuité des dérivées première et seconde au nœud i entre les segments i et $i + 1$.

En réécrivant ces conditions sous forme matricielle, on obtient :

$$\begin{cases} a_{i,0} & = p_{i,x} \\ a_{i,0} + a_{i,1} + a_{i,2} + a_{i,3} & = p_{i+1,x} \\ a_{i,1} + 2a_{i,2} + 3a_{i,3} - a_{i+1,1} & = 0 \\ 2a_{i,2} + 6a_{i,3} - 2a_{i+1,2} & = 0 \end{cases} \quad (3.10)$$

$$\Leftrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & -1 & 0 & 0 \\ 0 & 0 & 2 & 6 & 0 & 0 & -2 & 0 \end{bmatrix} \begin{bmatrix} a_{i,0} \\ a_{i,1} \\ a_{i,2} \\ a_{i,3} \\ a_{i+1,0} \\ a_{i+1,1} \\ a_{i+1,2} \\ a_{i+1,3} \end{bmatrix} = \begin{bmatrix} p_{i,x} \\ p_{i+1,x} \\ 0 \\ 0 \end{bmatrix}$$

Il y a N segments avec chacun quatre coefficients à déterminer. Pour pouvoir les déterminer, on généralise l'équation 3.10 aux N segments et on construit le système d'équations suivant :

$$\begin{aligned}
 Ma_x &= b_x \\
 \text{tel que} \\
 M &\in \mathbb{R}^{4N \times 4N} \\
 a_x &\in \mathbb{R}^{4N} \\
 b_x &\in \mathbb{R}^{4N}
 \end{aligned} \tag{3.11}$$

On identifie alors les matrices :

$$\begin{aligned}
 A_{t,1} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & 6 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \\
 A_{t,2} &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -2 & 0 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \\
 a_x &= [a_{1,0}, a_{1,1}, a_{1,2}, a_{1,3}, \dots, a_{N,0}, a_{N,1}, a_{N,2}, a_{N,3}]^T \in \mathbb{R}^{4N} \\
 b_x &= [p_{1,x}, p_{1+1,x}, 0, 0, \dots, p_{N,x}, p_{N+1,x}, 0, 0]^T \in \mathbb{R}^{4N}
 \end{aligned} \tag{3.12}$$

$A_{t,1}$ et $A_{t,2}$ correspondent à une découpe de la matrice de l'équation 3.10 en deux blocs. La matrice M est formée de $A_{t,1}$ et $A_{t,2}$ portant respectivement sur les coefficients du segment i et du segment $i + 1$ et présente une structure circulante par blocs. Le bloc $A_{t,2}$ en bas à gauche

assure la fermeture de la boucle du point N au point 1 :

$$M = \begin{bmatrix} A_{t,1} & A_{t,2} & & & \\ & A_{t,1} & A_{t,2} & & \\ & & \ddots & \ddots & \\ & & & A_{t,1} & A_{t,2} \\ A_{t,2} & & & & A_{t,1} \end{bmatrix} \in \mathbb{R}^{4N \times 4N}. \quad (3.13)$$

Par le théorème 0.1, la matrice M est inversible pour tout $N \geq 2$. Ainsi, le système $Ma_x = b_x$ admet une et une unique solution. Géométriquement, il existe une spline cubique qui représente la trajectoire de course passant par tous les points p_i et elle est unique.

Les coefficients s'obtiennent par $a_x = M^{-1}b_x$. Rappelons que par construction :

- $a_x[4i : 4i + 4] = [a_{i,0}, a_{i,1}, a_{i,2}, a_{i,3}]^\top$ correspondent aux coefficients de x_i
- $b_x[4i : 4i + 4] = [p_{i,x}, p_{i+1,x}, 0, 0]^\top$ correspondent aux coordonnées en x des points p_i et p_{i+1} de la spline i .

Le problème suivant est de déterminer la valeur des éléments de b_x puisqu'ils correspondent aux coordonnées en x des points de la trajectoire de course. Le problème sous-jacent est de déterminer ces valeurs en fonction des déviations latérales α .

3.2.3.2 Second membre et dépendance en α

En considérant le point i de la ligne de référence, on note x_i sa coordonnée en x et $n_{x,i}$ la coordonnée en x du vecteur normal au point i .

Grâce aux conditions 3.6 et 3.7, on a que seules les deux premières lignes de chaque bloc de b_x

$$\text{sont non nulles : } b_x[4i : 4i + 4] = \begin{bmatrix} p_{i,x} \\ p_{i+1,x} \\ 0 \\ 0 \end{bmatrix}.$$

Par 1.12, on a que $p_{i,x} = x_i + n_{x,i}\alpha_i$.

$$\text{Donc, on a que } b_x[4i : 4i + 4] = \begin{bmatrix} x_i + n_{x,i}\alpha_i \\ x_{i+1} + n_{x,i+1}\alpha_{i+1} \\ 0 \\ 0 \end{bmatrix}.$$

On identifie alors la structure de b_x en séparant la contribution en courbure de la ligne de référence x_i à celles des déviations latérales $n_{x,i}\alpha_i$:

$$b_x = q_x + M_x \alpha,$$

$$q_x = [x_1, x_2, 0, 0, \dots, x_{N-1}, x_N, 0, 0, x_N, x_1, 0, 0]^\top \in \mathbb{R}^{4N}$$

$$M_x \alpha = [n_{x,1} \alpha_1, n_{x,2} \alpha_2, 0, 0, \dots, n_{x,N-1} \alpha_{N-1}, n_{x,N} \alpha_N, 0, 0, n_{x,N} \alpha_N, n_{x,1} \alpha_1, 0, 0]^\top \in \mathbb{R}^{4N}$$

$$\text{où } M_x = \begin{bmatrix} n_{x,1} & 0 & 0 & \cdots & 0 \\ n_{x,2} & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & n_{x,2} & 0 & \cdots & 0 \\ 0 & n_{x,3} & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & n_{x,N} \\ 0 & 0 & 0 & \cdots & n_{x,1} \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{4N \times N} \quad (3.14)$$

q_x contient les abscisses de la ligne de référence et $M_x \alpha$ contient les déplacements latéraux selon les normales des points de la ligne de référence.

On peut maintenant décrire avec α une trajectoire interpolée par splines cubiques.

3.2.3.3 Matrices d'extraction et dérivées affines

Alors que 3.2.3.2 expose la dépendance en α des points d'interpolation des splines, la suite expose la dépendance en α de la courbure en différents points de la trajectoire. La définition de la courbure en un point donné (3.2) dépend à la fois de la dérivée première et seconde de la spline. Il faut donc déterminer une expression de ces dérivées en fonction de α . Par l'équation 3.5, on sait que la dérivée première au point i est $a_{i,1}$ et la dérivée seconde au point i est $2a_{i,2}$.

Posons :

$$\begin{aligned}
 x' &= [a_{1,1}, a_{2,1}, \dots, a_{N,1}]^\top \in \mathbb{R}^N, \text{ le vecteur des dérivées premières} \\
 x'' &= [2a_{1,2}, 2a_{2,2}, \dots, 2a_{N,2}]^\top \in \mathbb{R}^N, \text{ le vecteur des dérivées secondes}
 \end{aligned}$$

$$\begin{aligned}
 A_{\text{ex},b} &= \begin{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} & \mathbf{0}_{1 \times 4} & \cdots & \mathbf{0}_{1 \times 4} \\ \mathbf{0}_{1 \times 4} & \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0}_{1 \times 4} \\ \mathbf{0}_{1 \times 4} & \cdots & \mathbf{0}_{1 \times 4} & \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \end{bmatrix} \in \mathbb{R}^{N \times 4N} \\
 A_{\text{ex},c} &= \begin{bmatrix} \begin{bmatrix} 0 & 0 & 2 & 0 \end{bmatrix} & \mathbf{0}_{1 \times 4} & \cdots & \mathbf{0}_{1 \times 4} \\ \mathbf{0}_{1 \times 4} & \begin{bmatrix} 0 & 0 & 2 & 0 \end{bmatrix} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0}_{1 \times 4} \\ \mathbf{0}_{1 \times 4} & \cdots & \mathbf{0}_{1 \times 4} & \begin{bmatrix} 0 & 0 & 2 & 0 \end{bmatrix} \end{bmatrix} \in \mathbb{R}^{N \times 4N}
 \end{aligned} \tag{3.15}$$

On a donc $x' = A_{\text{ex},b} a_x$ et $x'' = A_{\text{ex},c} a_x$. En combinant avec l'équation 3.14 et en généralisant les résultats précédents à y , on obtient les expressions affines en α :

$$\begin{aligned}
 x'(\alpha) &= A_{\text{ex},b} M^{-1} q_x + A_{\text{ex},b} M^{-1} M_x \alpha \\
 x''(\alpha) &= A_{\text{ex},c} M^{-1} q_x + A_{\text{ex},c} M^{-1} M_x \alpha \\
 y'(\alpha) &= A_{\text{ex},b} M^{-1} q_y + A_{\text{ex},b} M^{-1} M_y \alpha \\
 y''(\alpha) &= A_{\text{ex},c} M^{-1} q_y + A_{\text{ex},c} M^{-1} M_y \alpha
 \end{aligned} \tag{3.16}$$

Ces expressions montrent la contribution à la fois de la ligne de référence (premier terme) et des déviations latérales (second terme) à la dérivée première et seconde en chaque point de la trajectoire.

3.2.3.4 Forme quadratique de la courbure

En développant le carré de la courbure (équation 3.2) :

$$\begin{aligned}\kappa_i^2 &= \frac{x_i'^2 y_i''^2 - 2 x_i' y_i' x_i'' y_i'' + y_i'^2 x_i''^2}{(x_i'^2 + y_i'^2)^3} \\ \kappa_i^2 &= \frac{x_i'^2}{(x_i'^2 + y_i'^2)^3} y_i''^2 + \frac{-2 x_i' y_i'}{(x_i'^2 + y_i'^2)^3} x_i'' y_i'' + \frac{y_i'^2}{(x_i'^2 + y_i'^2)^3} x_i''^2\end{aligned}\quad (3.17)$$

L'hypothèse centrale est que la dérivée première en chaque point de la trajectoire de course est proche de celle de la ligne centrale de référence. On fait l'approximation que :

$$\forall \alpha, \begin{cases} x_i'(\alpha) \approx x_i'(0) = A_{\text{ex},b} M^{-1} q_x \\ y_i'(\alpha) \approx y_i'(0) = A_{\text{ex},b} M^{-1} q_y \end{cases} \quad (3.18)$$

Les facteurs x_i' , y_i' sont constants, alors le dénominateur de κ_i^2 est constant. κ_i^2 devient quadratique dans les seules dérivées secondes. En regroupant les trois termes dans des matrices diagonales :

$$P_{xx} = \begin{bmatrix} \frac{y_1'^2}{(x_1'^2 + y_1'^2)^3} & & & \\ & \ddots & & \\ & & \frac{y_i'^2}{(x_i'^2 + y_i'^2)^3} & \\ & & & \ddots \\ & & & & \frac{y_N'^2}{(x_N'^2 + y_N'^2)^3} \end{bmatrix}$$

$$\begin{aligned}
 P_{yy} &= \begin{bmatrix} \frac{x_1'^2}{(x_1'^2+y_1'^2)^3} & & & \\ & \ddots & & \\ & & \frac{x_i'^2}{(x_i'^2+y_i'^2)^3} & \\ & & & \ddots \\ & & & & \frac{x_N'^2}{(x_N'^2+y_N'^2)^3} \end{bmatrix} \\
 P_{xy} &= \begin{bmatrix} \frac{-2x_1'y_1'}{(x_1'^2+y_1'^2)^3} & & & \\ & \ddots & & \\ & & \frac{-2x_i'y_i'}{(x_i'^2+y_i'^2)^3} & \\ & & & \ddots \\ & & & & \frac{-2x_N'y_N'}{(x_N'^2+y_N'^2)^3} \end{bmatrix}
 \end{aligned} \tag{3.19}$$

La somme des courbures au carré prend la forme matricielle :

$$\sum_{i=1}^N \kappa_i^2 = x''^\top P_{xx} x'' + x''^\top P_{xy} y'' + y''^\top P_{yy} y'' \tag{3.20}$$

3.2.3.5 Formulation du programme quadratique sans contrainte

En substituant les expressions affines des équations 3.16 dans l'équation 3.20 et en ne conservant que les termes dépendant de α , on identifie l'hessien H et le terme linéaire f :

$$\begin{aligned}
 H &= T_{nx}^\top P_{xx} T_{nx} + T_{nx}^\top P_{xy} T_{ny} + T_{ny}^\top P_{yy} T_{ny} \\
 f &= 2 T_{nx}^\top P_{xx} T_c q_x + 2 T_{ny}^\top P_{yy} T_c q_y + T_{nx}^\top P_{xy} T_c q_y + T_{ny}^\top P_{xy} T_c q_x \\
 \text{où : } T_{nx} &= A_{\text{ex},c} M^{-1} M_x \\
 T_{ny} &= A_{\text{ex},c} M^{-1} M_y
 \end{aligned} \tag{3.21}$$

Pour corriger les erreurs numériques, l'hessien H est symétrisé : $H \leftarrow \frac{1}{2}(H + H^\top)$. Le problème original 3.3 devient le problème d'optimisation quadratique non contraint :

$$\min_{\alpha} \quad \frac{1}{2} \alpha^\top H \alpha + f^\top \alpha \quad (3.22)$$

3.2.3.6 Contraintes au programme quadratique

Il faut rappeler que la trajectoire de course doit rester sur la piste. Deux familles de contraintes linéaires complètent le problème. Les limites de piste bornent chaque déviation, en réservant la demi-largeur du véhicule $W_{\text{veh}}/2$:

$$-\left(w_i^{\text{gauche}} - \frac{W_{\text{veh}}}{2}\right) \leq \alpha_i \leq w_i^{\text{droite}} - \frac{W_{\text{veh}}}{2}, \quad i = 1, \dots, N. \quad (3.23)$$

Les contraintes de courbure imposent $|\kappa_i| \leq \kappa_{\text{max}}$. La borne de courbure découle du modèle de bicycle cinématique, pour lequel, sous hypothèses de petits angles et d'absence de glissement, on a $\kappa = \tan(\delta)/L$ (Kapania & Gerdes, 2015). Dans la linéarisation faite, la courbure est approximée par une fonction affine de α , soit $\kappa \approx E_\kappa \alpha + \kappa_{\text{ref}}$ avec $E_\kappa = Q_y T_{ny} - Q_x T_{nx}$. Ces contraintes se traduisent par deux blocs d'inégalités linéaires $\pm E_\kappa \alpha \leq \kappa_{\text{max}} \mp \kappa_{\text{ref}}$, où Q_x et Q_y sont diagonales

et formées à partir des dérivées premières de la ligne de référence :

$$\begin{aligned}
 Q_x &= \begin{bmatrix} \frac{y_1'^2}{(x_1'^2 + y_1'^2)^3} & & & \\ & \ddots & & \\ & & \frac{y_i'^2}{(x_i'^2 + y_i'^2)^3} & \\ & & & \ddots \\ & & & & \frac{y_N'^2}{(x_N'^2 + y_N'^2)^3} \end{bmatrix} \\
 Q_y &= \begin{bmatrix} \frac{x_1'^2}{(x_1'^2 + y_1'^2)^3} & & & \\ & \ddots & & \\ & & \frac{x_i'^2}{(x_i'^2 + y_i'^2)^3} & \\ & & & \ddots \\ & & & & \frac{x_N'^2}{(x_N'^2 + y_N'^2)^3} \end{bmatrix}
 \end{aligned} \tag{3.24}$$

Ce qui donne :

$$\begin{aligned}
 E &= \begin{bmatrix} E_\kappa \\ -E_\kappa \end{bmatrix} \in \mathbb{R}^{2N \times N} \\
 \kappa_{limite} &= \begin{bmatrix} (\kappa_{\max} - \kappa_{\text{ref}}) \mathbf{1}_N \\ (\kappa_{\max} + \kappa_{\text{ref}}) \mathbf{1}_N \end{bmatrix} \in \mathbb{R}^{2N}
 \end{aligned} \tag{3.25}$$

En ajoutant les contraintes linéaires à 3.22, on obtient le problème d'optimisation suivant :

$$\begin{aligned}
 \min_{\alpha} \quad & \frac{1}{2} \alpha^\top H \alpha + f^\top \alpha \\
 \text{tel que} \quad & E \alpha \preccurlyeq \kappa_{limite}
 \end{aligned} \tag{3.26}$$

On a $H \in \mathbb{S}^N(\mathbb{R})$, $f \in \mathbb{R}^N$ et la contrainte pour α est linéaire, alors 3.26 est un problème d'optimisation quadratique.

Le théorème 0.3 donne H semi-définie positive, donc le programme quadratique est convexe. Par le théorème 0.6, l'ensemble de solutions est non vide, si bien qu'un minimiseur global existe : le

programme quadratique (3.26) a au moins une solution globale. Par ailleurs, le corollaire 0.6.1, nous donne un certificat de minimum global. Ce qui garantit que dans les conditions d'utilisation définies, le planificateur de trajectoire produit une trajectoire de course non seulement admissible, mais optimale. Il est toutefois important de noter qu'il n'y a pas d'unicité de solution globale.

3.2.4 Affinement itératif

Par la suite, Heilmeier *et al.* (2020) propose un affinement itératif de l'approximation de courbure. Après la résolution du QP initial (3.26), la courbure est recalculée sur la solution initiale, et les matrices du QP sont mises à jour avec une nouvelle linéarisation. Ce processus est répété pour un nombre fixe d'itérations, ce qui améliore la précision dans les sections à forte courbure où l'approximation linéaire (3.18) est la moins précise.

3.2.5 Génération du profil de vitesse

Étant donné une trajectoire de course, il reste à calculer un profil de vitesse qui respecte les limites dynamiques du véhicule.

3.2.5.1 Vitesse maximale selon la courbure

La vitesse maximale en chaque point est bornée par l'accélération latérale que la friction des pneus peut fournir, $v_i^2 |\kappa_i| \leq \mu g$ (Milliken & Milliken, 1995; Guiggiani, 2023). Ce qui donne une borne supérieure à la vitesse en chaque point :

$$\forall i \in \{1, \dots, N\}, \quad v_i \leq \sqrt{\frac{\mu g}{|\kappa_i|}} = v_{i,\max} \quad (3.27)$$

où g est l'accélération gravitationnelle et μ est le coefficient de frottement. Un schéma d'intégration aller-retour produit le profil de vitesse en appliquant les limites d'accélération et de freinage longitudinales, tout en tenant compte de la contrainte du cercle de friction qui combine les forces latérales et longitudinales.

3.2.5.2 Passe avant

À partir de chaque minimum local de vitesse, la vitesse de sortie maximale est calculée en se propageant vers l'avant avec l'accélération maximale :

$$\forall i \in \{1, \dots, N-1\}, \begin{cases} v_{i+1}^{\rightarrow} = \sqrt{v_i^{\rightarrow 2} + 2a_{\text{accel}}\Delta s_{i,i+1}} \\ v_{i+1} = \min(v_{i+1}^{\rightarrow}, v_{i+1,\text{max}}) \end{cases} \quad (3.28)$$

où a_{accel} est l'accélération maximale disponible après avoir alloué la friction latérale pour le virage, et $\Delta s_{i,i+1}$ est la longueur d'arc entre deux points consécutifs.

3.2.5.3 Passe en arrière

À partir de chaque point dont la courbure dépasse un seuil, la vitesse d'entrée maximale est calculée en se propageant vers l'arrière le long de la piste avec la décélération maximale autorisée par le cercle de friction :

$$\forall i \in \{1, \dots, N-1\}, \begin{cases} v_i^{\leftarrow} = \sqrt{v_{i+1}^{\leftarrow 2} + 2a_{\text{freinage}}\Delta s_{i,i+1}} \\ v_i = \min(v_i^{\leftarrow}, v_{i,\text{max}}) \end{cases} \quad (3.29)$$

où a_{freinage} est la décélération de freinage maximale disponible après avoir alloué la friction latérale pour le virage, et $\Delta s_{i,i+1}$ est la longueur d'arc entre deux points consécutifs.

3.2.5.4 Profil de vitesse final

Dans le modèle quasi stationnaire retenu, en chaque point la vitesse v_i respecte simultanément les contraintes de virage et d'accélération longitudinale. Ce profil reste toutefois une estimation ouverte : il ne remplace pas une validation en boucle fermée avec un contrôleur et un modèle dynamique complet.

3.3 Expérimentation

L'algorithme de planification de trajectoire est résumé par l'algorithme 3.1.

Algorithme 3.1 Algorithme de planification de trajectoire

Input : Ligne centrale $\{\vec{p}_i\}$ et limites de piste $w_i^{\text{gauche}}, w_i^{\text{droite}}$ Input : Nombre d'itérations IQP N_{IQP} Input : Paramètres du véhicule (W_{veh}, μ, accélérations) Output : Trajectoire de course $\{\vec{r}_i\}$ Output : Profil de vitesse $\{v_i\}$ 1 Charger la ligne centrale et les limites de piste; 2 Calculer la longueur d'arc s le long de la piste; 3 Calculer les angles de cap $\theta(s)$ et la courbure $\kappa(s)$; 4 Calculer les vecteurs normaux $\{\vec{n}_i\}$; 5 Construire l'hessien H à partir des coefficients de spline; 6 Construire le terme linéaire f; 7 Assembler les vecteurs de contraintes de limite de piste; 8 Résoudre le programme quadratique (3.26) avec OSQP; 9 $\alpha^* \leftarrow$ solution du QP; 10 for $k = 1, \dots, N_{\text{IQP}} - 1$ do 11 Former le chemin déplacé $\vec{p}_i + \alpha_i^* \vec{n}_i$; 12 Recalculer H et f sur ce chemin; 13 Mettre à jour les vecteurs de contraintes; 14 Résoudre le QP avec OSQP; 15 $\alpha^* \leftarrow$ nouvelle solution du QP; 16 end for 17 for $i = 1, \dots, N$ do 18 $\vec{r}_i \leftarrow \vec{p}_i + \alpha_i^* \vec{n}_i$ (équation 1.12); 19 end for 20 Initialiser v_i; 21 Calculer $v_{i,\text{max}}$ selon la limite de courbure (équation 3.27); 22 Calculer $v_i^{\rightarrow}$ et mettre à jour v_i par propagation avant (équation 3.28); 23 Calculer $v_i^{\leftarrow}$ et mettre à jour v_i par propagation arrière (équation 3.29);
--

3.3.1 Résultats

Le planificateur est évalué sur la piste de Montréal (Figure 2.1).

Les paramètres de friction et les limites d'accélération et de freinage utilisés dans la génération du profil de vitesse sont ceux du modèle de véhicule décrit au chapitre 1.

3.3.1.1 Trajectoires générées

Les figures 3.5 et 3.6 montrent les trajectoires générées avec une seule itération et quatre itérations respectivement. De façon qualitative, on peut voir que les trajectoires obtenues suivent la règle générale «extérieur — intérieur — extérieur» pour les virages.

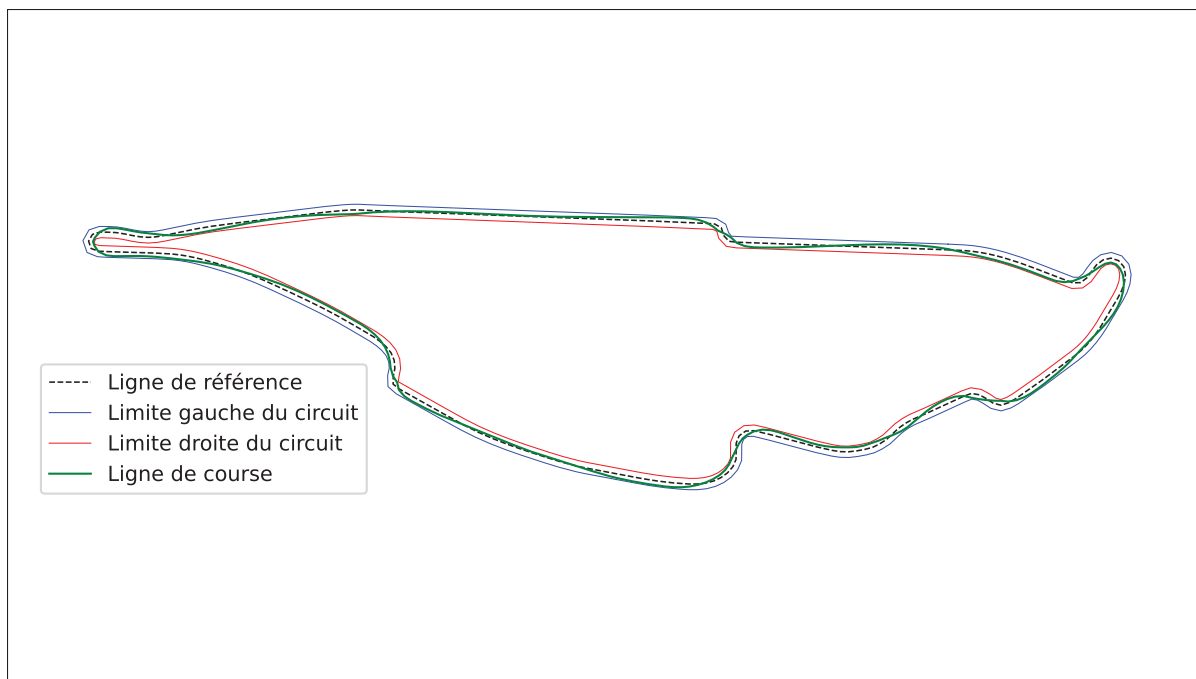


Figure 3.5 Trajectoire avec une itération sur le circuit de Montréal

Il peut être compliqué de discerner les différences entre les deux trajectoires de façon qualitative. Cependant, avec les figures 3.7 et 3.8 on peut voir que la trajectoire avec quatre itérations est plus lisse. En particulier, notons que la trajectoire à une itération présente une oscillation dans la transition entre le treizième et quatorzième virages (Figure 3.8).

La figure 3.9 montre l'intérêt de l'approche itérative : avec une seule itération les pics de courbure sont plus élevés par rapport à ceux de la ligne centrale qui est causée par l'approximation linéaire.

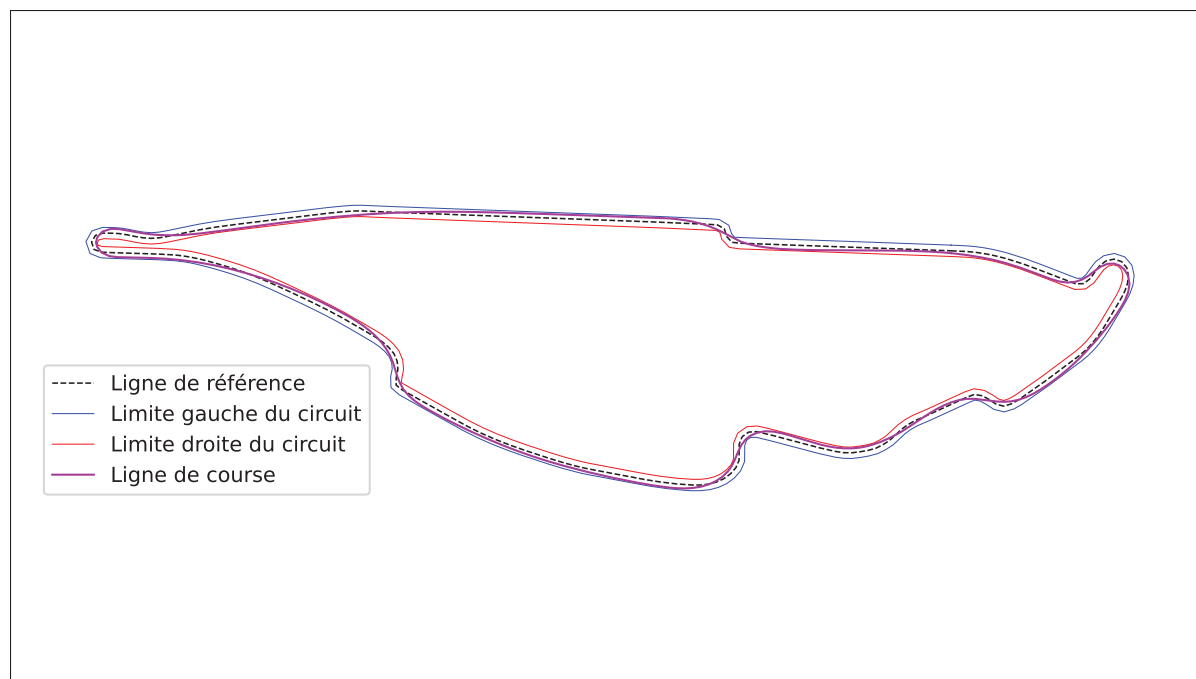


Figure 3.6 Trajectoire avec quatre itérations sur le circuit de Montréal

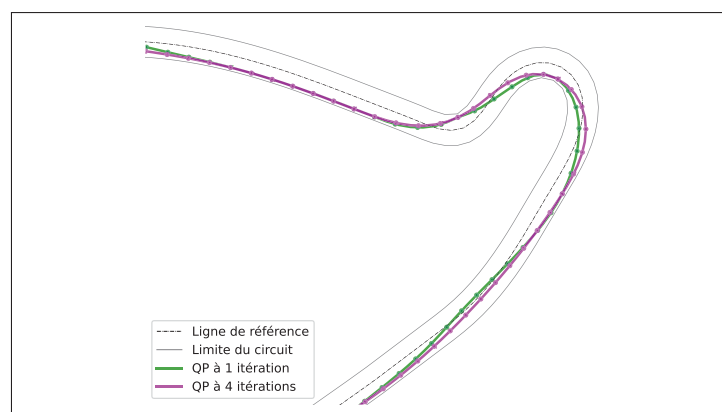


Figure 3.7 Superposition des trajectoires avec une itération et quatre itérations sur le premier virage du circuit de Montréal

Le tableau 3.1 montre quantitativement les améliorations apportées par la résolution répétée du QP. Le trajet est plus court que la ligne centrale et la courbure totale est réduite considérablement.

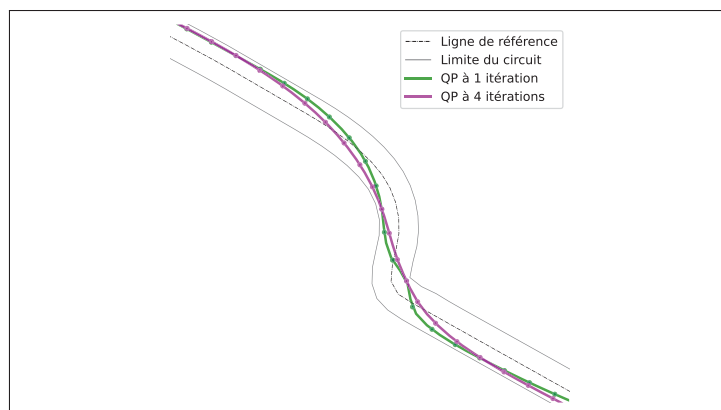


Figure 3.8 Superposition des trajectoires avec une itération et quatre itérations sur le treizième et quatorzième virages du circuit de Montréal

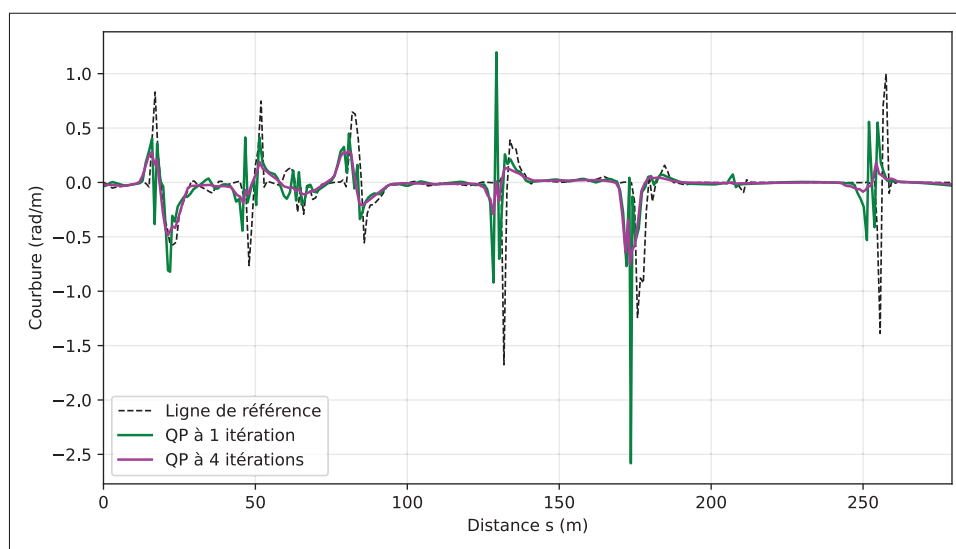


Figure 3.9 Profils de courbure des trajectoires calculées sur le circuit de Montréal

3.3.1.2 Profils de vitesse

Après avoir généré la trajectoire à courbure minimale, on calcule le profil de vitesse par intégration avant-arrière.

Les virages impliquent de devoir freiner avant d'entrer dans le virage puis de reprendre de la vitesse dans le virage comme le montre la figure 3.10.

Tableau 3.1 Statistiques sur les trajectoires générées sur le circuit de Montréal comparant la ligne centrale, la solution QP simple et la solution itérative IQP pour le circuit de Montréal

Trajectoire	L (m)	$\max \kappa $ (rad/m)	$\sum \kappa^2$	Itération	Temps de calcul (ms), $n = 100$
Ligne centrale	284.58	1.68	15.84	0	0
QP simple	280.90	2.58	16.31	1	194.60 ± 3.53
IQP	280.26	0.76	4.72	4	2558.26 ± 179.21

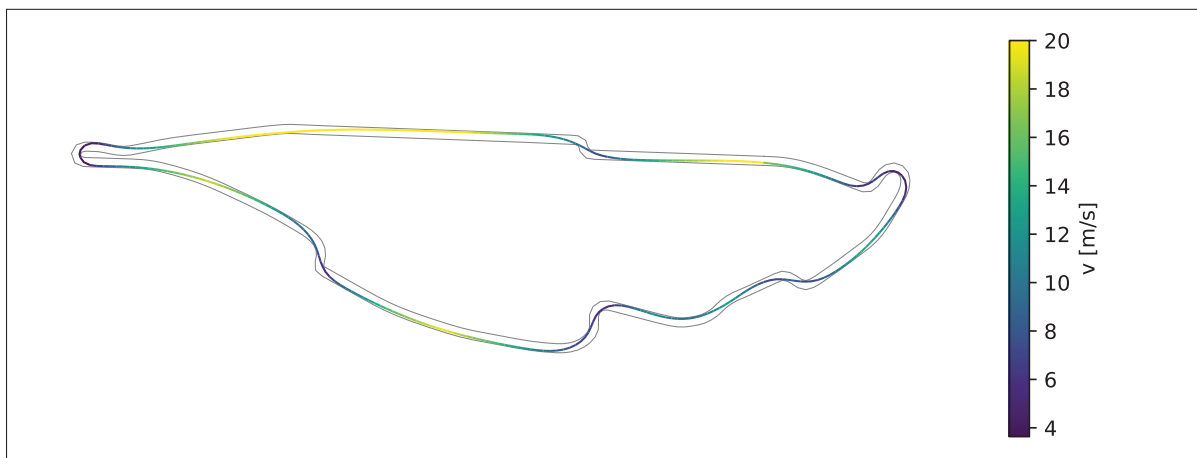


Figure 3.10 Profil de vitesse généré projeté sur la trajectoire IQP du circuit de Montréal

La figure 3.11 montre le profil de vitesse généré sur la trajectoire à courbure minimale et celui généré sur la ligne de référence (ligne centrale). Les zones dans lesquelles la courbure de la trajectoire de course est visiblement plus faible que celle de la ligne centrale correspondent aux virages (Figure 3.9). Dans ces zones, le profil de vitesse est plus élevé que sur la ligne centrale. Ceci permet non seulement d'augmenter la vitesse de passage dans les virages, mais aussi d'augmenter les vitesses atteintes sur les lignes droites et parfois même d'atteindre la vitesse maximale du véhicule.

Tableau 3.2 Statistiques sur le profil de vitesse généré sur la trajectoire à courbure minimale sur le circuit de Montréal

Trajectoire	$v_{\min}$ (m/s)	v_{moy} (m/s)	$v_{\max}$ (m/s)	Temps de calcul (ms), $n = 100$
Ligne centrale	1.34	9.17	18.92	1.01 ± 0.17
IQP (4 itérations)	1.99	10.52	20.00	0.94 ± 0.02

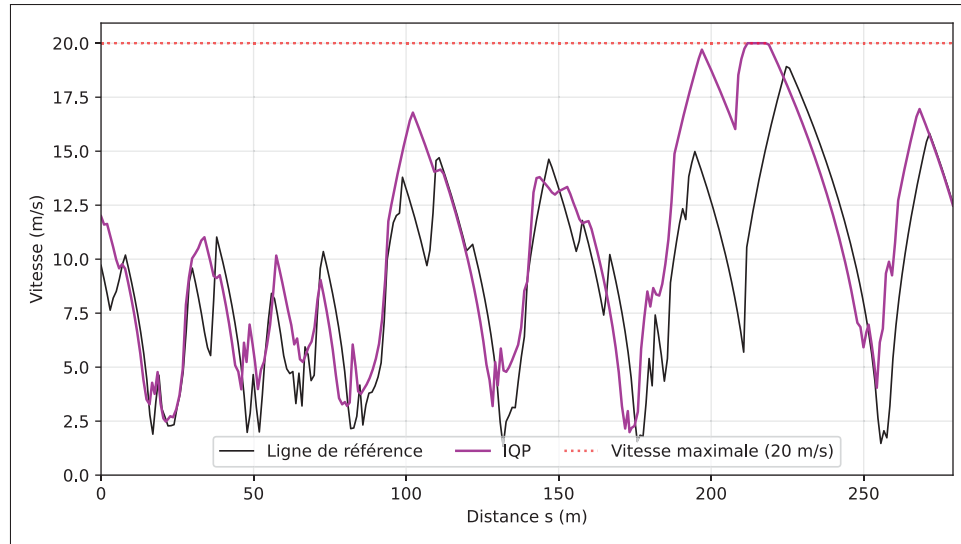


Figure 3.11 Profils de vitesse générés sur la trajectoire à courbure minimale et la ligne de référence (ligne centrale)

3.4 Conclusions

Ce chapitre a présenté un planificateur qui découple la trajectoire géométrique du profil de vitesse, en suivant la formulation à courbure minimale de Heilmeier *et al.* (2020). L'approche repose sur l'intuition qu'une trajectoire de plus faible courbure exige une accélération latérale moindre et autorise donc des vitesses plus élevées. Le tracé est d'abord optimisé en minimisant la somme des courbures au carré, puis un profil de vitesse lui est associé sous la contrainte du cercle de friction (section 3.2.5).

La résolution du problème repose sur deux choix de modélisation dont les résultats confirment la pertinence. D'une part, l'interpolation par splines cubiques assure une courbure continue le long du parcours (figures 3.7 et 3.8) et rend les dérivées secondes affines en α , ce qui permet de ramener l'objectif à une forme quadratique (section 3.2.3.4). D'autre part, l'approximation linéaire de la courbure introduit une erreur dans les sections les plus courbées, que l'affinement itératif corrige par résolutions successives du QP : sur le circuit de Montréal, la courbure maximale passe de 2.58 rad/m pour une itération à 0.76 rad/m après quatre itérations, et la courbure totale $\sum \kappa^2$ est divisée par plus de trois (tableau 3.1). Cette réduction de courbure se

traduit directement par un gain de vitesse, la vitesse moyenne progressant de 9.17 m/s sur la ligne centrale à 10.52 m/s sur la trajectoire optimisée (tableau 3.2). L'ensemble de la chaîne de calcul s'exécute en moins de trois secondes, ce qui, conjugué à la nature hors ligne du planificateur, permet de pré-calculer la trajectoire une fois pour toutes et de la stocker en vue d'un suivi en temps réel par un contrôleur, abordé au chapitre suivant.

Cette approche présente néanmoins des limitations. Le planificateur à courbure minimale suppose une représentation complète et exacte de la piste. Sa robustesse face à l'imprécision des données cartographiques n'a pas été étudiée. Par ailleurs, le cercle de friction qui borne les limites dynamiques s'appuie sur des caractéristiques physiques du véhicule supposées ici connues et fixes, alors qu'en pratique elles sont a priori inconnues et doivent être mesurées ou estimées. Enfin, le profil de vitesse repose sur un modèle quasi stationnaire et ne saurait se substituer à une validation en boucle fermée associant un contrôleur et un modèle dynamique complet du véhicule.

CHAPITRE 4

CONTRÔLE PAR COMMANDE PRÉDICTIVE

La commande prédictive est une stratégie d’asservissement qui optimise en ligne les actions de contrôle sur un horizon fini. La commande par *Model Predictive Control* (MPC) abordée dans ce chapitre s’appuie sur l’optimisation mathématique pour résoudre un programme contraint. Le MPC est un contrôleur prédictif, car il intègre la dynamique du système, les limites du circuit et les limites physiques sont explicitement considérées à chaque instant de commande. Cette capacité à anticiper le comportement futur du véhicule tout en respectant les contraintes fait du MPC une approche particulièrement adaptée à la course autonome, où le véhicule doit opérer proche de la limite de son enveloppe de performance.

Ce chapitre situe d’abord cette approche par rapport aux travaux connexes (section 4.1), en couvrant le MPC pour la course autonome (section 4.1.1), l’apprentissage en ligne (section 4.1.2) et la commande prédictive par échantillonnage (section 4.1.3). Il introduit ensuite les fondements théoriques (section 4.2) : la formulation générale du MPC (section 4.2.1), sa spécialisation à la course autonome (section 4.2.2), le modèle de prédiction (section 4.2.3) et les contraintes du problème d’optimisation (section 4.2.4). La mise en œuvre (section 4.3) détaille la discrétisation et l’horizon de prédiction choisis (section 4.3.1), le réglage des paramètres (section 4.3.2) et l’implémentation logicielle (section 4.3.3). Enfin, les résultats (section 4.4) analysent le suivi de trajectoire (section 4.4.1), le suivi de profil de vitesse (section 4.4.2) et la fluidité des actions (section 4.4.3).

4.1 Travaux connexes

La commande prédictive est devenue une approche de référence pour la course autonome, car elle permet d’optimiser les commandes tout en imposant explicitement les contraintes de piste, d’actionneurs et de dynamique du véhicule. On peut retrouver des travaux organisés autour de trois familles : les formulations déterministes de MPC et *Model Predictive Contouring Control* (MPCC) (Liniger, Domahidi & Morari, 2015; Evans *et al.*, 2024), les MPC fondés sur

l'apprentissage (*Learning-based MPC*, LMPC) (Kabzan, Hewing, Liniger & Zeilinger, 2019; Hewing, Wabersich, Menner & Zeilinger, 2020b), puis les approches par échantillonnage de type *Model Predictive Path Integral* (MPPI) (Williams, Goldfain, Drews, Rehg & Theodorou, 2017; Williams, Drews, Goldfain, Rehg & Theodorou, 2018).

4.1.1 MPC pour la course autonome

Les travaux de Liniger *et al.* (2015) constituent une référence importante pour la course autonome à petite échelle. Les auteurs s'attaquent à la course de voitures RC 1/43 comme un problème d'optimisation à horizon glissant, avec un objectif de maximisation de la progression sur la piste sous contraintes de limites de piste, d'actionneurs et d'évitement d'obstacles. Ils comparent une architecture hiérarchique planification-suivi de trajectoire à une formulation MPCC, où la génération de trajectoire et le suivi sont intégrés dans un même problème. Les problèmes non linéaires sont localement approximés par des programmes quadratiques résolus avec *Fast Optimization for Real-time Control on Embedded Systems* (FORCES) (Zanelli, Domahidi, Jerez & Morari, 2020), ce qui permet une exécution à 50 Hz sur une plateforme embarquée.

Verschueren, Zanon, Quirynen & Diehl (2016) abordent le même type de problème sous l'angle du *Nonlinear MPC* (NMPC) en formulant un programme non linéaire où on cherche à réduire directement le temps. La dynamique est reformulée dans le domaine spatial : l'abscisse curviligne (section 1.2.2) de la piste devient la variable indépendante et le temps devient une variable d'état à minimiser. Le problème non linéaire résultant est décomposé en une séquence de sous-problèmes quadratiques qui sont résolus avec *Automatic Control and Dynamic Optimization* (ACADO) (Houska, Ferreau & Diehl, 2011a,b).

Sur la plateforme BARC 1/10, Alcalá, Puig, Quevedo & Rosolia (2020) proposent une architecture à deux niveaux : une trajectoire optimisant le temps au tour est d'abord calculée hors ligne, puis suivie en ligne par un contrôleur *Linear Parameter-Varying MPC* (LPV-MPC). La dynamique non linéaire est approximée par un modèle linéaire à paramètres variables, ce qui conserve une structure convexe et permet une résolution avec *Operator Splitting Quadratic Program* (OSQP)

(Stellato, Banjac, Goulart, Bemporad & Boyd, 2020) à 30 Hz, avec un temps de calcul moyen de 14,9 ms. Les résultats sont validés à la fois en simulation et sur une plateforme physique.

Cataffo, Silano, Iannelli, Puig & Glielmo (2022) proposent un NMPC qui combine génération de trajectoire et suivi dans un seul problème d'optimisation. Le modèle de prédiction combine un modèle de bicyclette dynamique avec un modèle de pneu Pacejka (section 1.1.3.1) et un modèle d'un groupe motopropulseur. Les contraintes physiques de la piste sont formulées à partir de la projection sur la ligne centrale, et les obstacles statiques sont représentés par des zones d'exclusion circulaires. L'implémentation avec *Optimization Engine* (OpEn) (Sopasakis, Fresk & Patrinos, 2020) et *Proximal Averaged Newton-type method for Optimal Control* (PANOC) (Stella, Themelis, Sopasakis & Patrinos, 2017) atteint 30 Hz en simulation sur Gazebo, avec des temps de calcul moyens entre 0,9 et 1,4 ms sur un ordinateur portable.

Enfin, Tătulea-Codrean, Mariani & Engell (2020) étudient le coût de calcul d'un NMPC sur la plateforme F1TENTH en utilisant *Computer Algebra System with Automatic Differentiation* (CasADi) (Andersson, Gillis, Horn, Rawlings & Diehl, 2019) et *Interior Point OPTimizer* (IPOPT) (Wächter & Biegler, 2005) pour générer une politique optimale, puis en entraînant un réseau de neurones à reproduire cette politique. L'approche montre qu'un contrôleur appris peut approximer le comportement du NMPC et se transférer à un nouveau circuit simulé avec une erreur moyenne de commande de 7,3 %.

4.1.2 Apprentissage en ligne

Les formulations déterministes de MPC supposent que le modèle de prédiction décrit suffisamment bien le véhicule. En pratique, les erreurs de modélisation et les perturbations peuvent conduire à des violations des limites de la piste, en particulier proche des limites d'adhérence.

Les méthodes par apprentissage visent à améliorer le modèle de prédiction à partir de données collectées en ligne. Kabzan *et al.* (2019) augmentent un modèle de bicyclette dynamique par un processus gaussien qui apprend les erreurs résiduelles. Le processus gaussien corrige le modèle initial et fournit une estimation d'incertitude sur ses prédictions. Un critère de gain d'information

maintient un jeu de 300 points, et l'approche réduit les temps au tour d'environ 10 % tout en maintenant la voiture sur la piste.

Hewing, Kabzan & Zeilinger (2020a) proposent un MPC doté d'un processus gaussien et de contraintes probabilistes. L'incertitude du modèle est propagée sur l'horizon de prédiction, puis utilisée pour resserrer les contraintes sur les états et les commandes. La méthode est validée en simulation et sur voitures miniatures, montrant une amélioration simultanée de la performance et de la sécurité par rapport à un contrôleur déterministe fixe.

4.1.3 Commande prédictive par échantillonnage

La formulation par MPPI constitue une alternative aux formulations MPC résolues par optimisation mathématique. Williams, Drews, Goldfain, Rehg & Theodorou (2016) introduisent un MPPI fondé sur les notions physiques d'énergie libre et d'entropie relative. À chaque pas de temps, plusieurs trajectoires sont échantillonnées en parallèle sur une carte graphique, puis ces trajectoires sont agrégées selon une moyenne pondérée par le coût des perturbations échantillonnées pour former une nouvelle séquence de commandes. L'approche ne requiert pas de gradient du coût ou de la dynamique du système, ce qui facilite l'utilisation de coûts présentant des discontinuités. Sur la plateforme AutoRally 1/5, le contrôleur réalise une conduite agressive avec une vitesse cible de 8 m/s, incluant des glissades contrôlées proches, voire au-delà de la limite de friction.

L'implémentation retenue est une formulation NMPC déterministe résolue avec CasADi (Andersson *et al.*, 2019) et IPOPT (Wächter & Biegler, 2005) sur F1TENTH Gym. Ce choix privilégie la flexibilité de prototypage : le modèle dynamique et les contraintes peuvent être modifiés directement dans une formulation symbolique, sans générer un solveur spécialisé. Il se distingue des approches embarquées à très faible latence comme acados (Verschuere *et al.*, 2022), FORCES (Zanelli *et al.*, 2020) ou OpEn (Sopasakis *et al.*, 2020), mais convient au cadre expérimental simulé utilisé ici.

4.2 Fondements théoriques

4.2.1 Formulation générale du MPC

À chaque pas de temps t , le MPC résout un problème de commande optimale sur un horizon de prédiction de N pas. Compte tenu de l'état actuel $\mathbf{x}_t$, le contrôleur minimise une fonction de coût sur une séquence de commandes futures $\mathbf{u}_t, \dots, \mathbf{u}_{t+N-1}$:

$$\min_{\mathbf{u}_{t:t+N-1}} \sum_{k=0}^{N-1} \ell(\mathbf{x}_{t+k}, \mathbf{u}_{t+k}) + V_f(\mathbf{x}_{t+N})$$

sous les contraintes :

$$\mathbf{x}_{t+k+1} = f(\mathbf{x}_{t+k}, \mathbf{u}_{t+k}), \quad k = 0, \dots, N-1 \quad (4.1)$$

$$\mathbf{x}_{t+k} \in \mathcal{X}$$

$$\mathbf{u}_{t+k} \in \mathcal{U}$$

où f est le modèle de prédiction, ℓ est le coût instantané, V_f est le coût terminal, $\mathcal{X}$ est l'ensemble des états possibles et $\mathcal{U}$ est l'ensemble des commandes possibles. Seule la première commande optimale $\mathbf{u}_t^*$ est appliquée au système. L'horizon se déplace ensuite d'un pas et le problème est résolu à nouveau : c'est le principe de l'horizon glissant (Borrelli, Bemporad & Morari, 2017; Rawlings, Mayne & Diehl, 2020).

Les principaux avantages du MPC par rapport aux contrôleurs classiques sont la faculté de gérer simultanément les contraintes sur les états et les commandes sur un horizon futur, et l'utilisation d'un modèle prédictif pour anticiper le comportement futur. La principale limite est le coût de calcul : résoudre un problème d'optimisation en temps réel requiert un équilibre entre longueur de l'horizon, complexité du modèle et capacité de calcul disponible.

4.2.2 NMPC pour la course autonome

La course autonome sollicite une dynamique du véhicule non linéaire ce qui motive une formulation NMPC. Le modèle de prédiction exploite le modèle dynamique de bicyclette décrit

à la section 1.1.2, exprimé dans le repère curviligne décrit à la section 1.2.2 afin de prendre en compte directement la progression et les limites de piste. La commande optimisée est $\mathbf{u} = [\delta, a]^\top$, où δ désigne l'angle de braquage et a l'accélération longitudinale.

La fonction de coût est conçue pour maximiser la progression le long de la ligne de référence tout en pénalisant l'écart à l'état de référence et les variations de commandes :

$$\ell(\mathbf{x}, \mathbf{u}) = w_e e_{\text{lat}}^2 + w_v e_v^2 + w_\delta \dot{\delta}^2 + w_a \dot{a}^2 \quad (4.2)$$

où :

- e_{lat} est l'erreur latérale par rapport à la ligne centrale,
- e_v est l'écart de vitesse par rapport à la vitesse de référence,
- $\dot{\delta}$ est le taux de variation de l'angle de braquage,
- $\dot{a}$ est le taux de variation de l'accélération longitudinale.

Les deux derniers termes permettent de pénaliser les variations de commandes qui peuvent entraîner un comportement saccadé. Les poids w_e , w_v , w_δ , w_a permettent d'ajuster le compromis entre vitesse, précision du suivi et stabilité des commandes.

4.2.3 Modèle de prédiction

Le choix du modèle de prédiction est déterminant pour la qualité du NMPC. Le modèle cinématique de bicyclette (équations 1.1 à 1.4) est simple et ne requiert pas l'identification des paramètres de pneu, mais il ne capture pas les effets de glissement latéral qui deviennent significatifs à haute vitesse et dans les virages serrés.

Le modèle dynamique de bicyclette (équations 1.5 à 1.7) explicite l'influence des forces longitudinales et latérales agissant sur chaque essieu. Couplé à un modèle de pneu non linéaire, comme la formule magique de Pacejka (équation 1.8), il peut capturer la limite d'adhérence et la saturation des forces latérales.

L'implémentation retenue utilise le modèle dynamique de bicyclette à *pneus linéaires*, dans lequel les forces latérales sont proportionnelles aux angles de glissement par les rigidités de dérive C_{Sf} et C_{Sr} des essieux avant et arrière. Ce modèle de prédiction est apparié au modèle dynamique du véhicule du simulateur F1TENTH Gym, retenu au chapitre 2 : le contrôleur et le simulateur partageant la même dynamique, les prédictions tiennent compte du glissement des pneus et le contrôleur demeure stable proche de la limite d'adhérence. Le couplage à un modèle de pneu non linéaire de type Pacejka (équation 1.8) constitue une option qui capturerait la saturation des forces latérales, mais il n'a pas été retenu ici : il exigerait l'identification de paramètres supplémentaires.

L'expression curviligne de la dynamique dans le repère de Frenet (arc s , erreur latérale e_{lat} décrite à la section 1.2.2) est particulièrement commode pour formuler les contraintes du circuit : les limites de la chaussée deviennent de simples bornes sur e_{lat} , indépendantes de la géométrie globale du circuit.

4.2.4 Contraintes du problème d'optimisation

Les contraintes imposées au problème d'optimisation reflètent les limites physiques du véhicule en termes d'état et de commande.

4.2.4.1 Contraintes liées au circuit

L'erreur latérale doit être bornée par les largeurs de piste du circuit :

$$\begin{aligned} |e_{lat,i}| &\leq w_{i,droite} - d_{marge} \\ |e_{lat,i}| &\leq w_{i,gauche} - d_{marge} \end{aligned} \tag{4.3}$$

où $d_{marge} > 0$ est une marge de sécurité tenant compte des dimensions du véhicule.

4.2.4.2 Contraintes sur les commandes

L'angle de braquage est borné par les limites mécaniques de la direction et l'accélération longitudinale doit rester dans une plage physiquement possible :

$$\begin{aligned} |\delta_k| &\leq \delta_{\max} \\ a_{\min} &\leq a_k \leq a_{\max} \end{aligned} \tag{4.4}$$

4.2.4.3 Contraintes sur les variations des commandes

Pour limiter les oscillations, les variations de commande d'un pas de temps à l'autre sont également bornées :

$$\begin{aligned} |\delta_{k+1} - \delta_k| &\leq \Delta\delta_{\max} \\ |a_{k+1} - a_k| &\leq \Delta a_{\max} \end{aligned} \tag{4.5}$$

4.3 Mise en œuvre

Le contrôleur est exécuté en boucle fermée sur le simulateur F1TENTH Gym selon le principe de l'horizon glissant : à chaque pas de commande, l'état du véhicule dans le simulateur, le programme non linéaire est résolu, et seule la première commande optimale est appliquée. L'horizon se déplace ensuite d'un pas et le problème est résolu à nouveau, à une fréquence de 25 Hz.

4.3.1 Discrétisation et horizon de prédiction

La dynamique continue du véhicule est discrétisée avec une intégration de Runge-Kutta d'ordre 4 (Hairer & Wanner, 2015) à pas de temps fixe T_s . L'intégration RK4 offre un bon compromis entre précision de l'approximation et complexité numérique.

L'horizon de prédiction N et le pas de temps T_s définissent conjointement la fenêtre temporelle $T = N \cdot T_s$ sur laquelle le contrôleur anticipe le comportement du véhicule. L'implémentation retenue fixe $T_s = 0,04$ s, soit une fréquence de commande de 25 Hz, et $N = 40$ pas, ce qui correspond à une fenêtre d'anticipation de $T = N \cdot T_s = 1,6$ s. Un horizon plus long améliore la planification en virage. Le contrôleur «voit» le virage à l'avance et peut prévoir son point de freinage. En contrepartie, un nombre accru de variables d'optimisation implique un temps de calcul plus élevé. Le compromis entre anticipation et faisabilité temps-réel est un paramètre clé de la mise en œuvre du MPC.

4.3.2 Réglage des paramètres

4.3.2.1 Poids de la fonction de coût

Les poids w_e , w_v , w_δ et w_a de l'équation (4.2) contrôlent le compromis entre précision de suivi ($w_e e_{\text{lat}}^2$, $w_v e_v^2$) et régularité des commandes ($w_\delta \delta^2$, $w_a \dot{a}^2$). Les valeurs retenues sont consignées dans le tableau 4.1.

Tableau 4.1 Poids de la fonction de coût

Poids	Valeur
w_e	50
w_v	30
w_δ	1
w_a	0.2

Les poids ont été obtenus par essais et erreurs.

4.3.2.2 Marges et bornes

La marge de sécurité d_{marge} est fixée à la demi-largeur du véhicule ($d_{\text{marge}} = W_{\text{veh}}/2 = 0.155$ m). La limite de braquage est de $\delta_{\text{max}} = 3.2$ rad et les bornes d'accélération sont de $a_{\text{max}} = 9.51$ m/s² et $a_{\text{min}} = -a_{\text{max}}$. Ces valeurs sont obtenues à partir des paramètres du simulateur F1TENTH Gym.

4.3.2.3 Initialisation à chaud

Pour réduire le temps de calcul à chaque pas, la séquence de commandes optimale du pas précédent, décalée d'un pas, sert d'initialisation pour le problème en cours. Cette stratégie exploite la continuité temporelle de la solution et réduit le nombre d'itérations du solveur nécessaires à la convergence.

4.3.3 Implémentation logicielle

La formulation et la résolution du programme non linéaire reposent sur CasADi (Andersson *et al.*, 2019), une librairie de différentiation automatique symbolique et de formulation de problèmes d'optimisation, couplée à IPOPT (Wächter & Biegler, 2005), un solveur de point intérieur pour les programmes non linéaires. Cette suite d'outils est retenue pour sa flexibilité lors du prototypage : la formulation symbolique facilite les modifications du modèle de prédiction et de la fonction de coût sans recoder le solveur. Elle se distingue des outils spécialisés pour le déploiement embarqué à très faible latence, comme acados (Verschuere *et al.*, 2022), FORCES (Zanelli *et al.*, 2020) ou OpEn (Sopasakis *et al.*, 2020), mais convient au cadre expérimental simulé où la contrainte temps réel est moins stricte que sur matériel embarqué.

Le modèle de prédiction et l'intégrateur RK4 (section 4.3.1) sont construits une seule fois sous la forme d'une fonction CasADi symbolique. Le calcul symbolique permet de réutiliser cette fonction pour chaque étape de l'horizon de prédiction sans reconstruire la fonction à chaque fois. De la même façon, la fonction de coût et les contraintes sont assemblées une seule fois de façon symbolique : les variables de décision (états, commandes et variables d'écart, voir ci-dessous) ainsi qu'un unique vecteur de paramètres regroupant l'état courant $\mathbf{x}_t$, la dernière commande appliquée et la référence sur l'horizon sont déclarés symboliquement, et le programme non linéaire résultant est construit une seule fois avec IPOPT. La démarche de commande en boucle fermée est résumée par deux algorithmes : l'algorithme 4.1 construit le contrôleur une seule fois hors ligne.

Algorithme 4.1 Construction du contrôleur et commande en boucle fermée

Input : Modèle de prédiction f et intégrateur RK4 F_{RK4}
Input : Horizon de prédiction N et pas de temps T_s
Input : Poids de la fonction de coût w_e, w_v, w_δ, w_a
Input : Contraintes d'état $\mathcal{X}$ et de commande $\mathcal{U}$
Input : Nombre d'itérations totales N_{total}

- 1 $F_{\text{RK4}} \leftarrow$ intégrateur RK4 construit symboliquement à partir du modèle de prédiction f ;
- 2 $J, \mathcal{X}, \mathcal{U} \leftarrow$ fonction de coût (4.2) et contraintes 4.3–4.5 assemblées symboliquement à partir de F_{RK4} ;
- 3 solveur($\cdot$) $\leftarrow$ programme non linéaire construit avec IPOPT à partir de $J, \mathcal{X}$ et $\mathcal{U}$;
- 4 $\mathbf{u}_{-1} \leftarrow$ commande initiale;
- 5 solution_precedente $\leftarrow \emptyset$;
- 6 **for** $t = 0, \dots, N_{\text{total}} - 1$ **do**
- 7 $\mathbf{u}_t^*, \mathbf{u}_{t-1}, \text{solution_precedente} \leftarrow$
 algorithme 4.2(solveur($\cdot$), $\mathbf{u}_{t-1}, \text{solution_precedente}, N$);
- 8 Appliquer la première commande optimale $\mathbf{u}_t^*$ au véhicule;
- 9 **end for**

À chaque instant t le solveur est appelé pour obtenir la commande optimale $\mathbf{u}_t^*$ pour l'état $\mathbf{x}_t$ et la référence ref_t . Chaque résolution en boucle fermée ne fait alors que mettre à jour numériquement ce vecteur de paramètres, sans reconstruire le solveur. L'algorithme 4.2 décrit l'utilisation en ligne du solveur.

4.4 Résultats

La ligne de course et le profil de vitesse calculés hors ligne au chapitre 3 sont réutilisés comme référence à suivre par le MPC sur le simulateur F1TENTH Gym. Dans cette section sont présentés les résultats du MPC qui accomplit un temps au tour de 35,2 s sur le circuit réadapté de Montréal.

Algorithme 4.2 Obtention d'une commande optimale à l'instant t

```

Input : Solveur solveur(·)
Input : Commande appliquée au pas précédent  $\mathbf{u}_{t-1}$ 
Input : solution_precedente (ou  $\emptyset$  au premier pas)
Input : Horizon de prédiction  $N$ 
Output : Commande appliquée  $\mathbf{u}_t^*$ 
Output :  $\mathbf{u}_{t-1} \leftarrow \mathbf{u}_t^*$ , à réutiliser au pas suivant
Output : solution_precedente  $\leftarrow \mathbf{w}^*$ , à réutiliser au pas suivant

1  $\mathbf{x}_t \leftarrow$  état du véhicule;
2  $\text{ref}_t \leftarrow$  état de référence sur l'horizon  $[t, t + N]$ ;
3  $\mathbf{p}_t \leftarrow (\mathbf{x}_t, \mathbf{u}_{t-1}, \text{ref}_t)$ ;
4 if solution_precedente  $\neq \emptyset$  then
5   |  $\mathbf{w}_0 \leftarrow$  solution_precedente décalée d'un pas, dernière étape dupliquée;
6 else
7   |  $\mathbf{w}_0 \leftarrow$  état courant  $\mathbf{x}_t$  et dernière commande  $\mathbf{u}_{-1}$  répétés sur l'horizon;
8 end if
9  $\mathbf{w}^* \leftarrow \text{solveur}(\mathbf{p}_t, \mathbf{w}_0)$ ; // solution optimale du programme non linéaire
10  $\mathbf{x}_t^*, \dots, \mathbf{x}_{t+N}^*, \mathbf{u}_t^*, \dots, \mathbf{u}_{t+N-1}^* \leftarrow \mathbf{w}^*$ ;
11 solution_precedente  $\leftarrow \mathbf{w}^*$ ;
12  $\mathbf{u}_{t-1} \leftarrow \mathbf{u}_t^*$ ;

```

4.4.1 Suivi de trajectoire

La figure 4.1 montre que globalement la trajectoire suivie par le MPC reste proche de la ligne de course de référence.

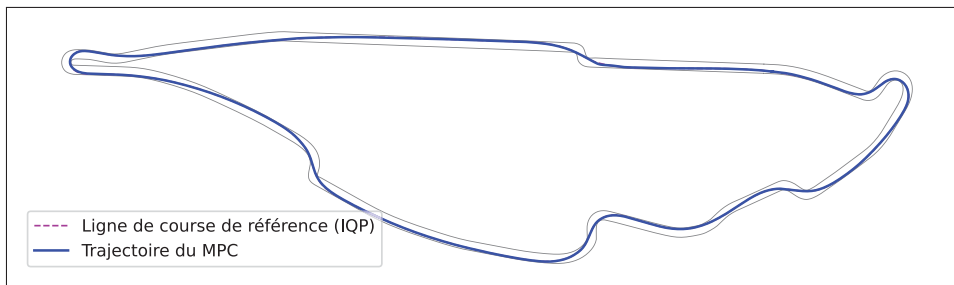


Figure 4.1 Trajectoire de référence et trajectoire suivie par le MPC sur le circuit de Montréal

Le suivi est particulièrement précis dans les portions à faible courbure, où la voiture reste quasiment confondue avec la trajectoire de référence. Le profil d'erreur latérale de la figure 4.2 confirme ce comportement.

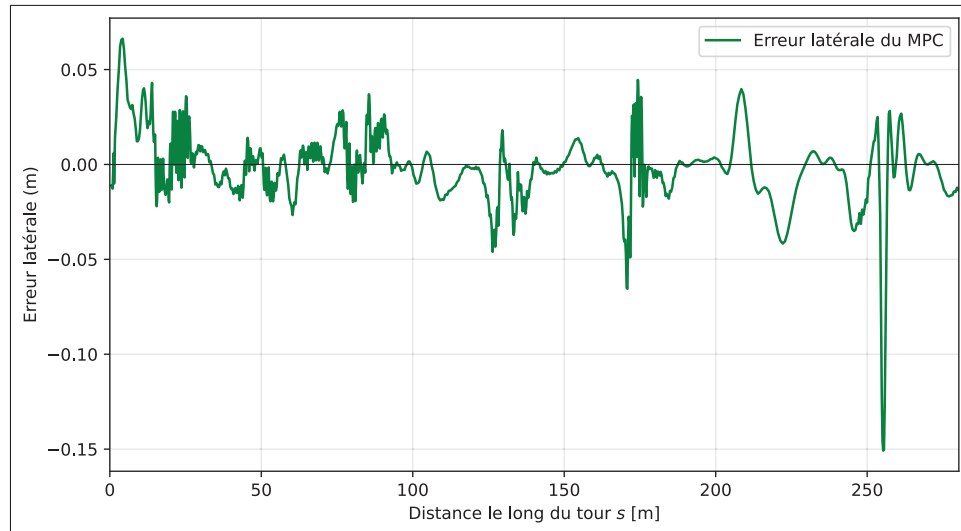


Figure 4.2 Erreur latérale par rapport à la trajectoire de référence du MPC

Les valeurs affichées dans la figure 4.2 montrent que l'erreur latérale absolue maximale mesurée est de 0,151 m, pour une erreur absolue moyenne de 0,001 m et une erreur quadratique moyenne de 0,020 m. Ce qui montre que le contrôleur maintient le véhicule sur le circuit tout en suivant la trajectoire de référence.

La figure 4.3 complète cette lecture en montrant que les écarts les plus importants se situent au niveau du point de corde des virages. La courbe de la trajectoire du MPC suit la courbe de référence sauf aux pics de déviation latérale. Ce qui peut s'expliquer par le fait que les zones au voisinage du point de corde des virages sont les plus proches des limites du circuit. Mais en dehors de ces zones le suivi est beaucoup plus précis.

4.4.2 Suivi du profil de vitesse

La figure 4.4 compare la vitesse réalisée en boucle fermée au profil de vitesse de référence issu de l'optimisation hors ligne. Globalement, le MPC reproduit en tendance le profil cible

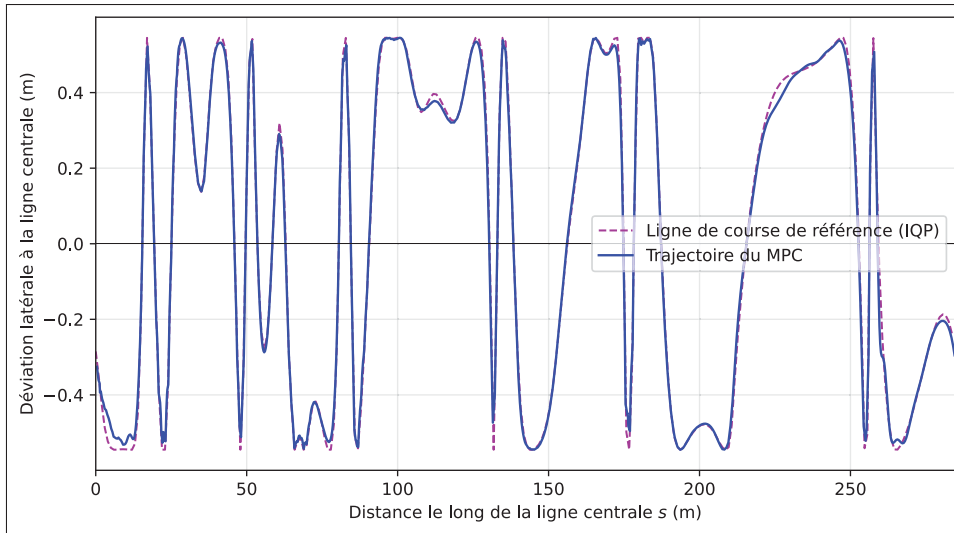


Figure 4.3 Déviation latérale par rapport à la ligne centrale de la trajectoire de référence et de la trajectoire suivie par le MPC

mais pas en termes d'amplitude dans les sections de haute vitesse. Les phases de réduction de vitesse avant les virages sont correctement anticipées. Cependant, dans les phases d'accélération donnant lieu à des accélérations longitudinales importantes, le MPC reste conservateur. Ce qui est cohérent avec une stratégie visant à préserver la stabilité latérale et le respect des contraintes de piste.

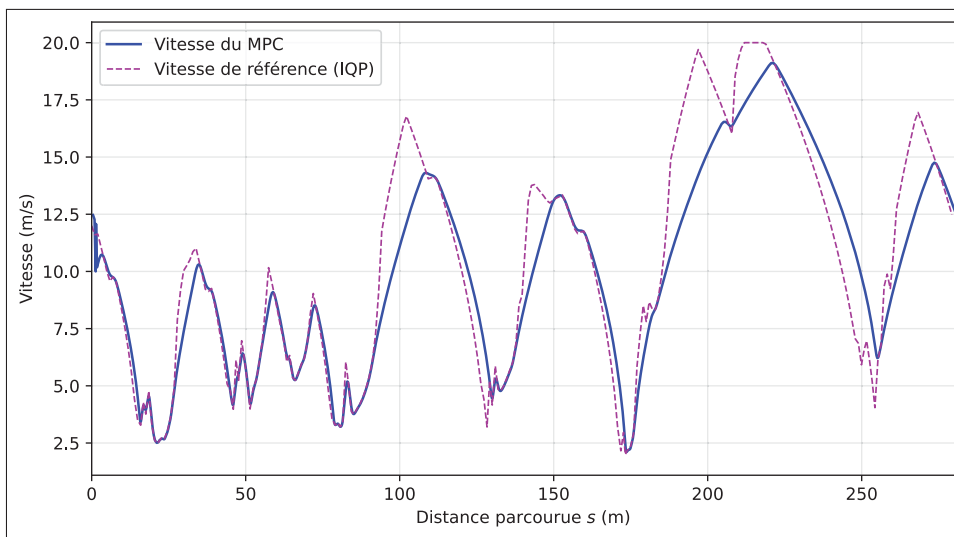


Figure 4.4 Vitesse suivie par le MPC et profil de vitesse de référence

Sur le tour réalisé, la vitesse moyenne mesurée est de 7,92 m/s, contre 10,52 m/s pour le profil de référence. L'écart moyen entre la vitesse de référence et la vitesse effectivement réalisée est donc de 2,60 m/s. Le MPC atteint rarement les vitesses de pointe du profil de référence. Le maximum atteint est de 19,11 m/s contre 20,0 m/s pour le profil de référence. Ces résultats montrent que l'expressivité du modèle de prédiction ne permet pas de reproduire exactement le profil de vitesse de référence.

4.4.3 Fluidité des actions

Les commandes générées par le MPC, présentées à la figure 4.5, restent globalement régulières au cours du tour.

La trace de l'accélération longitudinale est sans oscillations excessives. Pour l'accélération longitudinale, la valeur absolue maximale est de 6,71 m/s², avec une variation maximale pas à pas de 4,07 m/s². Ce comportement est cohérent avec l'introduction explicite de pénalités sur l'effort de commande et sur ses variations d'un pas à l'autre dans la fonction de coût.

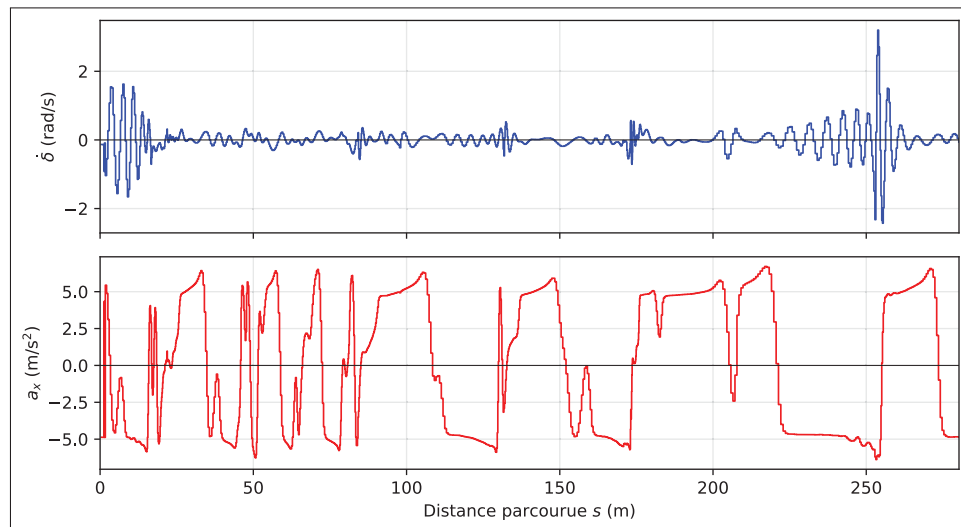


Figure 4.5 Commandes de direction et d'accélération du MPC

Cependant, la commande de direction présente des oscillations qui sont caractéristiques des effets couplés entre la dynamique latérale et la dynamique longitudinale d'un véhicule. Ce qui

conduit à un comportement erratique dans certaines portions du circuit. Quantitativement, la valeur absolue maximale du taux de braquage est de 3,20 rad/s, avec une variation maximale pas à pas de 3,60 rad/s.

L'effet de couplage s'explique par le fait que lorsque la voiture a une vitesse longitudinale élevée et qu'elle se trouve déplacée à gauche de la trajectoire de référence, la correction par la direction risque de l'emmener à droite de la référence. Puis l'effet inverse se reproduit créant alors un mouvement oscillatoire. Ce phénomène se retrouve amplifié lorsqu'on ajoute les exigences du suivi du profil de vitesse.

Les pénalités de type $w_\delta \dot{\delta}^2$ et $w_a \dot{a}^2$ jouent leur rôle pour favoriser des commandes fluides, ce qui contribue à la stabilité du suivi en boucle fermée. Mais il est possible que ces pénalités soient trop fortes et que le MPC soit trop conservateur, ce qui pourrait être corrigé en diminuant les poids des pénalités au risque de dégrader la stabilité du suivi.

4.5 Conclusions

Le contrôleur MPC étudié dans ce chapitre montre qu'une stratégie de suivi de trajectoire (géométrique et de vitesse longitudinale) s'appuyant sur un modèle dynamique de bicyclette permet de terminer un tour complet en maintenant le véhicule sur la piste, avec une erreur latérale et un profil de vitesse globalement cohérent avec la référence hors ligne, bien que conservateur. Dans ce cadre expérimental, le MPC constitue donc une approche structurée et interprétable pour exploiter explicitement les contraintes de piste et les limites du véhicule.

Les résultats mettent toutefois en évidence les limites de cette mise en œuvre. D'une part, les écarts observés dans les zones de forte courbure et le caractère conservateur du suivi longitudinal suggèrent que le modèle de prédiction retenu, basé sur des pneus linéaires, ne permet pas d'opérer correctement dans des situations de glissement latéral important.

Comme pour le planificateur, l'efficacité du MPC dépend fortement de la qualité du modèle du véhicule et de l'identification de ses paramètres. En pratique, des erreurs de modélisation

peuvent dégrader la précision de suivi ou conduire à des violations de contraintes. De plus, la nature non linéaire du problème interdit de garantir, à chaque itération, une convergence vers une solution acceptable. C'est pourquoi un système de secours, par exemple un contrôleur réactif, reste souhaitable pour assurer un comportement sûr en cas d'échec du solveur ou de dégradation trop importante du suivi.

Ces limites ouvrent plusieurs pistes d'amélioration. Un modèle dynamique de véhicule plus expressif, surtout au niveau du pneu, pourrait améliorer simultanément la fidélité du suivi. L'utilisation de solveurs tels que acados ou FORCES ouvre la voie à un déploiement en temps réel. Le chapitre 5 à la section 5.4.6 met cette approche en perspective avec une politique de contrôle apprise par un agent RL, afin de comparer plus directement le compromis entre qualité de suivi, coût de calcul et robustesse en boucle fermée.

CHAPITRE 5

CONTRÔLE PAR APPRENTISSAGE PROFOND PAR RENFORCEMENT

L'apprentissage profond par renforcement (*Deep Reinforcement Learning*) est une méthode d'apprentissage automatique couplant à la fois l'apprentissage par renforcement (*Reinforcement Learning*, RL) et l'apprentissage profond (*Deep Learning*, DL). Un agent apprend une politique par interaction avec son environnement dans un processus itératif. Contrairement au contrôleur MPC du chapitre 4, qui s'appuie sur un modèle explicite de la dynamique du véhicule et résout un problème d'optimisation à chaque pas de commande, l'agent découvre une stratégie de pilotage en maximisant une récompense cumulée au fil de nombreux essais en simulation. Cette capacité à apprendre un comportement performant sans s'appuyer sur l'identification des paramètres du véhicule ou d'une trajectoire de référence fait de l'apprentissage profond par renforcement une approche complémentaire pour la course autonome, où la dynamique proche de la limite d'adhérence est difficile à modéliser précisément.

Ce chapitre présente l'approche d'apprentissage profond par renforcement. Il situe d'abord cette approche par rapport aux travaux connexes (section 5.1), introduit les fondements théoriques nécessaires (section 5.2), décrit la mise en œuvre expérimentale (section 5.3), et présente les résultats d'entraînement (section 5.4).

5.1 Travaux connexes

5.1.1 Premières études en simulation

Les premières applications de l'apprentissage profond par renforcement à la course s'appuient sur des simulateurs tels que *The Open Racing Car Simulator* (TORCS) (Espié *et al.*, 2005). Perot, Jaritz, Toromanoff & De Charette (2017) ont démontré une approche de bout en bout dans un jeu de course réaliste en apprenant directement à partir de pixels bruts, établissant qu'une politique s'appuyant sur une architecture de réseau de neurones convolutifs entraînée uniquement par RL peut compléter des circuits. Remonda, Krebs, Veas, Luzhnica & Kern (2019) ont étendu cette

approche en utilisant des données de télémétrie plutôt que des images brutes, comparant plusieurs algorithmes dans TORCS et analysant les comportements de conduite appris par rapport à des pilotes professionnels. Guckiran & Bolat (2019) ont évalué l'algorithme *Soft Actor-Critic* (SAC) pour le contrôle dans TORCS, démontrant que les méthodes *off-policy* avec régularisation d'entropie ont des courbes d'apprentissage plus stables que les alternatives *on-policy*. L'approche retenue s'inscrit dans cette lignée mais l'applique à l'environnement FITENTH. Dans une méthode *off-policy* les données collectées sur l'environnement sont échantillonnées avec une politique d'exploration qui est différente de la politique de contrôle. Alors qu'avec une méthode *on-policy*, il n'y a pas de distinction entre la politique d'exploration et la politique de contrôle.

5.1.2 Performance en simulation face à des pilotes humains

Une avancée marquante est venue de l'application de l'apprentissage par renforcement à des simulateurs de course haute fidélité. Fuchs, Song, Kaufmann, Scaramuzza & Duerr (2021) ont entraîné un agent basé sur l'apprentissage par renforcement (agent RL) dans Gran Turismo Sport à l'aide d'un système de récompense de progression. L'agent RL surpasse l'intelligence artificielle intégrée au jeu et les performances humaines tirées d'une base de données de plus de 50 000 joueurs humains. C'est la première démonstration dans un simulateur photoréaliste. Wurman *et al.* (2022) ont poussé ce résultat plus loin avec Gran Turismo Sophy, un système basé sur SAC qui bat des pilotes humains champions. L'agent RL Sophy combine une fonction de récompense soigneusement conçue, un entraînement distribué à grande échelle et des techniques de régularisation. Ces résultats montrent que l'apprentissage par renforcement peut maîtriser le contrôle du véhicule à la limite absolue de performance même lorsqu'il est couplé à un simulateur fidèle. Ces travaux servent de référence sur le potentiel de SAC pour la course autonome.

5.1.3 Plateformes miniatures et transfert simulation-réalité

De l'échelle 1/10 à 1/18, le défi central est le transfert de simulation à la réalité. Balaji *et al.* (2020) ont introduit DeepRacer, une plateforme à l'échelle 1/18 couplée à un pipeline d'entraînement RL distribué, et ont démontré que des politiques entraînées en simulation peuvent être transférées

sur un véhicule physique grâce à la randomisation de domaine. Chisari, Liniger, Rupenyan, Gool & Lygeros (2021) ont présenté un transfert simulation-réalité sur une plateforme miniature en entraînant une politique purement en simulation avec randomisation du modèle de véhicule. Leur approche introduit une régularisation des actions de la politique et un espace d'action élargi pour garantir des actions fluides. Un entraînement de trois heures avec des données réelles permet d'atteindre des temps au tour comparables à un MPC tout en réduisant les violations de contraintes de moitié.

5.1.4 Apprentissage hors ligne et généralisation

Koirala & Fleming (2024) ont étudié les méthodes d'apprentissage par renforcement hors ligne pour F1TENTH. Ils reconnaissent que l'entraînement en ligne sur des circuits complexes échoue fréquemment. En utilisant des données de démonstration d'un contrôleur externe, ils entraînent différentes architectures de réseaux de neurones comme les *Transformers* ou les politiques de diffusion. Leur analyse met en évidence les défis de généralisation à de nouveaux circuits.

5.2 Fondements théoriques

Cette section présente les principes fondamentaux nécessaires à ce chapitre. Elle couvre les réseaux de neurones profonds, qui servent à approximer les politiques et les fonctions de valeur, le formalisme de l'apprentissage par renforcement et l'algorithme *Soft Actor-Critic*.

5.2.1 Réseaux de neurones

Les réseaux de neurones artificiels utilisés dans l'apprentissage profond (*Deep Learning*) s'inspirent des systèmes neuronaux biologiques : tout comme les neurones du cerveau reçoivent, traitent et transmettent des signaux, un neurone artificiel calcule un signal de sortie en fonction des signaux d'entrée. Ces neurones sont à la fois mis en parallèle dans une couche et en série dans un réseau. Le comportement collectif de nombreuses unités de ce type, organisées en couches, permet à un réseau d'apprendre des correspondances entrée-sortie complexes directement à

partir de données (Goodfellow, Bengio & Courville, 2016).

$$y = f\left(\sum_{i=1}^n w_i x_i + b\right) \quad (5.1)$$

Au niveau d'un neurone unique, l'unité forme une somme pondérée de ses entrées x_i , ajoute un biais b et applique une fonction d'activation f pour produire sa sortie. Le poids w_i attribué à chaque entrée détermine à quel point cette entrée influence la réponse du neurone. Le biais déplace le seuil d'activation du neurone. Les fonctions d'activation sont cruciales pour introduire la non-linéarité dans le réseau. Sans elles, toute composition de transformations linéaires s'effondre en une seule transformation affine, quelle que soit la profondeur, ce qui limite sévèrement la classe de fonctions représentables. Diverses fonctions d'activation sont utilisées en pratique.

La fonction sigmoïde $\sigma(z) = (1 + e^{-z})^{-1}$ (Figure 5.1) compresse les entrées vers l'intervalle $(0, 1)$.

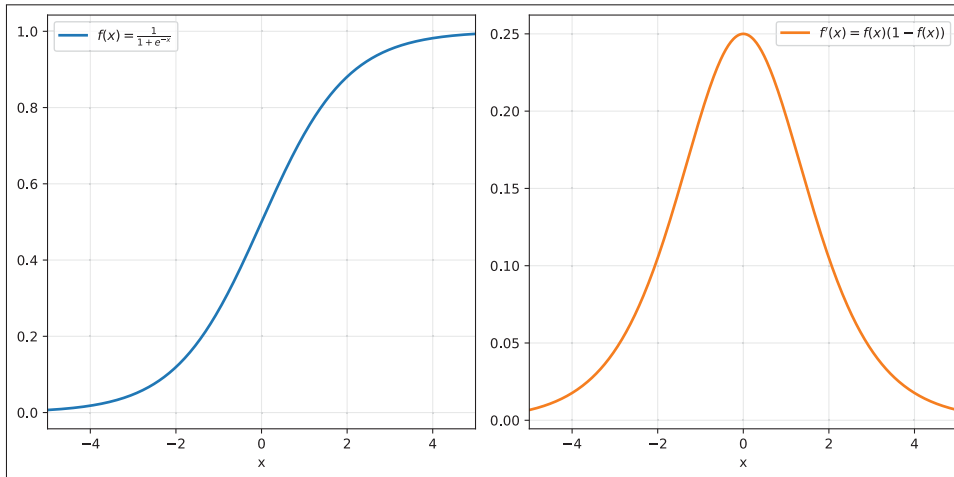


Figure 5.1 Fonction sigmoïde et sa dérivée

La tangente hyperbolique $\tanh(z)$ (Figure 5.2) transforme les entrées vers l'intervalle $(-1, 1)$ et est centrée en zéro, ce qui atténue une partie de l'asymétrie du sigmoïde.

L'unité linéaire rectifiée (ReLU) (Figure 5.3), définie comme $\sigma(z) = \max(0, z)$, laisse passer les valeurs positives inchangées et annule les entrées négatives.

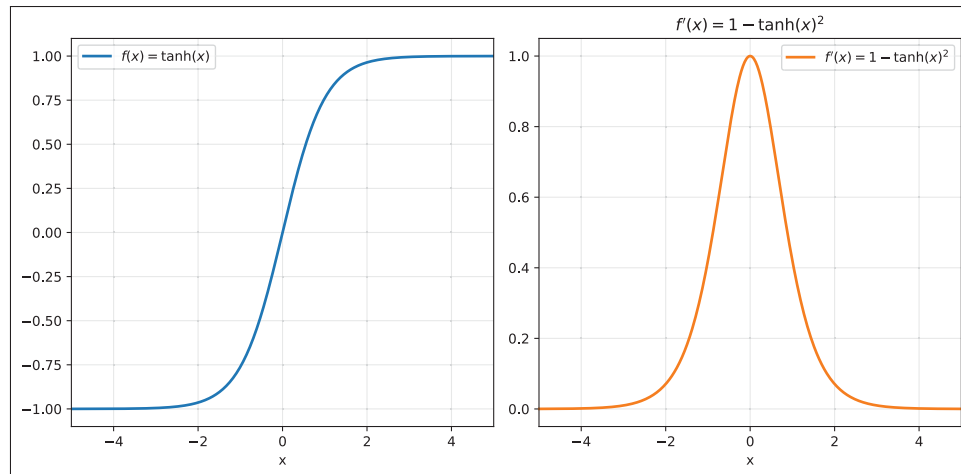


Figure 5.2 Fonction tangente hyperbolique et sa dérivée

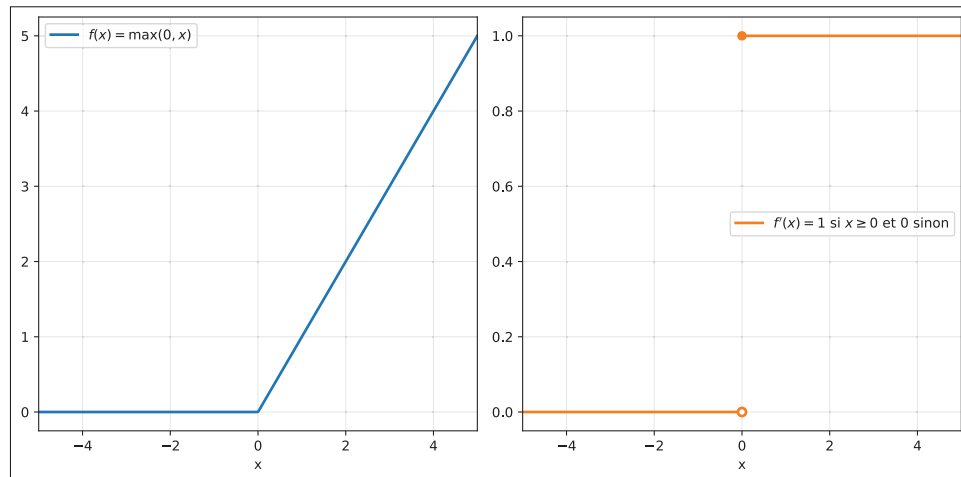


Figure 5.3 Fonction unité linéaire rectifiée et sa dérivée

Les neurones sont organisés en couches, où chaque couche reçoit les sorties de la couche précédente et produit une nouvelle représentation. Cette structure en couches est la source de l'expressivité des réseaux profonds : les couches inférieures tendent à détecter des motifs locaux simples dans l'entrée, tandis que les couches supérieures composent ces détecteurs en représentations progressivement plus abstraites (Bishop & Bishop, 2024). Deux décisions clés de conception gouvernent la capacité :

- la largeur : le nombre de neurones par couche,

- la profondeur : le nombre de couches.

Un réseau plus profond à largeur modérée peut représenter la même classe de fonctions avec beaucoup moins de paramètres en partageant et en réutilisant des détecteurs de caractéristiques de bas niveau. Il est à noter que les largeurs des couches peuvent être différentes.

Une distinction d'importance pratique sépare l'entraînement de l'inférence. L'architecture du réseau, c'est-à-dire le nombre de couches, leur largeur et la fonction d'activation, est fixée avant l'entraînement. Les composantes apprises sont les paramètres du réseau : les matrices de poids w_i et les vecteurs de biais b_i de chaque couche. Pendant l'entraînement, ces paramètres sont ajustés itérativement pour réduire une mesure de l'erreur de prédiction. Au cours de l'inférence, les paramètres sont maintenus fixes et le réseau agit comme une fonction déterministe.

Formellement, un réseau de neurones transforme une entrée $\mathbf{x} \in \mathbb{R}^{n_0}$ en une sortie $\mathbf{y} \in \mathbb{R}^{n_L}$ à travers L couches successives :

$$\mathbf{h}^{(l)} = f_l(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}), \quad l = 1, \dots, L \quad (5.2)$$

où $\mathbf{h}^{(0)} = \mathbf{x}$, $\mathbf{W}^{(l)}$ et $\mathbf{b}^{(l)}$ sont la matrice de poids et le vecteur de biais de la couche l , et f_l est la fonction d'activation appliquée élément par élément. Il est à noter que les fonctions d'activation peuvent être différentes pour chaque couche.

Le théorème d'approximation universelle (Hornik, Stinchcombe & White, 1989) stipule qu'un réseau de neurones avec une seule couche cachée et une largeur suffisante peut approximer n'importe quelle fonction continue sur un domaine compact pour une précision arbitraire (Bishop & Bishop, 2024). Dans la pratique, les réseaux plus profonds à largeur modérée généralisent mieux et s'entraînent plus efficacement sur les entrées à haute dimension, ce qui motive l'utilisation d'architectures profondes dans un large éventail de tâches d'apprentissage.

L'apprentissage profond a trouvé des applications directes dans la course autonome, au-delà des méthodes d'apprentissage par renforcement de ce chapitre. Chrosniak, Ning & Behl (2024) ont démontré que l'architecture de *Physics Informed Neural Network* (PINN) peut apprendre des

modèles de dynamique de véhicule directement à partir de données offrant une alternative aux modèles analytiques de pneus et de dynamique.

5.2.2 Entraînement, rétropropagation et optimisation

L'apprentissage dans un réseau de neurones est formulé comme un problème d'optimisation : trouver les paramètres $\theta = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^L$ qui minimisent une fonction de coût $\mathcal{L}(\theta)$. Cette fonction mesure la différence entre les prédictions du réseau et les valeurs vraies des données d'entraînement. On peut imaginer une surface de coût : les paramètres peuvent être considérés comme des coordonnées dans un espace de haute dimension. La fonction de coût définit une altitude à chaque point. L'entraînement est le processus de descente de la surface de coût vers le minimum de la fonction de coût.

Un optimiseur est un algorithme qui utilise l'information du gradient pour mettre à jour les paramètres itérativement afin que $\mathcal{L}$ diminue. L'optimiseur le plus simple, la descente de gradient, déplace les paramètres dans la direction du gradient négatif :

$$\theta \leftarrow \theta - \eta \nabla_{\theta} \mathcal{L} \quad (5.3)$$

où $\eta > 0$ est le taux d'apprentissage. Le gradient $\nabla_{\theta} \mathcal{L}$ lui-même est calculé par rétropropagation, un algorithme qui applique la règle de dérivation en chaîne à travers le graphe de calcul du réseau (Goodfellow *et al.*, 2016). Un seul passage inverse calcule tous les gradients simultanément, rendant possible l'entraînement de réseaux profonds.

En pratique ce gradient n'est pas calculé sur toutes les données d'entraînement, mais sur un petit sous-ensemble. Le gradient est approximé à l'aide de ce sous-ensemble échantillonné aléatoirement. L'avantage est la réduction des coûts de calcul et l'introduction de stochasticité dans la mise à jour des poids.

Un risque important est le surapprentissage : un réseau peut mémoriser les exemples d'entraînement plutôt que d'apprendre une correspondance généralisable. Pour éviter ce problème, un jeu de

validation est utilisé pour vérifier que la fonction de coût minimisée sur cet ensemble de données diminue. En effet, si les performances sur les données d'entraînement diminuent alors que les performances sur les données de validation augmentent, le réseau de neurones ne généralise pas, mais mémorise les exemples d'entraînement.

5.2.3 Processus de décision de Markov

Dans notre cas, nous cherchons à approximer une fonction qui prend en entrée une observation à un instant t de l'état s_t et qui retourne une action a_t . Une fois cette action prise, on se retrouve dans un nouvel état s_{t+1} . Nous formalisons ce problème résolu par apprentissage par renforcement comme un processus de décision de Markov (MDP) défini par le tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, où $\mathcal{S}$ est l'espace d'état, $\mathcal{A}$ est l'espace d'action, $P(s' | s, a)$ est la probabilité de transition de l'état s à l'état s' par l'action a , $r(s, a)$ est la fonction de récompense, et $\gamma \in [0, 1)$ est le facteur d'actualisation. La propriété de Markov exige que la probabilité de transition ne dépende que de l'état actuel et de l'action. L'objectif est de trouver une politique $\pi(a | s)$ qui maximise l'espérance du retour cumulatif actualisé :

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \quad (5.4)$$

L'équation de Bellman pour la fonction action-valeur $Q^\pi(s, a)$ exprime la relation récursive :

$$Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P} [\mathbb{E}_{a' \sim \pi} [Q^\pi(s', a')]] \quad (5.5)$$

La fonction action-valeur a pour objectif d'évaluer le rendement attendu avec l'action a dans l'état s sous la politique π .

Pour l'environnement de simulation F1TENTH utilisé, l'agent reçoit une observation partielle de l'environnement comme décrit dans la section 2.3.4. Bien qu'il s'agisse d'une représentation tronquée plutôt que de l'état complet de l'environnement, le formalisme est conservé et on se

retrouve dans un processus de décision de Markov partiellement observable (*Partially Observable Markov Decision Process*).

5.2.4 Méthodes de gradient de politique

Les méthodes de gradient de politique paramètrent directement la politique $\pi_\theta(a \mid s)$ et optimisent les paramètres θ en se déplaçant dans la direction du gradient de l'espérance du retour. Le théorème du gradient de politique (Sutton & Barto, 2018) donne :

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta} [\nabla_\theta \log \pi_\theta(a \mid s) \cdot Q^{\pi_\theta}(s, a)] \quad (5.6)$$

où $Q^{\pi_\theta}(s, a)$ est la fonction action-valeur sous la politique actuelle. Dans la pratique, Q^π est remplacé par une estimation d'avantage $\hat{A}_t = Q^\pi(s_t, a_t) - V^\pi(s_t)$.

Les méthodes acteur-critique réduisent la variance en apprenant à la fois une politique (acteur) et une fonction de valeur (critique) simultanément (Sutton & Barto, 2018). Le critique fournit des estimations d'avantages à faible variance, tandis que l'acteur améliore la politique à l'aide de ces estimations. Cette structure est à la base de l'algorithme SAC.

5.2.5 Soft Actor-Critic

Soft Actor-Critic (Haarnoja, Zhou, Abbeel & Levine, 2018a; Haarnoja *et al.*, 2018b) est un algorithme acteur-critique *off-policy* fondé sur le principe d'entropie maximale (Haarnoja *et al.*, 2018a). L'idée centrale est d'augmenter l'objectif standard de maximisation de récompense cumulative avec un terme d'entropie de politique :

$$\pi^* = \arg \max_{\pi} \sum_t \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} [r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot \mid s_t))] \quad (5.7)$$

Ce terme d'entropie incite l'agent à rester stochastique, ce qui favorise l'exploration des états non encore visités, permet de capturer plusieurs modes de comportement quasi-optimaux, et

confère une robustesse naturelle aux erreurs de modèle (Haarnoja *et al.*, 2018a). La politique déterministe conventionnelle est retrouvée à la limite $\alpha \rightarrow 0$.

Le coefficient de température α pondère le compromis entre récompense et entropie. Choisir une valeur adaptée est non trivial car l'échelle des récompenses varie entre les tâches et évolue au cours de l'entraînement. Haarnoja *et al.* (2018b) résolvent ce problème en formulant une optimisation sous contrainte d'entropie minimale : la température est apprise par descente de gradient duale de façon à maintenir l'entropie de la politique au-dessus d'un seuil cible $\mathcal{H}^* = -\dim(\mathcal{A})$, éliminant ainsi le réglage manuel de ce paramètre. Cette valeur, négative et proportionnelle à la dimension de l'espace d'action, est un heuristique empirique qui s'est avérée bien fonctionner sur l'ensemble des tâches de référence sans réglage par tâche (Haarnoja *et al.*, 2018b). Elle contraint la politique à rester suffisamment stochastique sans lui imposer d'être maximalement aléatoire.

L'algorithme repose sur trois composants qui interagissent à chaque itération d'entraînement (figure 5.4). Un *replay buffer* stocke les transitions $(s_t, a_t, r_t, s_{t+1}, d_t)$ collectées avec la politique courante, permettant la réutilisation des données pour un entraînement hors-politique et améliorant l'efficacité d'échantillonnage. L'acteur π_θ est un réseau de neurones dont les deux têtes de sortie produisent, pour chaque état s , la moyenne $\mu_\theta(s)$ et le logarithme de l'écart-type $\log \sigma_\theta(s)$ de la distribution d'action. Prédire $\log(\sigma)$ plutôt que σ directement garantit $\sigma_\theta(s) = \exp(\log \sigma_\theta(s)) > 0$.

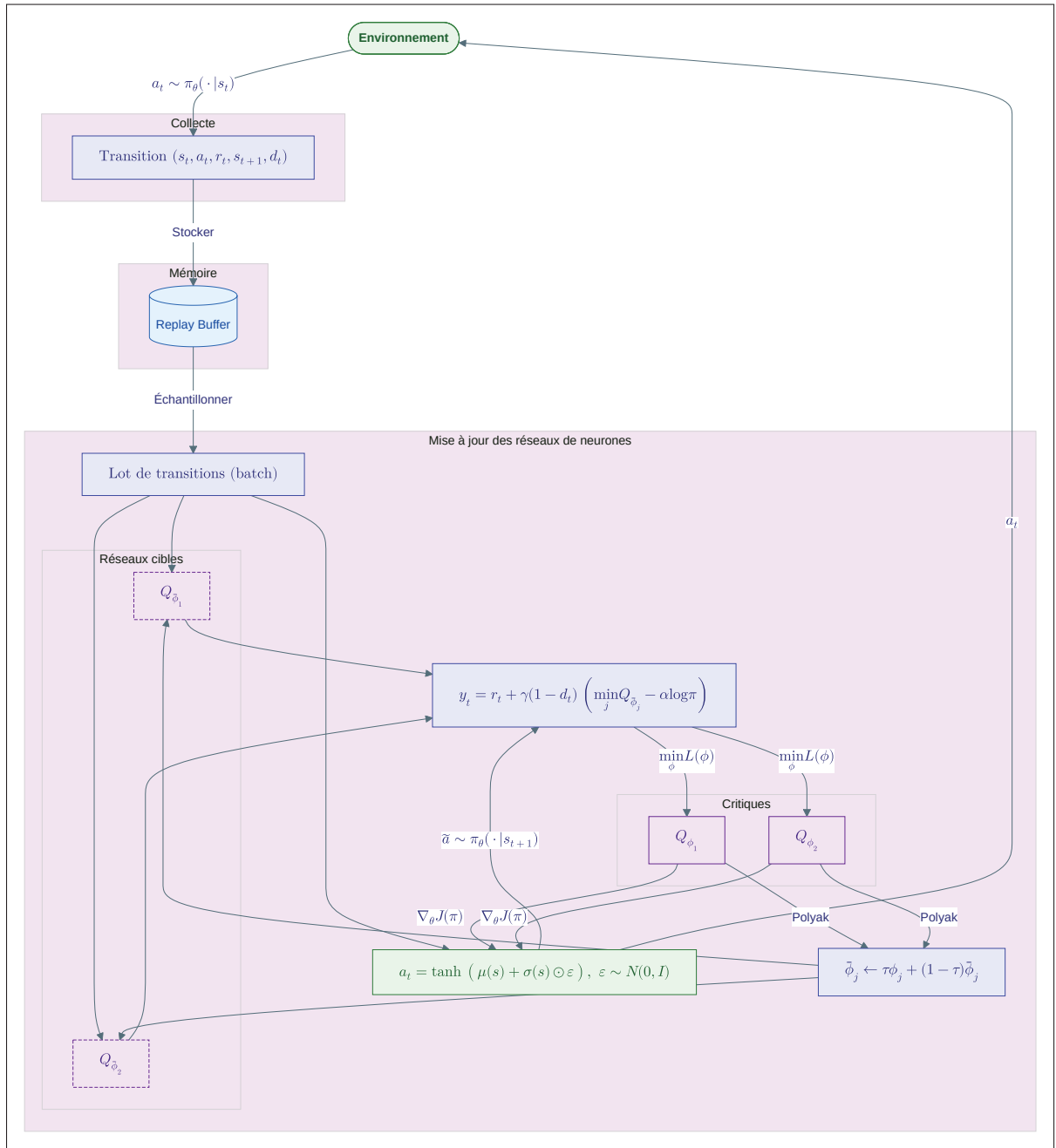


Figure 5.4 Vue d'ensemble de l'algorithme *Soft Actor-Critic*

Les actions sont ensuite tirées via l'astuce de reparamétrisation, qui permet de propager le gradient à travers l'échantillonnage :

$$a = \tanh(\mu_\theta(s) + \sigma_\theta(s) \odot \varepsilon), \quad \varepsilon \sim \mathcal{N}(0, I) \quad (5.8)$$

Le critique est constitué de deux réseaux de neurones Q_{ϕ_1} et Q_{ϕ_2} entraînés indépendamment. Leur rôle est de minimiser le résidu de Bellman (Haarnoja *et al.*, 2018b) :

$$y_t = r_t + \gamma(1 - d_t) \left(\min_{j=1,2} Q_{\bar{\phi}_j}(s_{t+1}, \tilde{a}_{t+1}) - \alpha \log \pi_{\theta}(\tilde{a}_{t+1} | s_{t+1}) \right) \quad (5.9)$$

Prendre le minimum des deux prédictions réduit le biais de surestimation connu pour dégrader les performances des méthodes de type *Value based* (Haarnoja *et al.*, 2018a). Deux réseaux cibles $Q_{\bar{\phi}_1}, Q_{\bar{\phi}_2}$ dont les paramètres suivent une moyenne pondérée exponentielle (*soft update* de Polyak) :

$$\bar{\phi}_j \leftarrow \tau \phi_j + (1 - \tau) \bar{\phi}_j \quad (5.10)$$

stabilisent l'apprentissage en lissant l'évolution des cibles de Bellman (Haarnoja *et al.*, 2018a).

5.3 Mise en œuvre

L'entraînement de l'agent RL est fait avec l'implémentation de SAC dans CleanRL (Huang *et al.*, 2022), qui fournit des implémentations d'algorithmes RL dans des fichiers uniques. CleanRL a été choisi pour sa transparence et sa facilité de modification. L'agent est entraîné sur F1TENTH Gym où il interagit avec la simulation à une fréquence de 10 Hz.

5.3.1 Architecture des réseaux de neurones

L'acteur est composé de deux couches cachées de 256 et 128 neurones avec des fonctions d'activation ReLU.

Le critique est constitué de deux réseaux Q jumeaux, chacun avec des couches cachées de 256 et 128 unités et des activations ReLU.

5.3.2 Espace d'état et espace d'action

L'observation et les actions reposent sur les mécanismes de F1TENTH Gym présentés aux sections 2.3.4 et 2.3.5.

L'observation se compose de sept états dynamiques et d'un *look ahead*. Les états sont la déviation latérale, la déviation de cap, les vitesses longitudinale v_x et latérale v_y , la vitesse de lacet $\dot{\psi}$, l'angle de glissement β et l'angle de direction δ . Le *look ahead* a un paramètre d'échantillonnage de 10 points espacés de 1,5 m. L'observation totalise ainsi 67 valeurs.

L'espace d'action regroupe deux actions continues : l'accélération longitudinale et l'angle de direction.

5.3.3 Conception de la récompense

Plutôt que de n'attribuer une récompense qu'à la fin d'un épisode, on emploie une méthode de *reward shaping*. Elle consiste à donner à chaque pas de temps une récompense positive ou négative selon l'action effectuée et l'état de l'agent. La fonction de récompense construite encourage une progression rapide sur le circuit tout en pénalisant les comportements instables et les commandes saccadées. La récompense combine quatre termes :

$$r_t = k_s \Delta s_t + k_\kappa \mathbb{1}_{|\beta_t| > \beta_0} |\beta_t| \|v_t\| + k_\delta \frac{|\Delta \delta_t|}{\Delta t} + k_a \frac{|\Delta a_t|}{\Delta t} \quad (5.11)$$

Le **premier terme** récompense la progression le long de la ligne centrale. Il est le seul signal positif. Il incite l'agent à parcourir le circuit aussi vite que possible.

Le **terme de glissement** pénalise les états instables du véhicule. Il ne s'active que lorsque **l'angle de glissement $|\beta_t|$ dépasse un seuil β_0** , et croît alors avec la vitesse $\|v_t\|$ pour décourager d'autant plus les dérapages à haute vitesse.

Les **termes de direction et d'accélération** pénalisent les variations de commande entre deux pas de temps consécutifs, limitant ainsi l'oscillation des actions et favorisant une conduite fluide.

Les poids de chaque terme sont consignés dans le tableau 5.1.

Une pénalité supplémentaire, proportionnelle à la vitesse et au dépassement latéral, s'applique lorsque le véhicule franchit les limites de la piste, et la récompense de progression y est alors annulée : $k_{prog} \leftarrow 0$.

Lors d'une terminaison, la récompense instantanée n'est plus celle de l'équation 5.11, mais une récompense négative dédiée. On constitue une liste de pénalités candidates initialisée à -1 , à laquelle on ajoute, pour chaque condition de terminaison active, une pénalité dont l'amplitude reflète la gravité de la situation :

- tête-à-queue : pénalité proportionnelle à la vitesse et à l'angle de glissement,
- progrès négatif persistant : pénalité constante de -4 ,

- sortie de piste : l’opposé de la pénalité de dépassement latéral décrite ci-dessus.

Les valeurs de ces pénalités ont été obtenues de façon empirique jusqu’à obtenir un comportement satisfaisant. La récompense terminale retenue est le minimum de ces trois termes négatifs, c’est-à-dire la pénalité la plus sévère parmi les conditions déclenchées.

5.3.4 Structure des épisodes

À l’entraînement, le véhicule est réinitialisé à une position aléatoire le long de la ligne centrale avec une vitesse initiale tirée uniformément dans l’intervalle $[1, 15]$ m/s, ce qui diversifie les états observés. À l’évaluation, il part de la ligne de départ à une vitesse fixe de 10 m/s. Un épisode est tronqué après une durée maximale de 25 s, soit 250 pas de temps.

La distinction entre terminaison et troncation est critique pour une estimation correcte de la fonction de valeur (Sutton & Barto, 2018).

5.3.5 Hyperparamètres

Les hyperparamètres SAC sont résumés dans le tableau 5.1.

Tableau 5.1 Hyperparamètres d’entraînement

Paramètre	Valeur
Taux d’apprentissage (acteur)	3×10^{-4}
Taux d’apprentissage (critique)	1×10^{-3}
Facteur d’actualisation γ	0,99
Taille du <i>replay buffer</i>	10^6
Taille d’un lot	256
Lissage des cibles τ	0,005
Température initiale α	0,2 (auto-ajustée)
Entropie cible $\mathcal{H}^*$	-2
Étapes d’échauffement	50 000
Fréquence de mise à jour de l’acteur	2
Couches cachées acteur	[256, 128]
Couches cachées critique	[256, 128]
Pas de temps total	1×10^6
Poids de progression k_s	2
Poids de glissement k_κ	-0,6
Seuil de glissement β_0	0,2 rad
Poids de direction k_δ	-0,1
Poids d’accélération k_a	-0,05

5.4 Résultats

Les résultats présentés correspondent à l'entraînement d'un agent RL avec l'algorithme SAC sur le circuit de Montréal, avec le modèle de bicyclette dynamique et la configuration de la section 5.3. L'entraînement couvre 1 000 000 pas de temps soit l'équivalent de 100 000 secondes de simulation et dure environ 3 heures sur l'ordinateur décrit à la section 2.1. L'agent RL est évalué tous les 5000 pas en partant de la ligne de départ à une vitesse initiale de 10 m/s.

5.4.1 Courbes d'apprentissage

La figure 5.5 comporte deux mesures d'évaluation en fonction du pas de temps d'entraînement : la récompense cumulative et la distance curviligne parcourue le long de la ligne centrale.

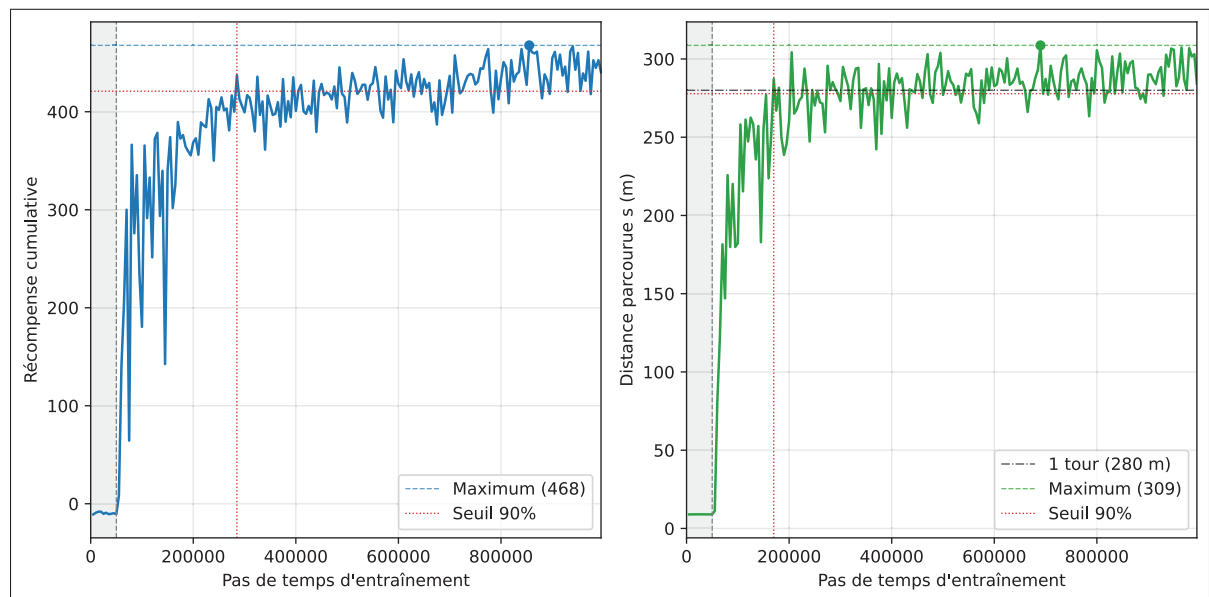


Figure 5.5 Récompense cumulative et progression cumulative d'évaluation sur le circuit de Montréal en fonction des pas de temps d'entraînement. La zone grisée indique la phase d'échauffement (5×10^4 pas)

Pendant les 50 000 premiers pas, qui correspondent à la phase d'échauffement, la récompense reste négative (environ -10) et la progression quasi nulle. Durant cette phase, il n'y a pas de mise à jour de gradient. Le *replay buffer* est rempli par des transitions avec lesquelles les actions sont choisies aléatoirement ce qui permet à l'agent de découvrir des états variés. Cette étape

garantit que l'apprentissage débute sur un ensemble de données suffisamment diversifié, ce qui stabilise les mises à jour ultérieures.

Après cette phase d'échauffement, on procède périodiquement à des mises à jour de gradient. La récompense croît brusquement peu après le début en passant de valeurs négatives à plus de 300 en une quinzaine de milliers de pas. Cette montée rapide est due à la réutilisation des transitions accumulées dans le *replay buffer*.

Après, la progression se stabilise rapidement : elle atteint 90 % de son maximum vers 170 000 pas et plafonne ensuite autour de 285 m. Un tour complet du circuit de Montréal correspond à une progression d'environ 280 m, l'agent est donc capable de boucler un tour entier sans sortie de piste. Ce plateau est borné par la durée de l'épisode d'évaluation (25 s), qui ne laisse que le temps de parcourir approximativement un tour complet. Une fois que l'agent a appris à boucler un tour, il faut s'attacher à la progression de l'accumulation des récompenses. La récompense continue de s'affiner plus longtemps, atteignant 90 % de son maximum vers 285 000 pas et son maximum de 467,7 à 855 000 pas. Ce décalage indique qu'après avoir appris à boucler le tour, l'agent continue d'optimiser la qualité de sa conduite via les pénalités de glissement et d'oscillation des entrées, sans gain de distance supplémentaire.

Les deux courbes restent non monotones après convergence. Cette variabilité provient en partie de l'ajustement automatique de la température d'entropie, qui maintient une exploration résiduelle.

5.4.2 Efficacité d'échantillonnage

L'efficacité d'échantillonnage est mesurée par le nombre de pas d'environnement nécessaire pour atteindre un seuil de performance, fixé ici à 90 % de la meilleure performance observée. Cette mesure est particulièrement pertinente pour SAC, car chaque transition stockée dans le *replay buffer* peut être réutilisée pour plusieurs mises à jour de gradient.

En retranchant les 50 000 pas d'échauffement, l'agent atteint le seuil de 90 % de progression après environ 120 000 pas d'apprentissage effectif, et le seuil de 90 % de récompense après environ 235 000 pas (figure 5.6). Au-delà, l'entraînement se poursuit jusqu'à 1 000 000 pas sans gain marginal notable. Sur l'intervalle de 300 000 à 1 000 000 pas, la récompense d'évaluation se maintient à 426 ± 21 et la progression à 285 ± 12 m. La majeure partie de la performance est donc acquise dans le premier quart de l'entraînement.

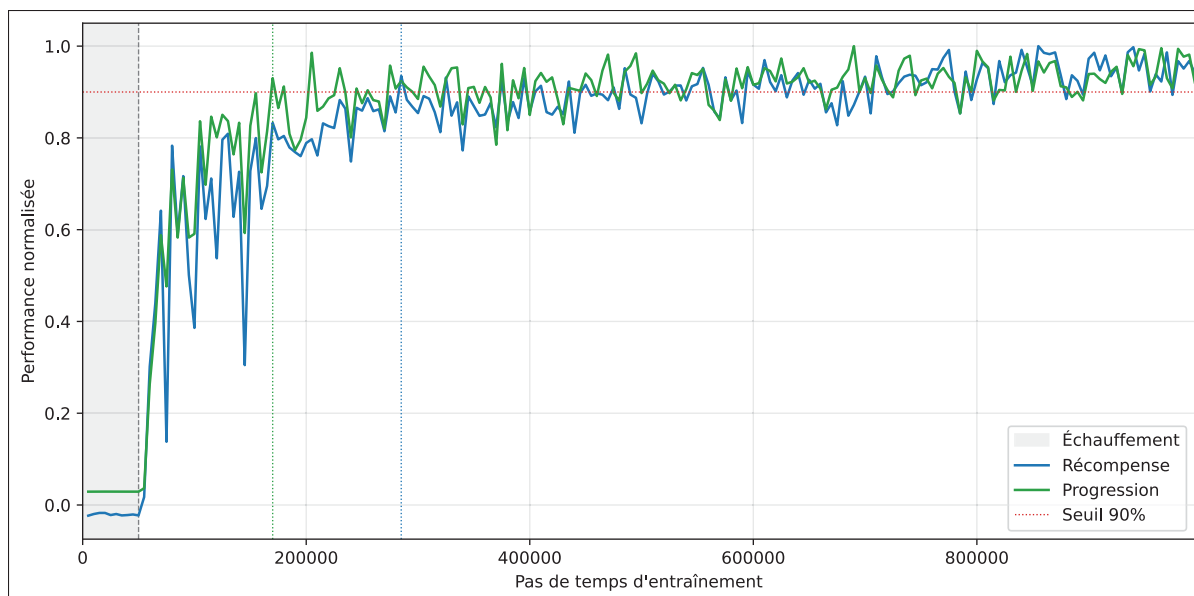


Figure 5.6 Récompense et progression d'évaluation normalisées par leur maximum, en fonction du pas de temps d'entraînement

5.4.3 Analyse de trajectoire

La politique est évaluée sur un tour fermé du circuit de Montréal, avec la même condition initiale de vitesse que dans les expériences précédentes. L'objectif de cette analyse est de vérifier si la stratégie apprise présente des caractéristiques de pilotage cohérentes : choix de trajectoire, adaptation de la vitesse aux virages, limitation du dérapage et régularité des commandes.

La figure 5.7 superpose la trajectoire de l'agent aux bordures réelles du circuit et à la ligne de course à courbure minimale du chapitre 3.

Sur la figure 5.8, on peut voir pour l'agent RL qu'il y a des plateaux plus longs aux niveaux des extremums qui correspondent à être plus proche des bordures. Comparée à la ligne de référence IQP, la trajectoire de l'agent RL est plus agressive dans l'utilisation de la largeur disponible.

La politique retrouve partiellement la géométrie globale de la ligne de course, sans la reproduire strictement comme le montre la figure 5.9. L'analyse du profil de vitesse donne une explication à la divergence des trajectoires. Il faut rappeler tout de même que l'agent RL ne pas reçoit pas de trajectoire de course à suivre.

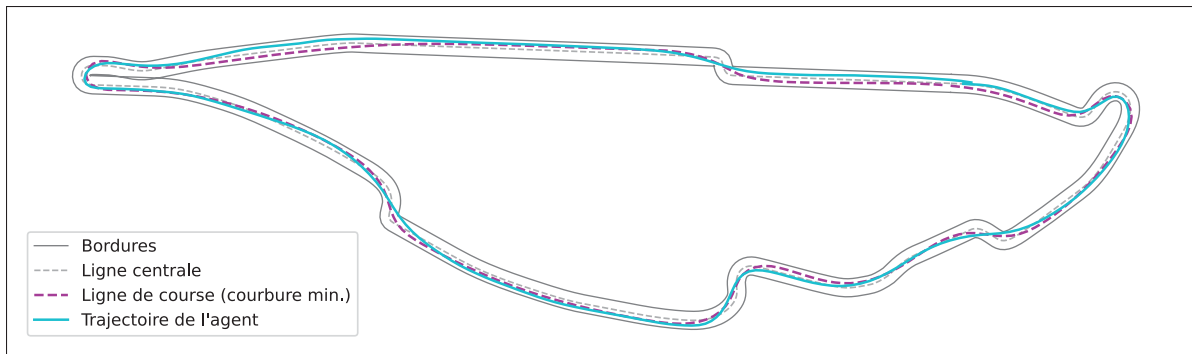


Figure 5.7 Trajectoire de l'agent SAC sur le circuit de Montréal, superposée à la ligne de course à courbure minimale calculée au chapitre 3

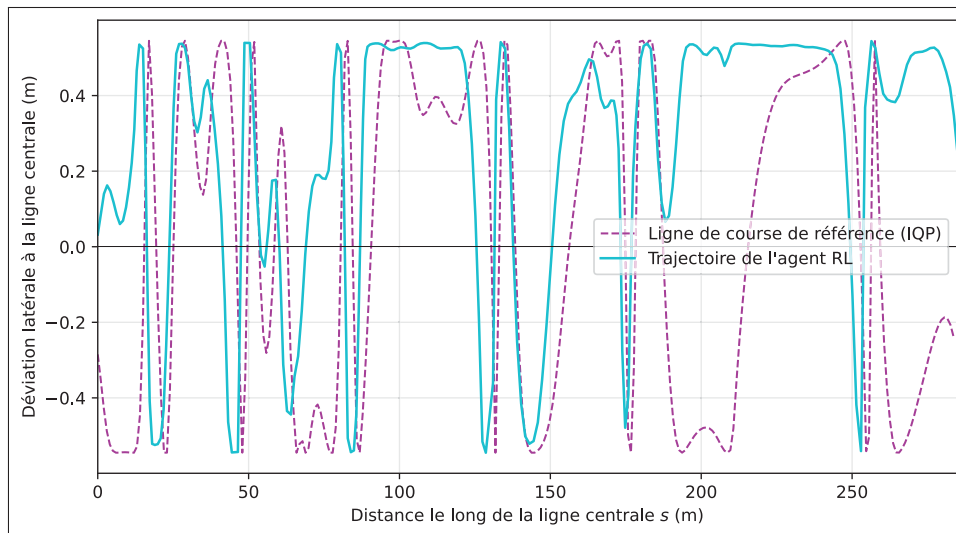


Figure 5.8 Déviation latérale par rapport à la ligne centrale de la trajectoire de référence et de la trajectoire suivie par l'agent RL

5.4.4 Profil de vitesse

La figure 5.10 montre que l'agent RL n'a pas un profil de vitesse cohérent avec le profil IQP planifié. Sa vitesse moyenne est de 11,16 m/s, avec un minimum de 4,33 m/s et un maximum de 14,43 m/s. Le profil planifié est plus agressif en moyenne, à 12,94 m/s, et atteint 20,00 m/s. Sur l'abscisse commune, l'écart moyen $v_{\text{plan}} - v_{\text{agent}}$ vaut 1,59 m/s et l'écart maximal atteint 8,60 m/s vers $s = 197,9$ m.

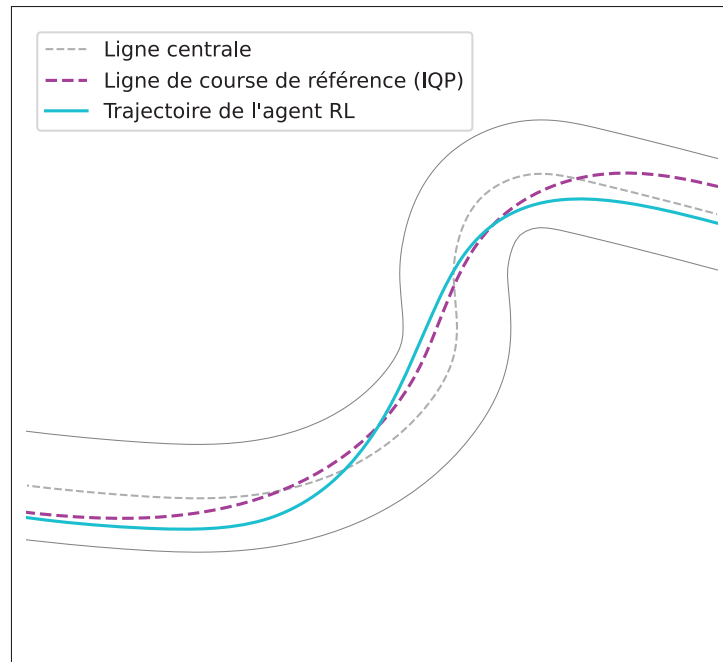


Figure 5.9 Superposition de la trajectoire IQP et de la trajectoire suivie par l'agent RL sur le sixième et septième virages du circuit de Montréal

Cette disparité de vitesse s'explique par le contrôle instable de la voiture où l'agent RL tente de maintenir une vitesse élevée en dérapant dans les virages. Tout au long de la course, l'angle de glisse β dépasse le régime dans lequel la voiture a un angle de glisse faible (figure 5.11). Sa valeur absolue moyenne est de 0,093 rad, tandis que le maximum atteint 0,819 rad de manière localisée. Le seuil de 0,2 rad, utilisé dans la fonction de récompense pour pénaliser les états de glissement importants, est dépassé sur 6,9 % de la distance parcourue. La pénalité ne supprime donc pas tous les dérapages, mais elle maintient la majorité du tour dans un régime de glissement. Ceci révèle que l'agent RL est capable de maintenir une vitesse élevée avec un glissement modéré sans sortir de piste.

5.4.5 Fluidité des actions

Les actions de l'agent RL présentées sur la figure 5.12 montrent que la politique apprise est bornée par les pénalités au niveau de ses actions. L'angle de direction présente des oscillations surtout au niveau des lignes droites. La commande d'accélération est plus fluide que la commande de direction. Les changements de signe restent nombreux, surtout pour la direction, mais ils ne

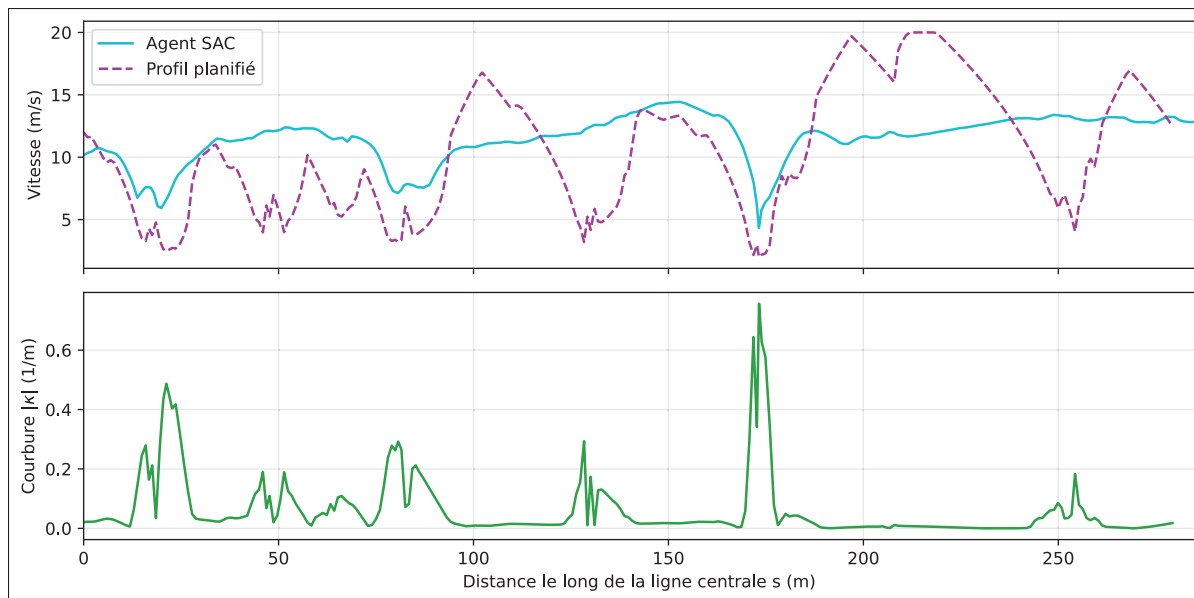


Figure 5.10 Profil de vitesse longitudinal de l'agent SAC comparé au profil planifié associé à la ligne de course à courbure minimale. La courbure absolue $|\kappa|$ localise les virages du circuit

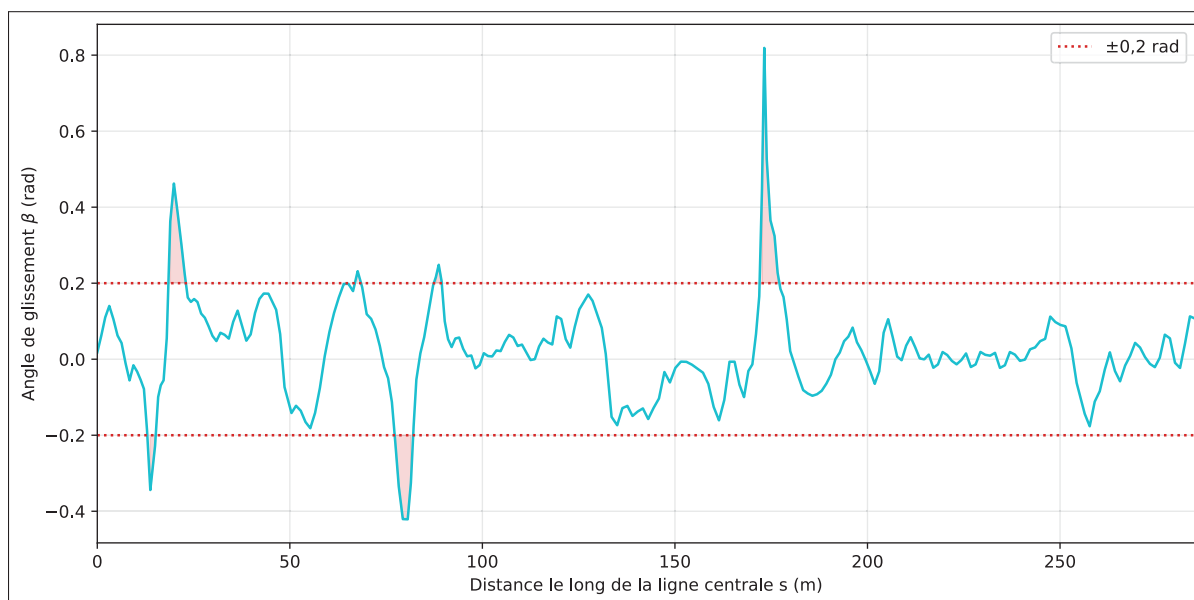


Figure 5.11 Angle de glissement β de l'agent RL

se traduisent pas par une perte de contrôle. Les pénalités d'oscillation des actions semblent donc

suffisantes pour obtenir une trajectoire exploitable, sans produire une commande parfaitement lisse.

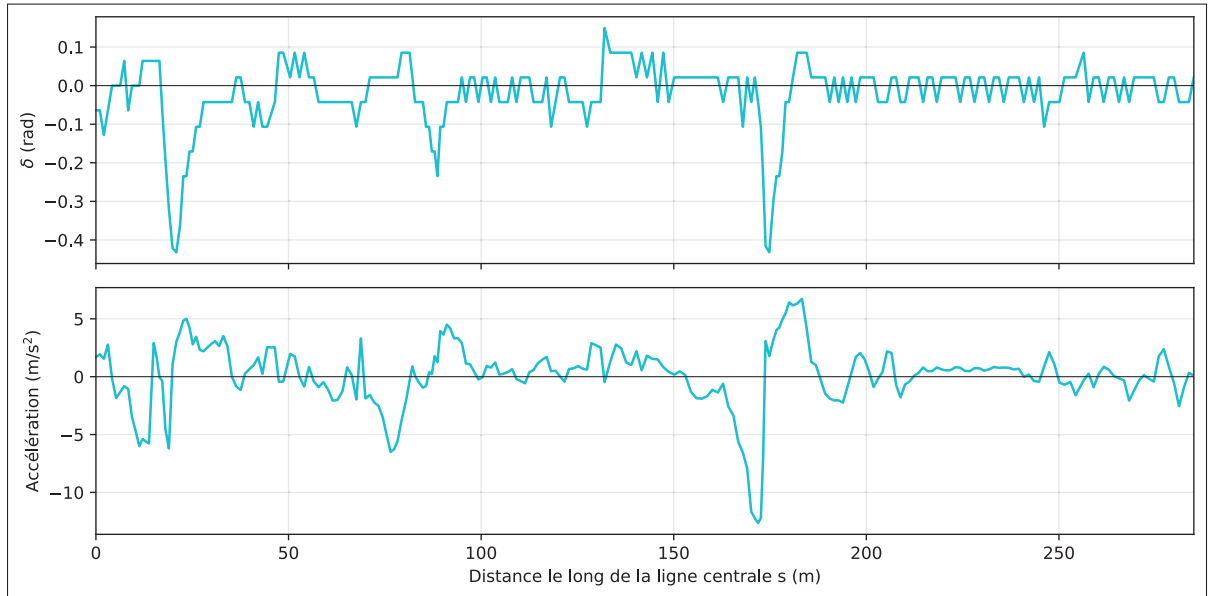


Figure 5.12 Commandes de direction et d'accélération de l'agent SAC le long du tour. La direction correspond à l'angle de braquage effectif du modèle dynamique

5.4.6 Comparaison avec le MPC

Cette sous-section s'attache à comparer la politique apprise par l'agent RL avec le contrôleur MPC. Une comparaison entre ces deux méthodes est assez naturelle car elles présentent des similitudes dans leur approche. L'une cherche à minimiser une fonction de coût, l'autre à maximiser une fonction de valeur qui ont toutes les deux une structure similaire. Le tableau 5.2 résume les performances obtenues avec chacun des contrôleurs.

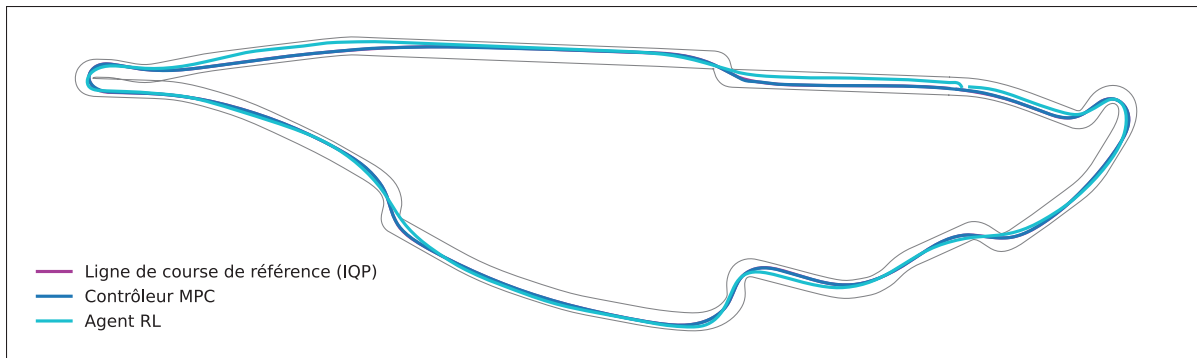


Figure 5.13 Trajectoires de référence, du MPC et de l'agent RL sur le circuit de Montréal

On observe sur la figure 5.13 qu'il y a une différence globale de trajectoire entre l'agent RL et le MPC.

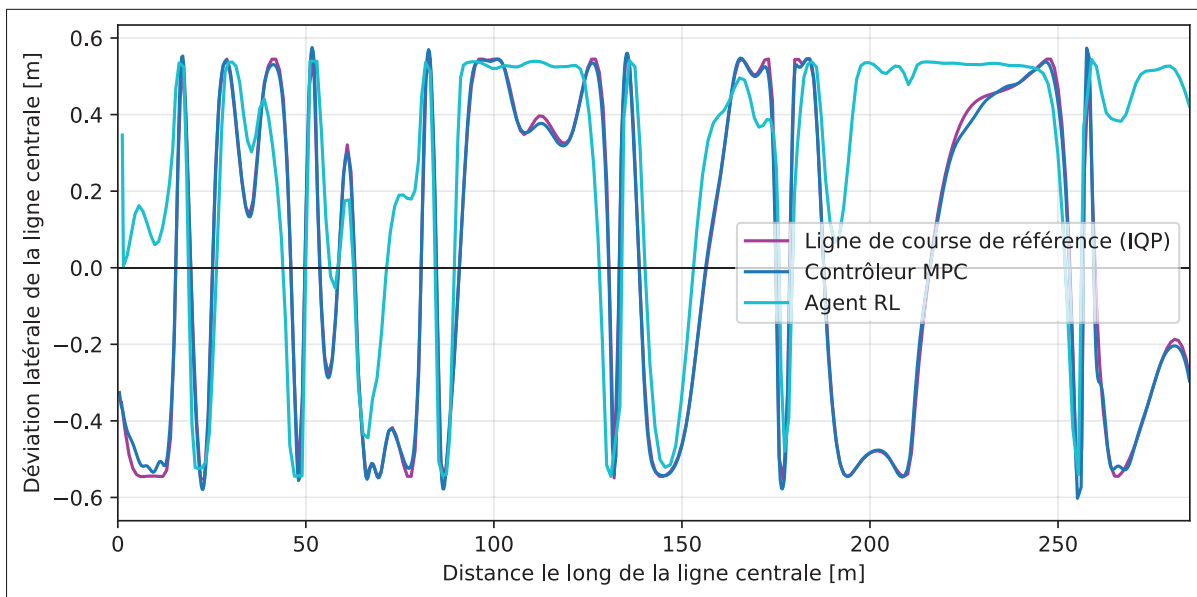


Figure 5.14 Déviation latérale du MPC et de l'agent RL

La trajectoire du MPC est quasiment confondue avec la trajectoire de référence comme le montre la figure 5.15.

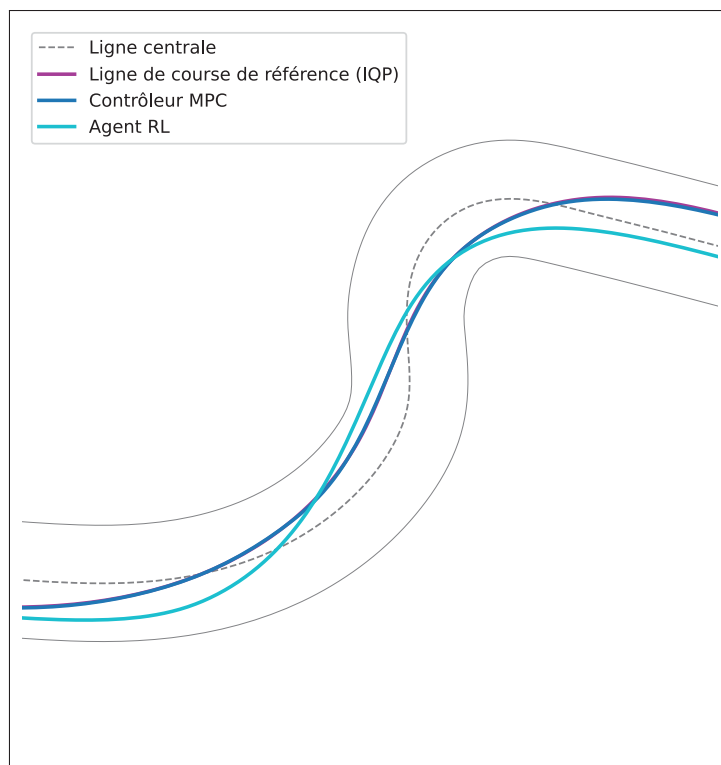


Figure 5.15 Superposition des trajectoires de référence, du MPC et de l'agent RL sur le sixième et septième virages du circuit de Montréal

Le temps au tour est de 35,40 s pour le MPC et de 24,90 s pour l'agent RL. Cette différence peut s'expliquer en partie par la différence de trajectoire. Mais une plus grande contribution à cette différence peut s'expliquer par les vitesses obtenues avec chacun des contrôleurs. Sur la figure 5.16, on peut voir que dans des zones où il est planifié de ralentir, l'agent RL maintient une vitesse plus élevée que le MPC. Ce qui permet à l'agent RL d'avoir une moyenne de vitesse de 11,16 m/s contre 7,92 m/s pour le MPC.

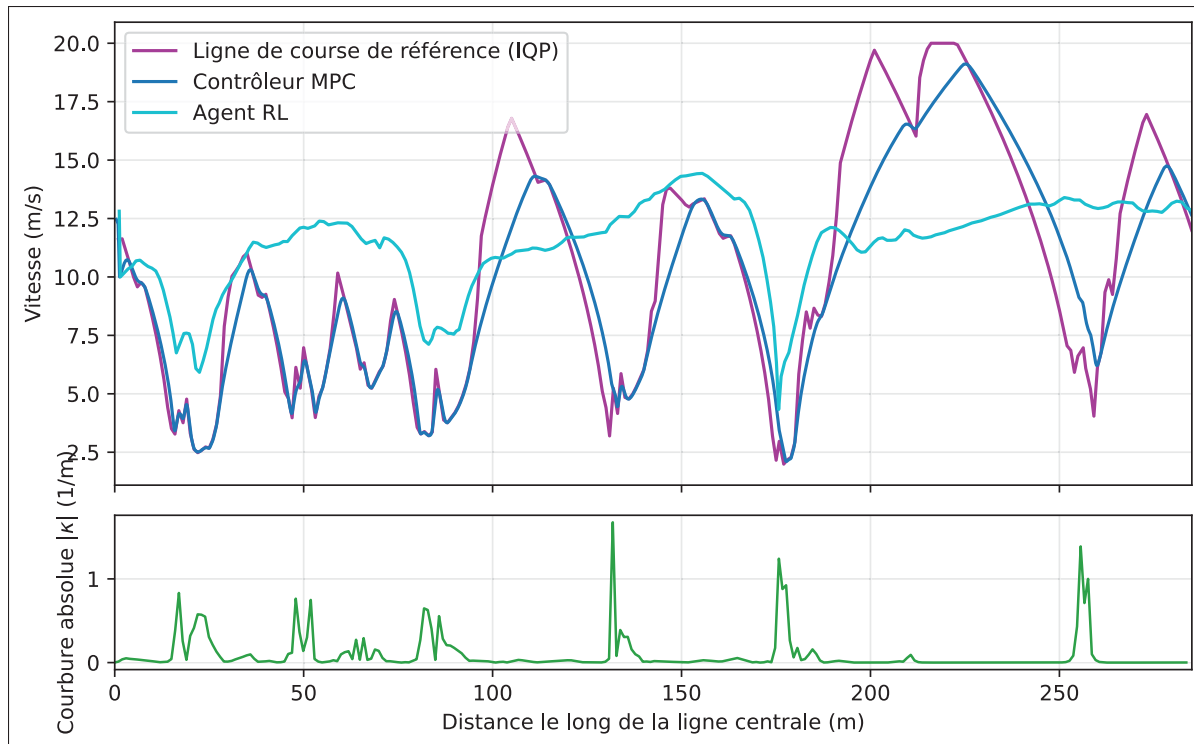


Figure 5.16 Profils de vitesse de référence, du MPC et de l'agent RL. La courbure absolue $|\kappa|$ localise les virages du circuit

Comme expliqué avant, l'agent RL apprend à maintenir une vitesse plus élevée en étant capable de garder le contrôle du véhicule avec des régimes d'angle de glisse que le MPC évite. Au maximum absolu, l'angle de glisse est de 0,819 rad pour l'agent RL contre 0,239 rad pour le MPC.

Le MPC est plus conservateur avec des extremums de commande de direction et d'accélération plus faibles que l'agent RL. La figure 5.18 montre que les commandes du MPC sont plus fluides que celles de l'agent RL.

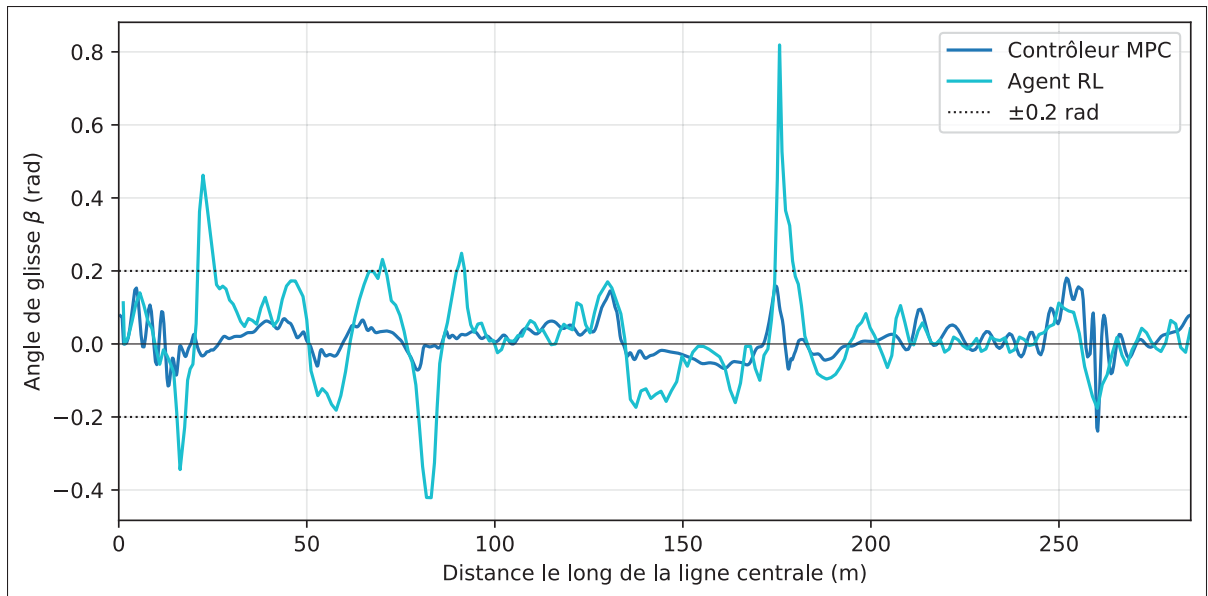


Figure 5.17 Angle de glissement du MPC et de l'agent RL

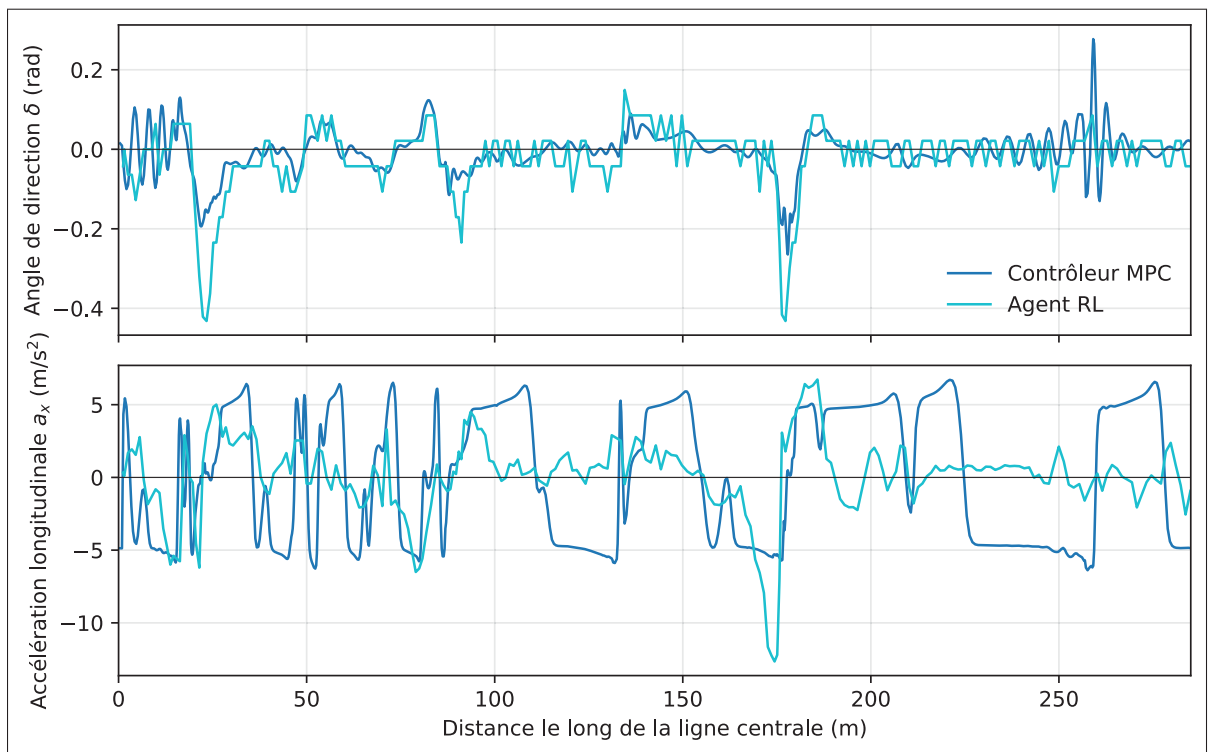


Figure 5.18 Commandes de direction et d'accélération du MPC et de l'agent RL

Tableau 5.2 Comparaison des performances du MPC et de l'agent RL

Métrique	MPC	RL
Temps au tour (s)	35.40	24.90
Fréquence d'opération (Hz)	25	10
Vitesse moyenne (m/s)	7.92	11.16
Vitesse max (m/s)	19.11	14.43
$\max \delta $ (rad)	0.277	0.432
$\max a_x $ (m/s ²)	6.71	12.66
$\max \beta $ (rad)	0.239	0.819

5.5 Conclusions

Ce chapitre montre qu'un agent entraîné avec l'algorithme SAC peut apprendre à piloter autour du circuit de Montréal sans sortir de piste en moins de 300 000 pas d'apprentissage effectif. La convergence vers une politique performante est obtenue en autorisant l'agent à explorer des états non désirables, par exemple des sorties de piste, plutôt qu'en les interdisant strictement. L'ajustement automatique de l'entropie évite par ailleurs le réglage manuel du compromis exploration-exploitation. Les courbes d'apprentissage (figure 5.5) révèlent deux phases distinctes : l'agent apprend d'abord rapidement à compléter un tour, puis affine plus lentement la qualité de sa conduite sous l'effet des pénalités de glissement et d'oscillation des commandes (section 5.3.3), sans gain de distance supplémentaire.

L'analyse de la politique finale (section 5.4.3) nuance toutefois ce succès. L'agent retrouve partiellement la géométrie globale de la ligne à courbure minimale (figure 5.7), comme en témoigne la corrélation modérée de leurs écarts latéraux, sans la reproduire strictement. Surtout, il atteint des vitesses élevées (figure 5.10) en exploitant un régime de glissement marqué (figure 5.11) plutôt qu'en suivant le profil de vitesse planifié : son allure moyenne reste supérieure à celle planifiée et celle obtenue avec le MPC. Le seuil de glissement de la fonction de récompense est franchi sur une fraction non négligeable du tour. Les pénalités d'oscillation suffisent à produire une trajectoire exploitable (figure 5.12), sans pour autant lisser parfaitement les commandes. L'agent démontre ainsi qu'une politique apprise par renforcement peut maîtriser le circuit de bout en bout à partir des seules observations en repère de Frenet, mais que les pénalités de la

fonction de récompense ne contraignent que partiellement le style de conduite. La comparaison avec le MPC permet de mettre en évidence les limites de l'agent RL qui adopte un comportement de conduite agressif, opposé à la stratégie du MPC qui vise un suivi de trajectoire proche de l'optimalité.

CONCLUSION ET RECOMMANDATIONS

Ce mémoire aborde la course autonome sur la plateforme F1TENTH en simulation, en partant de la modélisation du véhicule jusqu'à une politique de contrôle en passant par un planificateur de trajectoire hors ligne. Le travail sur lequel porte ce mémoire couvre l'infrastructure logicielle nécessaire à l'expérimentation, un planificateur de trajectoire s'appuyant sur l'optimisation, un contrôleur par commande prédictive et un agent entraîné par apprentissage par renforcement. La performance est considérée à la fois sous l'angle du temps au tour et sous celui de la fluidité du comportement.

6.1 Synthèse

Le chapitre 2 décrit l'infrastructure logicielle développée autour de F1TENTH Gym. Le simulateur a été adapté pour faciliter l'entraînement, l'évaluation et la comparaison de systèmes de commande de nature différente, avec des espaces d'observation et d'actions configurables, et un suivi de progression. Ces ajouts ne sont pas spécifiques à un algorithme donné et fournissent une base réutilisable pour la recherche sur F1TENTH Gym.

Le chapitre 3 présente un planificateur de trajectoire à courbure minimale, formulé comme un programme quadratique suivant l'approche de Heilmeier *et al.* (2020). Cette méthode fournit une trajectoire de course lisse ainsi qu'un profil de vitesse compatible avec les hypothèses dynamiques retenues. Elle joue le rôle de référence déterministe dans ce mémoire.

Le chapitre 4 décrit un contrôleur par commande prédictive (MPC) non linéaire, prenant explicitement en compte le modèle de véhicule et les contraintes de piste. Le MPC relie directement la commande au modèle et constitue un point de comparaison naturel face aux politiques apprises par sa nature prédictive.

Le chapitre 5 présente un agent RL entraîné avec l'algorithme *Soft Actor-Critic* (SAC) (Haarnoja *et al.*, 2018b) dans F1TENTH Gym, avec un suivi des expériences via MLflow (Zaharia *et al.*, 2018). Ce cadre facilite la traçabilité des essais, la comparaison des configurations d'entraînement et l'analyse des performances finales.

Ces trois derniers chapitres se lisent en termes de complémentarité plutôt que de concurrence. Aucune approche ne domine systématiquement les autres dans toutes les conditions. Le planificateur à courbure minimale fournit une référence géométrique et un indicateur de performance utiles pour structurer l'analyse. Le MPC apporte une commande interprétable, reliée explicitement à un modèle et à des contraintes, dont la charge de calcul se paye lors du déploiement. L'agent RL montre qu'une politique peut apprendre un comportement de course à partir d'interactions avec un simulateur, en déplaçant le coût de calcul vers une phase d'entraînement préalable au déploiement. Ce contraste recoupe celui, plus général, entre deux paradigmes : l'optimisation apporte structure, interprétabilité et respect des contraintes, là où l'apprentissage automatique apporte flexibilité et potentiel d'adaptation. La voie la plus prometteuse semble dès lors être celle de méthodes hybrides, capables d'exploiter le meilleur des deux paradigmes : la planification et l'apprentissage.

6.2 Limitations et hypothèses

6.2.1 Suppositions simplificatrices

Plusieurs hypothèses simplifient le problème étudié. L'état du véhicule (e.g., position, vitesse et orientation) est supposé connu exactement à partir du simulateur, alors qu'un déploiement physique nécessiterait de l'estimer à partir de capteurs bruités. La géométrie de la piste est considérée comme entièrement connue et statique. Enfin, le modèle dynamique employé repose sur une modélisation simplifiée du pneu, qui ne couvre pas toute la richesse des phénomènes observables en conditions réelles.

6.2.2 Limites méthodologiques

Les expériences sont menées uniquement en simulation, sans validation sur véhicule réel. L'évaluation ne porte que sur un seul circuit, et la généralisation à de nouveaux circuits n'est pas étudiée. Une recherche exhaustive d'hyperparamètres n'a pas été réalisée, les valeurs retenues s'appuyant sur les recommandations de la littérature et les valeurs par défaut de l'implémentation

utilisée. Par conséquent, la transférabilité directe des performances vers une plateforme physique ne peut être conclue à ce stade.

6.3 Travaux futurs

6.3.1 Déploiement futur

Une première direction porte sur le cadre expérimental et consiste à structurer un pipeline simulation vers réalité, incluant l'identification de paramètres, l'ajout de bruit dans les observations d'états et des protocoles de validation progressive sur le véhicule physique F1TENTH. Par ailleurs, l'utilisation d'un simulateur avec des modèles dynamiques de véhicule et de pneus plus réalistes afin de réduire l'écart entre simulation et réalité, pourrait être étudiée.

6.3.2 Course multi-agents

L'extension à la course multi-agents permettrait d'introduire des considérations de théorie des jeux absentes du cadre actuel. Cela ouvre la porte à l'adaptation en ligne au comportement d'un adversaire et l'arbitrage entre performance et sécurité lors des dépassements (Sinha *et al.*, 2020).

6.3.3 Rapprochement entre optimisation et apprentissage

Le rapprochement entre optimisation et apprentissage est un axe qui traverse aujourd'hui plusieurs domaines de la robotique au-delà de la course autonome. Une première voie est celle des *learning-based MPC*, où les données servent à améliorer le modèle de prédiction, à ajuster le coût et les contraintes, ou encore à garantir le respect des contraintes d'une politique apprise (Hewing *et al.*, 2020b). Par exemple, des travaux récents rendent le MPC différentiable et apprennent une politique qui adapte de façon continue les poids du MPC (Jahncke *et al.*, 2026). Réciproquement, dans les méthodes de *model-based RL*, l'environnement est remplacé par un modèle dynamique appris, souvent un réseau de neurones. L'analogie serait que l'agent s'entraîne dans un rêve qu'on appelle *World Model* (Ha & Schmidhuber, 2018). Cette approche a montré une meilleure généralisation sur le véhicule F1TENTH (Brunnbauer *et al.*, 2022). Des architectures récentes combinées à une optimisation de trajectoire dans un espace latent atteignent

des performances robustes sur un large éventail de tâches de commande continue (Hansen, Su & Wang, 2024). Une hybridation plus directe entre planification et apprentissage consiste à conditionner une politique apprise sur la trajectoire fournie par un planificateur, afin de conjuguer capacité de prédiction, respect explicite des contraintes et adaptation par apprentissage (Ghignone, Baumann & Magno, 2024). Les modèles génératifs, et en particulier les *Diffusion Policy*, offrent une manière d'exprimer des comportements multimodaux : un modèle de diffusion conditionné sur des mesures en ligne peut, par exemple, être intégré à une commande prédictive en temps réel pour piloter un véhicule à la limite de l'adhérence et s'adapter à différents pneus et conditions de route (Djeumou *et al.*, 2025).

Bien que ce mémoire ne traite que de la course autonome qui n'est qu'un petit pan de la robotique, les ramifications scientifiques et d'ingénierie ne sont pas négligeables. De fait, les prolongements explorables restent nombreux.

ANNEXE I

DÉMONSTRATIONS MATHÉMATIQUES

Théorème 0.1 (Inversibilité de la matrice du système de splines). *Soit $M \in \mathbb{R}^{4N \times 4N}$ la matrice définie à l'équation (3.13) pour $N \geq 2$ dont la structure est rappelée ci-dessous :*

$$M = \begin{bmatrix} A_{t,1} & A_{t,2} & & & \\ & A_{t,1} & A_{t,2} & & \\ & & \ddots & \ddots & \\ & & & A_{t,1} & A_{t,2} \\ A_{t,2} & & & & A_{t,1} \end{bmatrix} \in \mathbb{R}^{4N \times 4N}$$

$$\text{où : } A_{t,1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & 6 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \quad (\text{A I-1})$$

$$A_{t,2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -2 & 0 \end{bmatrix} \in \mathbb{R}^{4 \times 4}$$

Alors M est inversible.

Démonstration. Considérons la matrice $M \in \mathbb{R}^{4N \times 4N}$ définie à l'équation (3.13) pour $N \geq 2$, construite à partir des blocs $A_{t,1}$ et $A_{t,2}$ de l'équation (3.12).

Pour montrer que M est inversible, il suffit de montrer qu'elle est bijective. Or M est carrée donc, il suffit de montrer qu'elle est injective. C'est-à-dire que son noyau est réduit au vecteur nul : $\ker M = \{0_{\mathbb{R}^{4N}}\}$.

Soit $a = (a_{i,0}, a_{i,1}, a_{i,2}, a_{i,3})_{i=1}^N \in \ker M$. Par la structure périodique par blocs de M , les indices de segment sont interprétés modulo N .

$$\forall i \in \llbracket 1, N \rrbracket, a_{i+N, \bullet} = a_{i, \bullet} \quad (\text{A I-2})$$

Ainsi, les indices de segment évoluent dans le groupe cyclique $\mathbb{Z}/N\mathbb{Z}$.

La i -ième ligne de blocs de l'équation $Ma = 0$ s'écrit :

$$A_{t,1} \begin{bmatrix} a_{i,0} \\ a_{i,1} \\ a_{i,2} \\ a_{i,3} \end{bmatrix} + A_{t,2} \begin{bmatrix} a_{i+1,0} \\ a_{i+1,1} \\ a_{i+1,2} \\ a_{i+1,3} \end{bmatrix} = 0, \quad (\text{A I-3})$$

Ce qui correspond, par l'équation (3.10) prise avec un second membre nul, aux quatre relations scalaires suivantes, valables pour tout $i = 1, \dots, N$:

$$\begin{aligned} & \begin{cases} a_{i,0} & = 0 \\ a_{i,0} + a_{i,1} + a_{i,2} + a_{i,3} & = 0 \\ a_{i,1} + 2a_{i,2} + 3a_{i,3} - a_{i+1,1} & = 0 \\ 2a_{i,2} + 6a_{i,3} - 2a_{i+1,2} & = 0 \end{cases} \\ & \iff \begin{cases} a_{i,1} + a_{i,2} + a_{i,3} & = 0 \\ a_{i,1} + 2a_{i,2} + 3a_{i,3} - a_{i+1,1} & = 0 \\ 2a_{i,2} + 6a_{i,3} - 2a_{i+1,2} & = 0 \end{cases} \\ & \iff \begin{cases} a_{i,1} + a_{i,2} + a_{i,3} & = 0 \\ a_{i,1} + 2a_{i,2} + 3a_{i,3} & = a_{i+1,1} \\ 2a_{i,2} + 6a_{i,3} & = 2a_{i+1,2} \end{cases} \\ & \iff \begin{cases} a_{i,1} + a_{i,2} + a_{i,3} & = 0 \\ a_{i,1} + 2a_{i,2} + 3a_{i,3} & = a_{i+1,1} \\ a_{i,2} + 3a_{i,3} & = a_{i+1,2} \end{cases} \end{aligned} \quad (\text{A I-4})$$

Posons $u_i = a_{i,1}$, $v_i = a_{i,2}$ et $w_i = a_{i,3}$.

(A I-4) devient :

$$\begin{cases} u_i + v_i + w_i &= 0 \\ u_i + 2v_i + 3w_i &= u_{i+1} \\ v_i + 3w_i &= v_{i+1} \end{cases} \quad (\text{A I-5})$$

La première équation de (A I-5) donne

$$w_i = -u_i - v_i \quad (\text{A I-6})$$

En substituant (A I-6) dans la deuxième et la troisième équation de (A I-5), on obtient

$$\begin{aligned} &\begin{cases} u_{i+1} = u_i + 2v_i + 3(-u_i - v_i) \\ v_{i+1} = v_i + 3(-u_i - v_i) \end{cases} \\ &\Leftrightarrow \begin{cases} u_{i+1} = -2u_i - v_i \\ \textcolor{brown}{v}_{i+1} = -3u_i - 2v_i \end{cases} \end{aligned} \quad (\text{A I-7})$$

Ces deux relations permettent d'éliminer v_i au profit d'une récurrence portant uniquement sur (u_i) . En effet, en incrémentant de 1 les relations de (A I-7) :

$$\begin{cases} u_{i+2} &= -2u_{i+1} - v_{i+1} \\ v_{i+2} &= -3u_{i+1} - 2v_{i+1} \end{cases} \quad (\text{A I-8})$$

En substituant $\textcolor{brown}{v}_{i+1}$ de (A I-7) :

$$\begin{aligned} u_{i+2} &= -2u_{i+1} - (-3u_i - 2v_i) \\ u_{i+2} &= -2u_{i+1} + 3u_i + 2v_i \end{aligned} \quad (\text{A I-9})$$

(A I-7) donne :

$$\textcolor{blue}{v}_i = -2u_i - u_{i+1} \quad (\text{A I-10})$$

En substituant à nouveau :

$$u_{i+2} = -2u_{i+1} + 3u_i + 2(-2u_i - u_{i+1}) \quad (\text{A I-11})$$

$$u_{i+2} = -u_i - 4u_{i+1} \quad (\text{A I-12})$$

Ainsi, on a la récurrence d'ordre 2 :

$$\forall i \in \llbracket 1, N \rrbracket, u_i + 4u_{i+1} + u_{i+2} = 0 \quad (\text{A I-13})$$

Multiplions (A I-13) par u_{i+1} et sommons sur $i = 1, \dots, N$:

$$\sum_{i=1}^N u_{i+1}u_i + 4 \sum_{i=1}^N u_{i+1}^2 + \sum_{i=1}^N u_{i+1}u_{i+2} = 0 \quad (\text{A I-14})$$

Par l'équation (A I-2), toute somme sur $i = 1, \dots, N$ est invariante par décalage des indices :

$$\begin{aligned} \sum_i u_{i+1}^2 &= \sum_i u_i^2 \\ \sum_i u_{i+1}u_{i+2} &= \sum_i u_iu_{i+1} \end{aligned} \quad (\text{A I-15})$$

L'équation (A I-14) se réduit à :

$$\begin{aligned} 4 \sum_{i=1}^N u_i^2 + 2 \sum_{i=1}^N u_iu_{i+1} &= 0 \iff \\ 2 \sum_{i=1}^N u_i^2 + \sum_{i=1}^N u_iu_{i+1} &= 0 \end{aligned} \quad (\text{A I-16})$$

Considérons la somme :

$$\begin{aligned}
 \sum_{i=1}^N [(u_i + u_{i+1})^2 + 2u_i^2] &= \sum_{i=1}^N [(u_i^2 + 2u_i u_{i+1} + u_{i+1}^2) + 2u_i^2] \\
 \sum_{i=1}^N [(u_i + u_{i+1})^2 + 2u_i^2] &= \sum_{i=1}^N [3u_i^2 + 2u_i u_{i+1} + u_{i+1}^2] \\
 \sum_{i=1}^N [(u_i + u_{i+1})^2 + 2u_i^2] &= 3 \sum_{i=1}^N u_i^2 + 2 \sum_{i=1}^N u_i u_{i+1} + \sum_{i=1}^N u_{i+1}^2 \\
 \sum_{i=1}^N [(u_i + u_{i+1})^2 + 2u_i^2] &= 4 \sum_{i=1}^N u_i^2 + 2 \sum_{i=1}^N u_i u_{i+1} \quad \text{par (A I-15)}
 \end{aligned} \tag{A I-17}$$

Par (A I-16) :

$$\begin{aligned}
 \sum_{i=1}^N [(u_i + u_{i+1})^2 + 2u_i^2] &= 0 \quad \text{où} \quad \forall i \in \llbracket 1, N \rrbracket, (u_i + u_{i+1})^2 + 2u_i^2 \geq 0 \\
 \implies \forall i \in \llbracket 1, N \rrbracket, (u_i + u_{i+1})^2 + 2u_i^2 &= 0 \quad \text{où} \quad \forall i \in \llbracket 1, N \rrbracket, \begin{cases} (u_i + u_{i+1})^2 \geq 0 \\ 2u_i^2 \geq 0 \end{cases} \\
 \implies \forall i \in \llbracket 1, N \rrbracket, \begin{cases} u_i^2 = 0 \\ (u_i + u_{i+1})^2 = 0 \end{cases} \\
 \iff \forall i \in \llbracket 1, N \rrbracket, \begin{cases} u_i = 0 \\ u_i + u_{i+1} = 0 \end{cases}
 \end{aligned} \tag{A I-18}$$

$$\begin{aligned}
& \begin{cases} v_i = -2u_i - u_{i+1} & \text{par (A I-10)} \\ w_i = -u_i - v_i & \text{par (A I-6)} \end{cases} \\
\Longleftrightarrow & \begin{cases} v_i = -2 \times 0 - 0 \\ w_i = -0 - v_i \end{cases} \\
\Longleftrightarrow & \begin{cases} v_i = 0 \\ w_i = -v_i \end{cases} \\
\Longleftrightarrow & \begin{cases} v_i = 0 \\ w_i = 0 \end{cases}
\end{aligned} \tag{A I-19}$$

En regroupant les résultats obtenus :

$$\forall i \in \llbracket 1, N \rrbracket, \begin{cases} a_{i,0} = 0 & \text{par (A I-4)} \\ u_i = a_{i,1} = 0 & \text{par (A I-18)} \\ v_i = a_{i,2} = 0 & \text{par (A I-19)} \\ w_i = a_{i,3} = 0 & \text{par (A I-19)} \end{cases} \tag{A I-20}$$

Alors $a = 0_{\mathbb{R}^{4N}}$

$$\forall a \in \mathbb{R}^{4N}, Ma = 0 \implies a = 0_{\mathbb{R}^{4N}} \tag{A I-21}$$

$\ker M = \{0_{\mathbb{R}^{4N}}\}$ donc la matrice M est injective. Or M est carrée donc bijective.

$\forall N \in \mathbb{N}, N \geq 2 \implies M \text{ est inversible.}$

Lemme 0.2 (Forme quadratique positive de l'hessien). Soit $H \in \mathbb{R}^{N \times N}$ l'hessien du programme quadratique à courbure minimale, défini à l'équation (3.21) par :

$$H = T_{nx}^\top P_{xx} T_{nx} + T_{nx}^\top P_{xy} T_{ny} + T_{ny}^\top P_{yy} T_{ny} \quad (\text{A I-22})$$

où P_{xx} , P_{xy} et P_{yy} sont les matrices de l'équation (3.19) et T_{nx}, T_{ny} sont les matrices de l'équation (3.21).

Supposons que la ligne de référence soit une courbe régulière :

$$\forall i \in \llbracket 1, N \rrbracket, x_i'^2 + y_i'^2 > 0 \quad (\text{A I-23})$$

Alors, la forme quadratique associée à H est positive i.e.,

$$\forall \alpha \in \mathbb{R}^N, \alpha^\top H \alpha \geq 0 \quad (\text{A I-24})$$

Démonstration. Pour tout $i \in \llbracket 1, N \rrbracket$, notons $d_i = (x_i'^2 + y_i'^2)^3 > 0$ le dénominateur commun de l'équation (3.19), de sorte que les entrées diagonales des trois matrices s'écrivent

$$(P_{xx})_{ii} = \frac{y_i'^2}{d_i}, \quad (P_{xy})_{ii} = \frac{-2x_i'y_i'}{d_i}, \quad (P_{yy})_{ii} = \frac{x_i'^2}{d_i}. \quad (\text{A I-25})$$

(A I-23) garantit $d_i > 0$, ce qui donne un sens aux quotients ci-dessus.

Soit $\alpha \in \mathbb{R}^N$. Posons $u = T_{nx}\alpha$ et $v = T_{ny}\alpha$ dans $\mathbb{R}^N$. La forme quadratique de H s'écrit :

$$\alpha^\top H \alpha = u^\top P_{xx} u + u^\top P_{xy} v + v^\top P_{yy} v \quad (\text{A I-26})$$

Par l'équation (3.16), u et v sont les parties linéaires en α des dérivées secondes x'' et y'' . Les matrices P_{xx} , P_{xy} et P_{yy} étant diagonales, l'équation (A I-26) se scinde en une somme sur les

points :

$$\begin{aligned}
 \alpha^\top H \alpha &= \sum_{i=1}^N \left(\frac{y_i'^2}{d_i} u_i^2 - \frac{2x_i' y_i'}{d_i} u_i v_i + \frac{x_i'^2}{d_i} v_i^2 \right) \\
 \alpha^\top H \alpha &= \sum_{i=1}^N \frac{1}{d_i} \left(y_i' u_i^2 - 2y_i' x_i' v_i + x_i' v_i^2 \right) \quad (\text{A I-27}) \\
 \alpha^\top H \alpha &= \sum_{i=1}^N \frac{1}{d_i} (y_i' u_i - x_i' v_i)^2 \quad \text{car} \quad \forall (a, b) \in \mathbb{R}^2, (a - b)^2 = a^2 - 2ab + b^2
 \end{aligned}$$

Puis pour chaque terme de la somme (A I-27), on a :

$$\begin{aligned}
 \forall i \in \llbracket 1, N \rrbracket, \quad & \begin{cases} \frac{1}{d_i} > 0 \\ (y_i' u_i - x_i' v_i)^2 \geq 0 \end{cases} \\
 \implies \forall i \in \llbracket 1, N \rrbracket, \quad & \frac{1}{d_i} (y_i' u_i - x_i' v_i)^2 \geq 0
 \end{aligned} \quad (\text{A I-28})$$

Une somme finie de termes positifs ou nuls est positive ou nulle, donc par (A I-27) et (A I-28) :

$\forall \alpha \in \mathbb{R}^N, \alpha^\top H \alpha \geq 0$, la forme quadratique associée à H est positive.

Théorème 0.3 (L'hessien symétrisé est semi-défini positif). *Soit $H \in \mathbb{R}^{N \times N}$ l'hessien du programme quadratique à courbure minimale, défini à l'équation (3.21) par :*

$$H = T_{nx}^\top P_{xx} T_{nx} + T_{nx}^\top P_{xy} T_{ny} + T_{ny}^\top P_{yy} T_{ny} \quad (\text{A I-29})$$

où P_{xx} , P_{xy} et P_{yy} sont les matrices de l'équation (3.19) et T_{nx}, T_{ny} sont les matrices de l'équation (3.21).

Supposons que la ligne de référence soit une courbe régulière :

$$\forall i \in \llbracket 1, N \rrbracket, x_i'^2 + y_i'^2 > 0 \quad (\text{A I-30})$$

Alors, l'hessien symétrisé $\frac{1}{2}(H + H^\top) \in \mathbb{S}_N$ est semi-défini positif, i.e.,

$$\forall \alpha \in \mathbb{R}^N, \alpha^\top \frac{1}{2}(H + H^\top) \alpha \geq 0 \quad (\text{A I-31})$$

Démonstration. Introduisons les matrices :

$$G = \begin{pmatrix} y'_1 & & \\ & \ddots & \\ & & y'_N \end{pmatrix} T_{nx} - \begin{pmatrix} x'_1 & & \\ & \ddots & \\ & & x'_N \end{pmatrix} T_{ny} \in \mathbb{R}^{N \times N}$$

$$D = \begin{pmatrix} \frac{1}{d_1} & & \\ & \ddots & \\ & & \frac{1}{d_N} \end{pmatrix} \quad (\text{A I-32})$$

Notons que :

$$G\alpha = \begin{pmatrix} y'_1 & & \\ & \ddots & \\ & & y'_N \end{pmatrix} T_{nx}\alpha - \begin{pmatrix} x'_1 & & \\ & \ddots & \\ & & x'_N \end{pmatrix} T_{ny}\alpha$$

$$G\alpha = \begin{pmatrix} y'_1 & & \\ & \ddots & \\ & & y'_N \end{pmatrix} u - \begin{pmatrix} x'_1 & & \\ & \ddots & \\ & & x'_N \end{pmatrix} v \in \mathbb{R}^N \quad (\text{A I-33})$$

$$\forall i \in \llbracket 1, N \rrbracket, (G\alpha)_i = y'_i u_i - x'_i v_i$$

Et :

$$(G^\top D G)^\top = (D G)^\top (G^\top)^\top$$

$$(G^\top D G)^\top = G^\top D^\top G$$

$$(G^\top D G)^\top = G^\top D G \in \mathbb{S}_N$$

$$\implies \frac{1}{2}(G^\top D G + (G^\top D G)^\top) = G^\top D G \quad (\text{A I-34})$$

En partant de (A I-27) :

$$\begin{aligned}
 \alpha^\top H \alpha &= \sum_{i=1}^N \frac{1}{d_i} (y'_i u_i - x'_i v_i)^2 \\
 \alpha^\top H \alpha &= \sum_{i=1}^N \frac{1}{d_i} ((G\alpha)_i)^2 \\
 \alpha^\top H \alpha &= \sum_{i=1}^N (G\alpha)_i \frac{1}{d_i} (G\alpha)_i \\
 \alpha^\top H \alpha &= (G\alpha)^\top D (G\alpha) \\
 \alpha^\top H \alpha &= \alpha^\top G^\top D G \alpha
 \end{aligned} \tag{A I-35}$$

Or $\forall n \in \mathbb{N}^*, \forall A, B \in \mathbb{R}^{n \times n}, \forall \alpha \in \mathbb{R}^n, \alpha^\top A \alpha = \alpha^\top B \alpha \iff \frac{1}{2}(A + A^\top) = \frac{1}{2}(B + B^\top)$ et par (A I-34) :

$$\frac{1}{2}(H + H^\top) = G^\top D G \tag{A I-36}$$

Or par le lemme 0.2 : $\forall \alpha \in \mathbb{R}^N, \alpha^\top H \alpha \geq 0$.

Puis par (A I-35) : $\forall \alpha \in \mathbb{R}^N, \alpha^\top G^\top D G \alpha \geq 0$.

Par (A I-34) :

$\forall \alpha \in \mathbb{R}^N, \alpha^\top \frac{1}{2}(H + H^\top) \alpha \geq 0$, l'hessien symétrisé est semi-défini positif.

Théorème 0.4 (Théorème de Borel-Lebesgue ou Heine-Borel dans $\mathbb{R}^N$). *Un sous-ensemble $A \subseteq \mathbb{R}^N$ est compact si et seulement si A est fermé et borné.*

Une démonstration est donnée dans Bourbaki (2007).

Théorème 0.5 (Théorème de Weierstrass). *Soit f une fonction d'un espace compact A dans $\mathbb{R}$. Si f est continue, alors f est bornée et atteint ses bornes.*

Une démonstration est donnée dans Sebestik (1992).

Théorème 0.6 (Existence d'un minimiseur global du programme quadratique). *Soit le programme quadratique à courbure minimale (3.26), de fonction objectif $J(\alpha) = \frac{1}{2} \alpha^\top H \alpha + f^\top \alpha$ et d'ensemble admissible*

$$\mathcal{F} = \left\{ \alpha \in \mathbb{R}^N \left| \begin{cases} l \leq \alpha \leq r \\ E\alpha \preceq \kappa_{\text{limite}} \end{cases} \right. \right\} \quad (\text{A I-37})$$

où les contraintes de l'équation (3.23) s'écrivent $l_i = -(w_i^{\text{gauche}} - \frac{W_{\text{veh}}}{2})$ et $r_i = w_i^{\text{droite}} - \frac{W_{\text{veh}}}{2}$. On suppose que :

- le véhicule tient sur la piste : $\forall i \in \llbracket 1, N \rrbracket, \begin{cases} w_i^{\text{gauche}} \geq \frac{W_{\text{veh}}}{2} \\ w_i^{\text{droite}} \geq \frac{W_{\text{veh}}}{2} \end{cases} \iff \forall i \in \llbracket 1, N \rrbracket, l_i \leq 0 \leq r_i$
- la courbure de la ligne de référence respecte la borne maximale : $\forall i \in \llbracket 1, N \rrbracket, |\kappa_{\text{ref},i}| \leq \kappa_{\text{max}}$

Alors :

- $\mathcal{F}$ est non vide et compact
- J atteint un minimum global sur $\mathcal{F}$

Démonstration. Écrivons $\mathcal{F} = \mathcal{B} \cap \mathcal{C}$, où :

- $\mathcal{B} = \{\alpha \in \mathbb{R}^N \mid l \leq \alpha \leq r\}$
- $\mathcal{C} = \bigcap_{j=1}^N \{\alpha \in \mathbb{R}^N \mid (E\alpha)_j \leq (\kappa_{\text{limite}})_j\}$

Soit $j \in \llbracket 1, N \rrbracket$. Considérons :

$$f_j : \begin{cases} \mathbb{R}^N \rightarrow \mathbb{R} \\ \alpha \mapsto (E\alpha)_j \end{cases} \quad (\text{A I-38})$$

$$C_j = \{\alpha \in \mathbb{R}^N \mid f_j(\alpha) \leq (\kappa_{\text{limite}})_j\}$$

Soit $\alpha, \beta \in \mathbb{R}^N$ et $\lambda \in \mathbb{R}$.

$$\begin{aligned} f_j(\alpha + \lambda\beta) &= (E(\alpha + \lambda\beta))_j \\ f_j(\alpha + \lambda\beta) &= \sum_{k=1}^N E_{jk} (\alpha_k + \lambda\beta_k) \\ f_j(\alpha + \lambda\beta) &= \sum_{k=1}^N E_{jk} \alpha_k + E_{jk} \lambda\beta_k \\ f_j(\alpha + \lambda\beta) &= \sum_{k=1}^N E_{jk} \alpha_k + \lambda \sum_{k=1}^N E_{jk} \beta_k \\ f_j(\alpha + \lambda\beta) &= f_j(\alpha) + \lambda f_j(\beta) \end{aligned} \quad (\text{A I-39})$$

L'application f_j est linéaire.

Notons $\|\cdot\|$ une norme sur $\mathbb{R}^N$ et $e_j^\top = \begin{bmatrix} E_{j1} & \dots & E_{jN} \end{bmatrix}^\top \in \mathbb{R}^N$ la j -ème ligne de E . Alors $f_j(\alpha) = e_j \cdot \alpha$. Considérons :

$$\begin{aligned} |f_j(\alpha) - f_j(\beta)| &= |f_j(\alpha - \beta)| \quad \text{car } f_j \text{ est linéaire} \\ |f_j(\alpha) - f_j(\beta)| &= |e_j \cdot (\alpha - \beta)| \\ |f_j(\alpha) - f_j(\beta)| &= |\langle e_j, \alpha - \beta \rangle| \\ |f_j(\alpha) - f_j(\beta)| &\leq \|e_j\| \|\alpha - \beta\| \quad \text{par inégalité de Cauchy-Schwarz} \end{aligned} \quad (\text{A I-40})$$

Ainsi f_j est $\|e_j\|$ -lipschitzienne donc continue.

Posons $c_j = (\kappa_{\text{limite}})_j$.

$$\begin{aligned}
 \alpha &\in C_j \\
 \iff (E\alpha)_j &\leq c_j \\
 \iff f_j(\alpha) &\leq c_j \\
 \iff f_j(\alpha) &\in (-\infty, c_j] \\
 \iff \alpha &\in f_j^{-1}((-\infty, c_j])
 \end{aligned} \tag{A I-41}$$

Donc C_j est l'image réciproque de $(-\infty, c_j]$ par f_j . Or $(-\infty, c_j]$ est fermé et f_j est continue, donc C_j est fermé.

Puis, une intersection finie de fermés est fermée : $\bigcap_{j=1}^N C_j = C$ est fermé.

Supposons $\alpha \in \mathcal{B}$. Alors :

$$\begin{aligned}
 \forall i \in \llbracket 1, N \rrbracket, \alpha_i &\in [l_i, u_i] \\
 \forall i \in \llbracket 1, N \rrbracket, |\alpha_i| &\leq \max(|l_i|, |u_i|)
 \end{aligned} \tag{A I-42}$$

Soit $(\varepsilon_1, \dots, \varepsilon_N)$ la base canonique de $\mathbb{R}^N$. Alors $\alpha = \sum_{i=1}^N \alpha_i \varepsilon_i$.

$$\begin{aligned}
 \|\alpha\| &= \left\| \sum_{i=1}^N \alpha_i \varepsilon_i \right\| \\
 \|\alpha\| &\leq \sum_{i=1}^N \|\alpha_i \varepsilon_i\| \\
 \|\alpha\| &\leq \sum_{i=1}^N |\alpha_i| \|\varepsilon_i\| \\
 \|\alpha\| &\leq \sum_{i=1}^N \max(|l_i|, |u_i|) \|\varepsilon_i\|
 \end{aligned} \tag{A I-43}$$

Donc $\mathcal{B}$ est borné.

$\mathcal{B}$ peut se récrire comme le produit cartésien des segments $[l_i, u_i]$: $\mathcal{B} = \prod_{i=1}^N [l_i, u_i]$. Or

- le produit cartésien de fermés est fermé
- $\forall i \in \llbracket 1, N \rrbracket$, $[l_i, u_i]$ est fermé

Donc $\mathcal{B}$ est fermé.

Ainsi $\mathcal{B}$ est fermé et borné, donc par le théorème Borel-Lebesgue (0.4) $\mathcal{B}$ est compact.

Comme $\mathcal{F} = \mathcal{B} \cap C$ est l'intersection d'un compact et d'un fermé :

$\mathcal{F}$ est compact.

La première hypothèse donne :

$$\begin{aligned}
 & \forall i \in \llbracket 1, N \rrbracket, l_i \leq 0 \leq u_i \\
 & \Longleftrightarrow l \preccurlyeq \mathbf{0} \preccurlyeq r \\
 & \Longleftrightarrow \mathbf{0} \in \mathcal{B}
 \end{aligned} \tag{A I-44}$$

Par définition $\kappa_{\text{limite}} = \begin{bmatrix} (\kappa_{\max} - \kappa_{\text{ref}})\mathbf{1}_N \\ (\kappa_{\max} + \kappa_{\text{ref}})\mathbf{1}_N \end{bmatrix} \in \mathbb{R}^{2N}$

La seconde hypothèse donne :

$$\begin{aligned}
 & \forall i \in \llbracket 1, N \rrbracket, |\kappa_{\text{ref},i}| \leq \kappa_{\max} \\
 & \Rightarrow \forall i \in \llbracket 1, N \rrbracket, \begin{cases} \kappa_{\max} \geq 0 \\ \kappa_{\max} \geq \kappa_{\text{ref},i} \end{cases} \\
 & \Rightarrow \forall i \in \llbracket 1, N \rrbracket, \begin{cases} \kappa_{\max} \geq 0 \\ \kappa_{\max} - \kappa_{\text{ref},i} \geq 0 \\ \kappa_{\max} + \kappa_{\text{ref},i} \geq 0 \end{cases}
 \end{aligned} \tag{A I-45}$$

Donc :

$$\begin{aligned}
 & \forall i \in \llbracket 1, N \rrbracket, \begin{cases} \kappa_{\text{limite},i} = (\kappa_{\max} - \kappa_{\text{ref},i}) \\ \kappa_{\text{limite},N+i} = (\kappa_{\max} + \kappa_{\text{ref},i}) \end{cases} \\
 & \Longleftrightarrow \forall i \in \llbracket 1, N \rrbracket, \begin{cases} \kappa_{\text{limite},i} \geq 0 \\ \kappa_{\text{limite},N+i} \geq 0 \end{cases} \\
 & \Longleftrightarrow \mathbf{0} \preccurlyeq \kappa_{\text{limite}}
 \end{aligned} \tag{A I-46}$$

Or $E\mathbf{0} = \mathbf{0}$, donc $\mathbf{0} \in C$.

Ainsi $\mathbf{0} \in \mathcal{B} \cap C = \mathcal{F}$.

La ligne de référence $\alpha_{\text{ref}} = \mathbf{0} \in \mathbb{R}^N$ appartient à $\mathcal{F}$:

$$\{\mathbf{0}\} \subseteq \mathcal{F} \neq \emptyset, \mathcal{F} \text{ est non vide.}$$

On a montré que $\mathcal{F}$ est non vide et compact.

La fonction objectif J est un polynôme des coordonnées de α , donc continue.

Supposons $\alpha \in \mathcal{F}$. En développant le produit matriciel dans J :

$$\begin{aligned} J(\alpha) &= \frac{1}{2} \alpha^\top H \alpha + f^\top \alpha \\ J(\alpha) &= \frac{1}{2} \alpha^\top \begin{bmatrix} \sum_{j=1}^N H_{1j} \alpha_j \\ \vdots \\ \sum_{j=1}^N H_{Nj} \alpha_j \end{bmatrix} + \sum_{i=1}^N f_i \alpha_i \\ J(\alpha) &= \frac{1}{2} \sum_{i=1}^N \alpha_i \sum_{j=1}^N H_{ij} \alpha_j + \sum_{i=1}^N f_i \alpha_i \\ J(\alpha) &= \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N H_{ij} \alpha_i \alpha_j + \sum_{i=1}^N f_i \alpha_i \\ J(\alpha) &= \frac{1}{2} \sum_{i=1}^N H_{ii} \alpha_i^2 + \sum_{1 \leq i < j \leq N} H_{ij} \alpha_i \alpha_j + \sum_{i=1}^N f_i \alpha_i \quad \text{car } H \in \mathbb{S}_N^+ \end{aligned} \tag{A I-47}$$

Chaque terme est un monôme en les coordonnées $\alpha_1, \dots, \alpha_N$ de degré 2 ou 1.

J est un polynôme en plusieurs indéterminées continu sur $\mathcal{F}$.

Or $\mathcal{F}$ est compact non vide.

Donc par le théorème de Weierstrass (0.5) :

$$\exists \alpha^\star \in \mathcal{F}, \forall \alpha \in \mathcal{F}, J(\alpha^\star) \leq J(\alpha) \tag{A I-48}$$

$$J \text{ atteint un minimum global sur } \mathcal{F}$$

Notons qu'a priori un tel $\alpha^\star$ n'est pas unique.

Corollaire 0.6.1 (Caractérisation d'un minimum global du programme quadratique). *Soit le programme quadratique à courbure minimale (3.26), de fonction objectif $J(\alpha) = \frac{1}{2} \alpha^\top H \alpha + f^\top \alpha$ et d'ensemble admissible*

$$\mathcal{F} = \left\{ \alpha \in \mathbb{R}^N \left| \begin{cases} l \leq \alpha \leq r \\ E\alpha \preceq \kappa_{\text{limite}} \end{cases} \right. \right\} \quad (\text{A I-49})$$

Les conditions de Karush-Kuhn-Tucker (KKT) associées sont :

$$H\alpha^\star + f + E^\top \mu^\star - \lambda^\star + \nu^\star = \mathbf{0} \quad (\text{A I-50})$$

$$\begin{cases} E\alpha^\star \preceq \kappa_{\text{limite}} \\ l \preceq \alpha^\star \preceq r \end{cases} \quad (\text{A I-51})$$

$$\mu^\star \succeq \mathbf{0}, \quad \lambda^\star \succeq \mathbf{0}, \quad \nu^\star \succeq \mathbf{0} \quad (\text{A I-52})$$

$$\begin{cases} \mu_j^\star ((E\alpha^\star)_j - (\kappa_{\text{limite}})_j) = 0, & j = 1, \dots, 2N \\ \lambda_i^\star (l_i - \alpha_i^\star) = 0, & i = 1, \dots, N \\ \nu_i^\star (\alpha_i^\star - r_i) = 0, & i = 1, \dots, N \end{cases} \quad (\text{A I-53})$$

Avec le lagrangien :

$$\mathcal{L}(\alpha, \mu, \lambda, \nu) = \frac{1}{2} \alpha^\top H \alpha + f^\top \alpha + \mu^\top (E\alpha - \kappa_{\text{limite}}) + \lambda^\top (l - \alpha) + \nu^\top (\alpha - r) \quad (\text{A I-54})$$

où $\mu \in \mathbb{R}^{2N}$ et $\lambda, \nu \in \mathbb{R}^N$.

Alors, il faut et il suffit que $\alpha^\star \in \mathcal{F}$ vérifie les conditions de Karush-Kuhn-Tucker pour être un minimum global.

Démonstration. Par le théorème 0.3, $H \in \mathbb{S}_N^+$ donc J est convexe.

Les contraintes du programme quadratique sont linéaires.

Donc le programme quadratique est convexe.

Puis selon Boyd & Vandenberghe (2004) :

- un minimum local d'un programme convexe est un minimum global,
- toute solution des conditions KKT est un optimum global.

Le minimiseur α^* de l'équation (A I-48) est ainsi un optimum global certifié.

BIBLIOGRAPHIE

- Alcalá, E., Puig, V., Quevedo, J. & Rosolia, U. (2020). Autonomous racing using Linear Parameter Varying-Model Predictive Control (LPV-MPC). *Control Engineering Practice*, 95, 104270. doi : 10.1016/j.conengprac.2019.104270.
- Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B. & Diehl, M. (2019). CasADi : a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1), 1–36. doi : 10.1007/s12532-018-0139-4.
- Babu, V. S. & Behl, M. (2020). fltenth.dev - An Open-source ROS based F1/10 Autonomous Racing Simulator. *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pp. 1614–1620. doi : 10.1109/CASE48305.2020.9216949.
- Bakker, E., Pacejka, H. B. & Lidner, L. (1989). A New Tire Model with an Application in Vehicle Dynamics Studies. pp. 890087. doi : 10.4271/890087.
- Balaji, B., Mallya, S., Genc, S., Gupta, S., Dirac, L., Khare, V., Roy, G., Sun, T., Tao, Y., Townsend, B., Calleja, E., Muralidhara, S. & Karuppasamy, D. (2020). DeepRacer : Autonomous Racing Platform for Experimentation with Sim2Real Reinforcement Learning. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2746–2754. doi : 10.1109/ICRA40945.2020.9197465.
- Betz, J., Zheng, H., Liniger, A., Rosolia, U., Karle, P., Behl, M., Krovi, V. & Mangharam, R. (2022). Autonomous Vehicles on the Edge : A Survey on Autonomous Vehicle Racing. *IEEE Open Journal of Intelligent Transportation Systems*, 3, 458–488. doi : 10.1109/ojits.2022.3181510. arXiv :2202.07008 [cs].
- Bishop, C. M. & Bishop, H. (2024). *Deep learning : foundations and concepts*. Cham, Switzerland : Springer. doi : 10.1007/978-3-031-45468-4.
- Borrelli, F., Bemporad, A. & Morari, M. (2017). *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press. doi : 10.1017/9781139061759.
- Bourbaki, N. (2007). *Topologie générale*. Berlin, Heidelberg : Springer Berlin Heidelberg. doi : 10.1007/978-3-540-33982-3.
- Boyd, S. & Vandenberghe, L. (2004). *Convex Optimization* (éd. 1). Cambridge University Press. doi : 10.1017/CBO9780511804441.
- Brunnbauer, A., Berducci, L., Brandstatter, A., Lechner, M., Hasani, R., Rus, D. & Grosu, R. (2022). Latent Imagination Facilitates Zero-Shot Transfer in Autonomous Racing. *2022 International Conference on Robotics and Automation (ICRA)*, pp. 7513–7520. doi : 10.1109/ICRA46639.2022.9811650.

- Cardamone, L., Loiacono, D., Lanzi, P. L. & Bardelli, A. P. (2010). Searching for the optimal racing line using genetic algorithms. *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pp. 388–394. doi : 10.1109/ITW.2010.5593330.
- Cataffo, V., Silano, G., Iannelli, L., Puig, V. & Glielmo, L. (2022). A Nonlinear Model Predictive Control Strategy for Autonomous Racing of Scale Vehicles. *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 100–105. doi : 10.1109/SMC53654.2022.9945279.
- Chisari, E., Liniger, A., Rupenyan, A., Gool, L. V. & Lygeros, J. [arXiv :2011.13332 [cs] version : 2]. (2021). Learning from Simulation, Racing in Reality. arXiv. Repéré le 2026-02-08 à <http://arxiv.org/abs/2011.13332>.
- Chrosniak, J., Ning, J. & Behl, M. (2024). Deep Dynamics : Vehicle Dynamics Modeling With a Physics-Constrained Neural Network for Autonomous Racing. *IEEE Robotics and Automation Letters*, 9(6), 5292–5297. doi : 10.1109/LRA.2024.3388847.
- Djeumou, F., Lew, T. J., Ding, N., Thompson, M., Suminaka, M., Greiff, M. & Subosits, J. (2025). One Model to Drift Them All : Physics-Informed Conditional Diffusion Model for Driving at the Limits. *Proceedings of The 8th Conference on Robot Learning*, pp. 604–630. Repéré à <https://proceedings.mlr.press/v270/djeumou25a.html>.
- Espié, E., Guionneau, C., Wymann, B., Dimitrakakis, C., Coulom, R. & Sumner, A. (2005). TORCS, The Open Racing Car Simulator. Repéré à <https://www.semanticscholar.org/paper/TORCS%2C-The-Open-Racing-Car-Simulator-Espi%C3%A9-Guionneau/b9c4d931665ec87c16fcd44cae8fdaec1215e81e#citing-papers>.
- Evans, B. D., Trumpp, R., Caccamo, M., Jahncke, F., Betz, J., Jordaan, H. W. & Engelbrecht, H. A. [arXiv :2402.18558 [cs]]. (2024). Unifying F1TENTH Autonomous Racing : Survey, Methods and Benchmarks. arXiv. Repéré le 2026-03-22 à <http://arxiv.org/abs/2402.18558>.
- Fuchs, F., Song, Y., Kaufmann, E., Scaramuzza, D. & Duerr, P. (2021). Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning. *IEEE Robotics and Automation Letters*, 6(3), 4257–4264. doi : 10.1109/LRA.2021.3064284. arXiv :2008.07971 [cs].
- Ghignone, E., Baumann, N. & Magno, M. (2024). TC-Driver : A Trajectory-Conditioned Reinforcement Learning Approach to Zero-Shot Autonomous Racing. *IEEE Transactions on Field Robotics*, 1, 527–536. doi : 10.1109/TFR.2024.3499912.

- Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep learning*. MIT Press. Repéré à <https://www.deeplearningbook.org/>.
- Guckiran, K. & Bolat, B. (2019). Autonomous Car Racing in Simulation Environment Using Deep Reinforcement Learning. *2019 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pp. 1–6. doi : 10.1109/ASYU48272.2019.8946332.
- Guiggiani, M. (2023). *The Science of Vehicle Dynamics : Handling, Braking, and Ride of Road and Race Cars*. Cham : Springer International Publishing. doi : 10.1007/978-3-031-06461-6.
- Ha, D. & Schmidhuber, J. (2018). World Models. doi : 10.5281/ZENODO.1207631.
- Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. (2018a). Soft actor-critic : Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *Proceedings of the 35th international conference on machine learning*, 80(Proceedings of machine learning research), 1861–1870. Repéré à <https://proceedings.mlr.press/v80/haarnoja18b.html>.
- Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P. & Levine, S. (2018b). Soft Actor-Critic Algorithms and Applications. Repéré le 2026-02-09 à <https://arxiv.org/abs/1812.05905v2>.
- Hairer, E. & Wanner, G. (2015). Runge–Kutta Methods, Explicit, Implicit. Dans Engquist, B. (Éd.), *Encyclopedia of Applied and Computational Mathematics* (pp. 1282–1285). Berlin, Heidelberg : Springer Berlin Heidelberg. doi : 10.1007/978-3-540-70529-1_144.
- Hansen, N., Su, H. & Wang, X. [arXiv :2310.16828 [cs]]. (2024). TD-MPC2 : Scalable, Robust World Models for Continuous Control. arXiv. Repéré le 2026-04-13 à <http://arxiv.org/abs/2310.16828>.
- Heilmeyer, A., Wischnewski, A., Hermansdorfer, L., Betz, J., Lienkamp, M. & Lohmann, B. (2020). Minimum curvature trajectory planning and control for an autonomous race car. *Vehicle System Dynamics*, 58(10), 1497–1527. doi : 10.1080/00423114.2019.1631455. _eprint : <https://doi.org/10.1080/00423114.2019.1631455>.
- Hewing, L., Kabzan, J. & Zeilinger, M. N. (2020a). Cautious Model Predictive Control Using Gaussian Process Regression. *IEEE Transactions on Control Systems Technology*, 28(6), 2736–2743. doi : 10.1109/TCST.2019.2949757.
- Hewing, L., Wabersich, K. P., Menner, M. & Zeilinger, M. N. (2020b). Learning-Based Model Predictive Control : Toward Safe Learning in Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(Volume 3, 2020), 269–296. doi : 10.1146/annurev-control-090419-075625.

- Hornik, K., Stinchcombe, M. & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359–366. doi : 10.1016/0893-6080(89)90020-8.
- Houska, B., Ferreau, H. & Diehl, M. (2011a). ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3), 298–312.
- Houska, B., Ferreau, H. & Diehl, M. (2011b). An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10), 2279–2285. doi : 10.1016/j.automatica.2011.08.020.
- Huang, S., Dossa, R. F. J., Ye, C., Braga, J., Chakraborty, D., Mehta, K. & Araújo, J. a. G. (2022). CleanRL : High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274), 1–18. Repéré à <http://jmlr.org/papers/v23/21-1342.html>.
- Jahncke, F., Zarrouki, B., Piccinini, M., D'sa, J., Isele, D., Bae, S. & Betz, J. (2026). Differentiable Weights-Varying Nonlinear MPC via Gradient-Based Policy Learning : An Autonomous Vehicle Guidance Example. *IEEE Robotics and Automation Letters*, 11(3), 3724–3731. doi : 10.1109/LRA.2026.3662644.
- Kabzan, J., Hewing, L., Liniger, A. & Zeilinger, M. N. (2019). Learning-Based Model Predictive Control for Autonomous Racing. *IEEE Robotics and Automation Letters*, 4(4), 3363–3370. doi : 10.1109/LRA.2019.2926677. Conference Name : IEEE Robotics and Automation Letters.
- Kapania, N. R. & Gerdes, J. C. (2015). Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling. *Vehicle System Dynamics*, 53(12), 1687–1704. doi : 10.1080/00423114.2015.1055279.
- Kapania, N. R., Subosits, J. & Christian Gerdes, J. (2016). A Sequential Two-Step Algorithm for Fast Generation of Vehicle Racing Trajectories. *Journal of Dynamic Systems, Measurement, and Control*, 138(9), 091005. doi : 10.1115/1.4033311.
- Kloeser, D., Schoels, T., Sartor, T., Zanelli, A., Prison, G. & Diehl, M. (2020). NMPC for Racing Using a Singularity-Free Path-Parametric Model with Obstacle Avoidance. *IFAC-PapersOnLine*, 53(2), 14324–14329. doi : 10.1016/j.ifacol.2020.12.1376.
- Koirala, P. & Fleming, C. [arXiv :2408.04198 [cs]]. (2024). F1tenth Autonomous Racing With Offline Reinforcement Learning Methods. arXiv. Repéré le 2026-03-22 à <http://arxiv.org/abs/2408.04198>.

- Kong, J., Pfeiffer, M., Schildbach, G. & Borrelli, F. (2015). Kinematic and dynamic vehicle models for autonomous driving control design. *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1094–1099. doi : 10.1109/IVS.2015.7225830.
- Liniger, A., Domahidi, A. & Morari, M. (2015). Optimization-based autonomous racing of 1 :43 scale RC cars. *Optimal Control Applications and Methods*, 36(5), 628–647. doi : 10.1002/oca.2123.
- Milliken, W. F. & Milliken, D. L. (1995). *Race car vehicle dynamics*. Warrendale, PA : Society of Automotive Engineers Internat.
- Novi, T., Liniger, A., Capitani, R. & Annicchiarico, C. (2020). Real-time control for at-limit handling driving on a predefined path. *Vehicle System Dynamics*, 58(7), 1007–1036. doi : 10.1080/00423114.2019.1605081.
- O’Kelly, M., Zheng, H., Karthik, D. & Mangharam, R. (2020). F1TENTH : An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning. *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, pp. 77–89. Repéré à <https://proceedings.mlr.press/v123/o-kelly20a.html>.
- Pacejka, H. B. & Bakker, E. (1992). THE MAGIC FORMULA TYRE MODEL. *Vehicle System Dynamics*, 21(sup001), 1–18. doi : 10.1080/00423119208969994.
- Perantoni, G. & Limebeer, D. J. (2014). Optimal control for a Formula One car with variable parameters. *Vehicle System Dynamics*, 52(5), 653–678. doi : 10.1080/00423114.2014.889315. _eprint : <https://doi.org/10.1080/00423114.2014.889315>.
- Perot, E., Jaritz, M., Toromanoff, M. & De Charette, R. (2017). End-to-End Driving in a Realistic Racing Game with Deep Reinforcement Learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 474–475. doi : 10.1109/CVPRW.2017.64.
- Rajamani, R. (2012). *Vehicle Dynamics and Control*. Boston, MA : Springer US. doi : 10.1007/978-1-4614-1433-9.
- Raji, A., Liniger, A., Giove, A., Toschi, A., Musiu, N., Morra, D., Verucchi, M., Caporale, D. & Bertogna, M. (2022). Motion Planning and Control for Multi Vehicle Autonomous Racing at High Speeds. *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2775–2782. doi : 10.1109/ITSC55140.2022.9922239.

- Rawlings, J. B., Mayne, D. Q. & Diehl, M. M. (2020). *Model Predictive Control : Theory, Computation, and Design*. Nob Hill Publishing. Repéré à <http://www.nobhillpublishing.com/mpc-paperback/index-mpc.html>.
- Remonda, A., Krebs, S., Veas, E., Luzhnica, G. & Kern, R. (2019). Formula RL : Deep Reinforcement Learning for Autonomous Racing using Telemetry Data. doi : 10.13140/RG.2.2.30678.09283.
- Scheffe, P., Henneken, T. M., Kloock, M. & Alrifae, B. (2023). Sequential Convex Programming Methods for Real-Time Optimal Trajectory Planning in Autonomous Vehicle Racing. *IEEE Transactions on Intelligent Vehicles*, 8(1), 661–672. doi : 10.1109/TIV.2022.3168130.
- Sebestik, J. (1992). *Logique et mathématique chez Bernard Bolzano*. Paris : J. Vrin.
- Sezer, V. & Gokasan, M. (2012). A novel obstacle avoidance algorithm : “Follow the Gap Method”. *Robotics and Autonomous Systems*, 60(9), 1123–1134. doi : 10.1016/j.robot.2012.05.021.
- Sinha, A., O’Kelly, M., Zheng, H., Mangharam, R., Duchi, J. & Tedrake, R. [arXiv :2003.03900 [cs]]. (2020). FormulaZero : Distributionally Robust Online Adaptation via Offline Population Synthesis. arXiv. Repéré à <https://arxiv.org/abs/2003.03900>.
- Sopasakis, P., Fresk, E. & Patrinos, P. (2020). OpEn : Code Generation for Embedded Nonconvex Optimization*. *IFAC-PapersOnLine*, 53(2), 6548–6554. doi : 10.1016/j.ifacol.2020.12.071.
- Stella, L., Themelis, A., Sopasakis, P. & Patrinos, P. [arXiv :1709.06487 [math.OC]]. (2017). A Simple and Efficient Algorithm for Nonlinear Model Predictive Control. arXiv. Repéré le 2026-06-08 à <http://arxiv.org/abs/1709.06487>.
- Stellato, B., Banjac, G., Goulart, P., Bemporad, A. & Boyd, S. (2020). OSQP : an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4), 637–672. doi : 10.1007/s12532-020-00179-2.
- Subosits, J., Lee, J., Manuel, S., Tylkin, P. & Balachandran, A. [arXiv :2412.03803 [cs]]. (2024). Towards an Autonomous Test Driver : High-Performance Driver Modeling via Reinforcement Learning. arXiv. Repéré le 2026-02-01 à <http://arxiv.org/abs/2412.03803>.
- Sutton, R. S. & Barto, A. G. (2018). *Reinforcement Learning : An Introduction* (éd. 2). MIT Press. Repéré à <https://mitpress.mit.edu/9780262039246/reinforcement-learning/>.

- Theodosis, P. A. & Gerdes, J. C. (2011). Generating a Racing Line for an Autonomous Racecar Using Professional Driving Techniques. *ASME 2011 Dynamic Systems and Control Conference and Bath/ASME Symposium on Fluid Power and Motion Control, Volume 2*, pp. 853–860. doi : 10.1115/DSCC2011-6097.
- Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Shen, A. T. J. & Younis, O. G. [Language : en]. (2023). Gymnasium. [object Object]. Repéré le 2024-03-16 à <https://zenodo.org/record/8127026>.
- Tătulea-Codrean, A., Mariani, T. & Engell, S. (2020). Design and Simulation of a Machine-learning and Model Predictive Control Approach to Autonomous Race Driving for the F1/10 Platform. *IFAC-PapersOnLine*, 53(2), 6031–6036. doi : 10.1016/j.ifacol.2020.12.1669.
- Verschueren, R., Zanon, M., Quirynen, R. & Diehl, M. (2016). Time-optimal race car driving using an online exact hessian based nonlinear MPC algorithm. *2016 European Control Conference (ECC)*, pp. 141–147. doi : 10.1109/ECC.2016.7810277.
- Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., Duijkeren, N. v., Zanelli, A., Novoselnik, B., Albin, T., Quirynen, R. & Diehl, M. (2022). acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 14(1), 147–183. doi : 10.1007/s12532-021-00208-8.
- Wächter, A. & Biegler, L. T. (2005). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1), 25–57. doi : 10.1007/s10107-004-0559-y.
- Williams, G., Drews, P., Goldfain, B., Rehg, J. M. & Theodorou, E. A. (2016). Aggressive driving with model predictive path integral control. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1433–1440. doi : 10.1109/ICRA.2016.7487277.
- Williams, G., Goldfain, B., Drews, P., Rehg, J. M. & Theodorou, E. A. [Version Number : 1]. (2017). Autonomous Racing with AutoRally Vehicles and Differential Games. arXiv. Repéré le 2026-02-08 à <https://arxiv.org/abs/1707.04540>.
- Williams, G., Drews, P., Goldfain, B., Rehg, J. M. & Theodorou, E. A. (2018). Information-Theoretic Model Predictive Control : Theory and Applications to Autonomous Driving. *IEEE Transactions on Robotics*, 34(6), 1603–1622. doi : 10.1109/TRO.2018.2865891.

- Wischnewski, A., Geisslinger, M., Betz, J., Betz, T., Fent, F., Heilmeier, A., Hermansdorfer, L., Herrmann, T., Huch, S., Karle, P., Nobis, F., Ögretmen, L., Rowold, M., Sauerbeck, F., Stahl, T., Trauth, R., Lienkamp, M. & Lohmann, B. [arXiv : 2202.03807 [cs] Number : arXiv :2202.03807]. (2022). Indy Autonomous Challenge – Autonomous Race Cars at the Handling Limits. arXiv.
- Wurman, P. R., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T. J., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., Gilpin, L., Khandelwal, P., Kompella, V., Lin, H., MacAlpine, P., Oller, D., Seno, T., Sherstan, C., Thomure, M. D., Aghabozorgi, H., Barrett, L., Douglas, R., Whitehead, D., Dürr, P., Stone, P., Spranger, M. & Kitano, H. (2022). Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 602(7896), 223–228. doi : 10.1038/s41586-021-04357-7.
- Xue, H., Yue, T. & Dolan, J. M. [arXiv :2309.09186 [cs]]. (2023). Spline-Based Minimum-Curvature Trajectory Optimization for Autonomous Racing. arXiv. Repéré le 2026-03-11 à <http://arxiv.org/abs/2309.09186>.
- Zaharia, M. A., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F. & Zumar, C. (2018). Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Eng. Bull.*, 41, 39–45. Repéré à <https://api.semanticscholar.org/CorpusID:83459546>.
- Zanelli, A., Domahidi, A., Jerez, J. & Morari, M. (2020). FORCES NLP : an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, 93(1), 13–29. doi : 10.1080/00207179.2017.1316017.