

Modélisation de la cinématique/dynamique d'une voiture RC

Chen Kevin, Sabri Ameziane, Amaury Rodriguez, Abdelaziz
Kallel, Paul Pannetier

Encadrant: Faiz Ben Amar



Table des matières

1 Présentation du projet	3
2 Étude du modèle	3
2.1 Modèle à quatre roues non linéaire	3
2.2 Modèle bicyclette	5
3 Résolution des équations différentielles	8
3.1 Choix du solver d'équations différentielles	8
3.2 Résolution sur Matlab	9
4 Modélisation du LiDAR	10
5 Changement de l'angle de braquage	11
6 Analyse des résultats	14
7 Conclusions et Axes D'Améliorations	18
Bibliographie et Annexe	19

1 Présentation du projet

L'objectif de ce projet est l'obtention d'un modèle de véhicule autonome de robot mobile rapide au vue de participer à la compétition de course de mini-voiture autonome organisé par l'Université Paris-Saclay en avril 2022.

En effet, le but est de construire une voiture permettant de finir un circuit prédéfini le plus rapidement possible. Le modèle et les dimensions de voiture étant imposées par les organisateurs, il s'agit d'une voiture électrique à 4 roues motrices (Tamiya XV-01 Lancia Delta). Ces voitures très légères et rapides nécessitent le développement d'un modèle dynamique qui prend en compte l'inertie de la voiture, les forces au sol et les glissements entre les roues et le sol.

Nous supposons dans cette étude que la vitesse du robot est constante et que nous aurons seulement à gérer l'angle de braquage des roues avant afin de maintenir le robot sur le circuit. Nous supposons également qu'il n'existe pas d'obstacle (pas de concurrent) sur le circuit.

Le modèle de simulation nécessite également la construction d'un modèle de l'environnement qui est constitué d'un circuit fermée avec des bordures qui servent à la sécurité et au détecteur LiDAR embarqué sur le véhicule. Ce capteur fournit un nuage de points 2D dans un plan sur 360° avec une fréquence 10 Hz.

Un modèle de simulation du capteur a été développé pour rendre compte de l'environnement local du robot.

Pour finir, nous avons proposé un stratégie de commande qui permet d'ajuster l'angle de braquage en fonction de l'environnement local et la configuration du robot sur le circuit.

Ces différents modules ont été développés sous Matlab. Les résultats sont illustrés à travers des animations graphiques et des courbes indispensables pour l'analyse et l'optimisation des différents paramètres de la course.

2 Étude du modèle

2.1 Modèle à quatre roues non linéaire

Pour l'étude du modèle, on s'est appuyé sur la thèse réalisée par Mohamed Fnadi qui se trouve en annexe.

On s'intéresse tout d'abord à un modèle de type 4 roues comme on peut le voir ci-dessous :

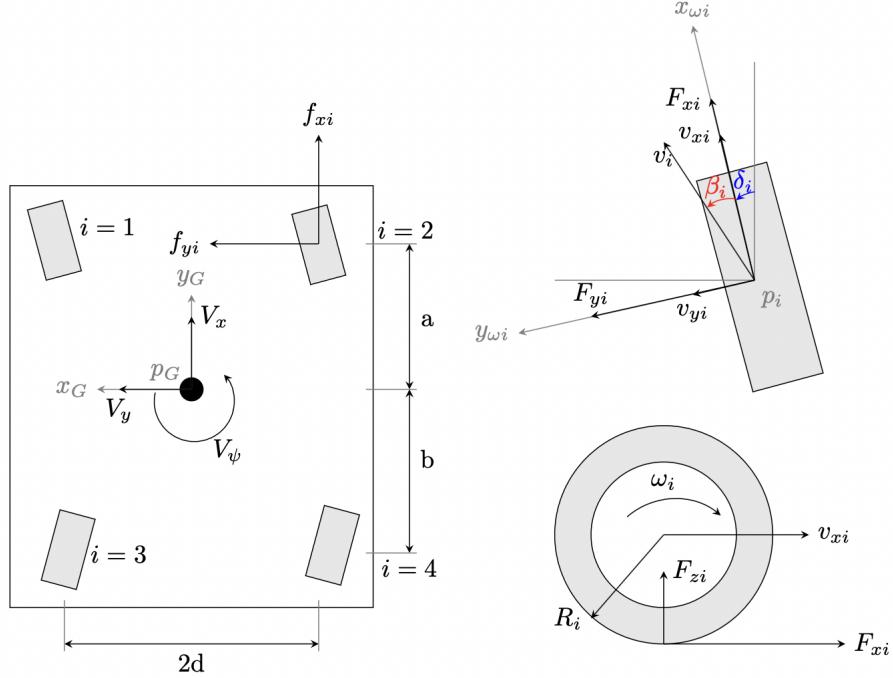


FIGURE 1 – Modèle à 4 roues

En supposant dans un premier temps que le véhicule évolue dans un plan de lacet horizontal, on applique le principe fondamental de la dynamique et on obtient les différentes équations différentielles des différentes dynamiques (longitudinales V_x , latérale V_y , de lacet V_ψ) ci-dessous :

$$\begin{aligned} m\dot{V}_x &= m\dot{V}_y V_\psi + \sum_{i=1}^4 f_{xi}, \\ m\dot{V}_y &= -m\dot{V}_x V_\psi + \sum_{i=1}^4 f_{yi}, \\ I_z \dot{V}_\psi &= a(f_{y1} + f_{y2}) - b(f_{y3} + f_{y4}) + d(-f_{x1} + f_{x2} - f_{x3} + f_{x4}), \end{aligned}$$

avec m la masse du véhicule, f_{xi} et f_{yi} les forces de contact roue/sol le long des axes longitudinale et latérale de la roue i , I_z le moment d'inertie du véhicule au centre de gravité et a , b , d l'empattement avant et arrière et la distance de la demi-voie.

Les composantes de la force longitudinale f_{xi} et latérale f_{yi} du pneu dans le repère du véhicule s'écrivent comme suit :

$$\begin{aligned} f_{xi} &= F_{xi} \cos \delta_i - F_{yi} \sin \delta_i, \\ f_{yi} &= F_{xi} \sin \delta_i + F_{yi} \cos \delta_i, \end{aligned} \quad \text{pour } i \text{ allant de 1 à 4,}$$

avec δ_i l'angle de braquage à la roue i, on émet l'hypothèse aussi que les angles de braquage des roues droites et gauches de chaque essieu sont égaux. De manière générale, les efforts longitudinaux et latéraux dépendent de plusieurs paramètres (ex : le coefficient de frottement, taux de glissement, etc.), dans notre modélisation on va supposer un comportement linéaire de la roue pneumatique et pour cela on va introduire le modèle bicyclette.

2.2 Modèle bicyclette

On dérive notre modèle non linéaire à 4 roues présenté précédemment pour obtenir en modèle bicyclette c'est-à-dire que l'on regroupe les deux roues avant ainsi que les deux roues arrière comme le montre le schéma ci-dessous :

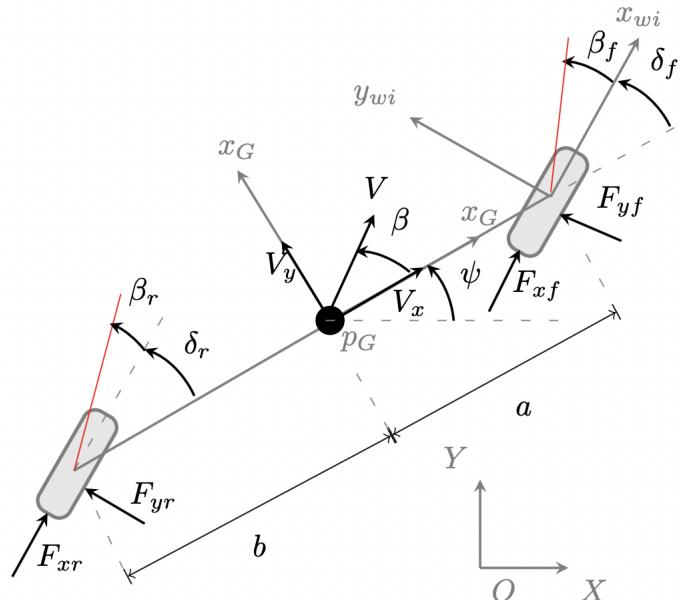


FIGURE 2 – Modèle dynamique bicyclette

Avec ce modèle les équations qu'on avait précédemment se simplifient et donnent :

$$\begin{aligned} m\dot{V}_x &= m\dot{V}_y V_\psi + 2f_{xf} + 2f_{xr}, \\ m\dot{V}_y &= -m\dot{V}_x V_\psi + 2f_{yf} + 2f_{yr}, \\ I_z \dot{V}_\psi &= 2af_{yf} - 2bf_{yr}, \end{aligned}$$

avec $f_{x(f,r)}$ et $f_y(f,r)$ les forces longitudinales et latérales avant et arrière dans le repère du véhicule R_G qui sont exprimées comme ceci :

$$\begin{aligned} f_{x(f,r)} &= F_{x(f,r)} \cos \delta_{(f,r)} - F_{y(f,r)} \sin \delta_{(f,r)}, \\ f_{y(f,r)} &= F_{x(f,r)} \sin \delta_{(f,r)} - F_{y(f,r)} \cos \delta_{(f,r)}, \end{aligned}$$

avec $F_{x(f,r)}$ et $F_{y(f,r)}$ les forces longitudinales et latérales avant et arrière dans le repère R_w et δ_f et δ_r les angles de braquage avant et arrière,
On obtient les équations de mouvement du véhicule dans le repère inertiel R_G suivant :

$$\begin{aligned} \dot{X} &= V_x \cos \psi - V_y \sin \psi \\ \dot{Y} &= V_x \sin \psi - V_y \cos \psi \\ \dot{\psi} &= V_\psi \end{aligned}$$

où \dot{X} et \dot{Y} sont les vitesses longitudinale et latérale du véhicule dans le repère inertiel R_G , $\dot{\psi}$ l'angle d'orientation du véhicule dans le plan de lacet (le lacet est le mouvement de rotation horizontal d'un mobile autour de son axe vertical), V_x , V_y , V_ψ les dynamiques longitudinale, latérale et de lacet du véhicule.
De plus, en utilisant les équations cinématiques du véhicule et en introduisant les angles de dérive avant et arrière β_r et β_f , on a les relations :

$$\begin{aligned} \tan(\delta_f + \beta_f) &= \frac{V_y + aV_\psi}{V_x}, \\ \tan(\delta_r + \beta_r) &= \frac{V_y - bV_\psi}{V_x}, \end{aligned}$$

En supposant que les angles de braquage et de dérive sont faibles, on peut réécrire ces deux équations comme suit :

$$\begin{aligned} \beta_f &= \frac{V_y + aV_\psi}{V_x} - \delta_f, \\ \beta_r &= \frac{V_y - bV_\psi}{V_x} - \delta_r, \end{aligned}$$

En introduisant l'hypothèse de pseudo-glissement, on obtient une relation de proportionnalité entre l'effort latéral et l'angle de dérive c'est-à-dire que l'on a :

$$F_{yi} = C_i \beta_i$$

où C_i sont les rigidités latérales de la roue i
En utilisant cette hypothèse et les équations précédentes, on obtient :

$$\begin{aligned} F_{yf} &= C_f \left(\frac{V_y + aV_\psi}{V_x} - \delta_f \right), \\ F_{yr} &= C_r \left(\frac{V_y - bV_\psi}{V_x} - \delta_r \right), \end{aligned}$$

où C_f et C_r sont les rigidités de dérives latérales avant et arrière et $F_{y(f,r)}$ les forces latérales avant et arrière.

En s'intéressant à la géométrie locale du terrain(ex : pente, bosse, ect.), cela peut provoquer des changement de trajectoire qu'il faut prendre en compte comme montré dans le schéma ci-dessous :

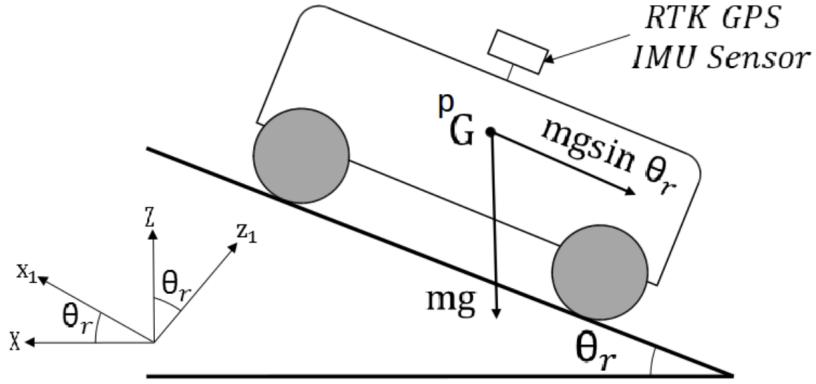


FIGURE 3 – Véhicule avec un terrain en pente

On a aussi l'hypothèse que la vitesse longitudinale $V_x (\dot{V}_x = 0)$ est constante, et du coup en prenant en compte la géométrie locale du terrain, on obtient ces nouvelles équations :

$$\begin{aligned}\dot{V}_y &= -V_x V_\psi - 2 \frac{V_y}{m V_x} (C_f + C_r) - 2 \frac{V_\psi}{m V_x} (a C_f + b C_r) + 2 \frac{C_f}{m} \delta_f - 2 \frac{C_r}{m} \delta_r + g \cos \phi_r \sin \theta_r \\ \dot{V}_\psi &= -2 \frac{V_y}{I_z V_x} (a C_f - b C_r) - 2 \frac{V_\psi}{I_z V_x} (a^2 C_f + b^2 C_r) + 2 \frac{a C_f}{I_z} \delta_f - 2 \frac{b C_r}{I_z} \delta_r\end{aligned}$$

avec m la masse du véhicule, I_z le moment d'inertie du véhicule, C_f et C_r les rigidités de dérives latérales avant et arrière, ϕ_r et θ_r les angles de la géométrie locale du terrain dans notre cas on considère que le terrain est plat donc ces 2 termes sont nuls, on considère aussi que δ_r l'angle de braquage est nul (car ce n'est pas un véhicule aux 4 roues motrices).

Au final, on a ces 6 équations :

$$\begin{aligned}\dot{X} &= V_x \cos \psi - V_y \sin \psi \\ \dot{Y} &= V_x \sin \psi - V_y \cos \psi \\ \dot{\psi} &= V_\psi \\ \dot{V}_x &= 0 \\ \dot{V}_y &= -V_x V_\psi - 2 \frac{V_y}{m V_x} (C_f + C_r) - 2 \frac{V_\psi}{m V_x} (a C_f + b C_r) + 2 \frac{C_f}{m} \delta_f \\ \dot{V}_\psi &= -2 \frac{V_y}{I_z V_x} (a C_f - b C_r) - 2 \frac{V_\psi}{I_z V_x} (a^2 C_f + b^2 C_r) + 2 \frac{a C_f}{I_z} \delta_f\end{aligned}$$

3 Résolution des équations différentielles

3.1 Choix du solver d'équations différentielles

Sur Matlab, on a plusieurs choix comme ode45, ode23tb, ode15s etc...
On a choisi d'utiliser ode45 pour deux raisons :

- La rapidité de calcul, nécessaire pour une simulation qui appellera notre solver tous les 0.5 seconde environ
- Le pas d'intégration variable en fonction de la variation de la solution, ce qui est bénéfique pour accélérer les calculs dans les cas les plus simples

Exemple de résolution avec ode45 de l'équation $y' = -0.5y + 50$

```
1 t_0=0;
2 y_0=1;
3 t_f=10;
4
5 odefun= @(t,y) -0.5*(y)+50;
6 [t,y]=ode45(odefun,[t_0,t_f],y_0);
7
8 figure (Name="ode45");
9 plot(t,y, '--*b');
10
```

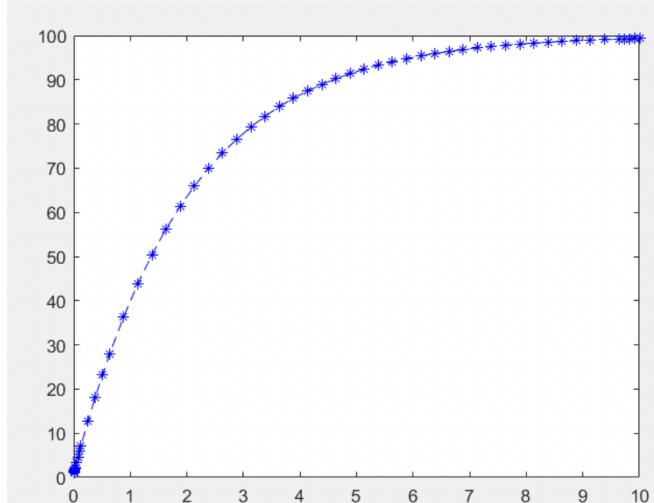


FIGURE 4 – Graphique de y en fonction de t

3.2 Résolution sur Matlab

Afin de modéliser notre voiture sur Matlab, on a implémenter nos 6 équations et à l'aide de la fonction ode45 présentée précédemment, dans notre cas on a un vecteur de dimension 6, un intervalle de temps sur lequel on veut résoudre l'équation et les conditions initiales qui sont la position initiale de la voiture et sa vitesse initiale. Cette fonction nous retourne un tableau avec les solutions des équations pour chaque temps de l'intervalle.

On a donné des valeurs à toutes les constantes en s'appuyant sur les caractéristiques de cette [voiture](#). La seule variable que l'on modifie pour modifier les trajectoires est δ_f l'angle de braquage avant.

On a ensuite fait une fonction qui nous permet d'afficher le mouvement de la voiture en fonction du temps, par exemple on peut modéliser un virage en donnant une condition sur δ_f qui va modifier en fonction de la position en x de la voiture comme on peut le voir dans cette [vidéo](#) avec ces conditions initiales :

Paramètres	Symbole	Valeur	Unité
Masse du véhicule	m	0.340	kg
Rigidité de dérive latérale avant	C_f	1000	N/rad
Rigidité de dérive latérale arrière	C_r	1000	N/rad
Position du centre de gravité par rapport aux roues avant	a	0.2	m
Position du centre de gravité par rapport aux roues arrières	b	0.2	m
Moment d'inertie	I_z	0.01	$kg.m^2$

4 Modélisation du LiDAR

Pour implémenter le LiDAR de la voiture, on modélise ce capteur sur Matlab aussi. Le LiDAR équipé sur la voiture a une résolution de 0.2° , mais pour des questions de puissance de calcul et d'optimisation du code, nous ne prendrons pas tous les points calculables. La résolution du LiDAR simulée est modifiable selon la précision désirée.

Pour la simulation du LiDAR, nous utilisons une fonction qui prend en argument :

- la résolution du LiDAR(pas_θ) en radian
- la portée du LiDAR(r) en dm
- les coordonnées de la position de la voiture
- les coordonnées du circuit extérieur
- les coordonnées du circuit intérieur

On a ici par exemple l'utilisation de notre fonction pour 2 tirs :

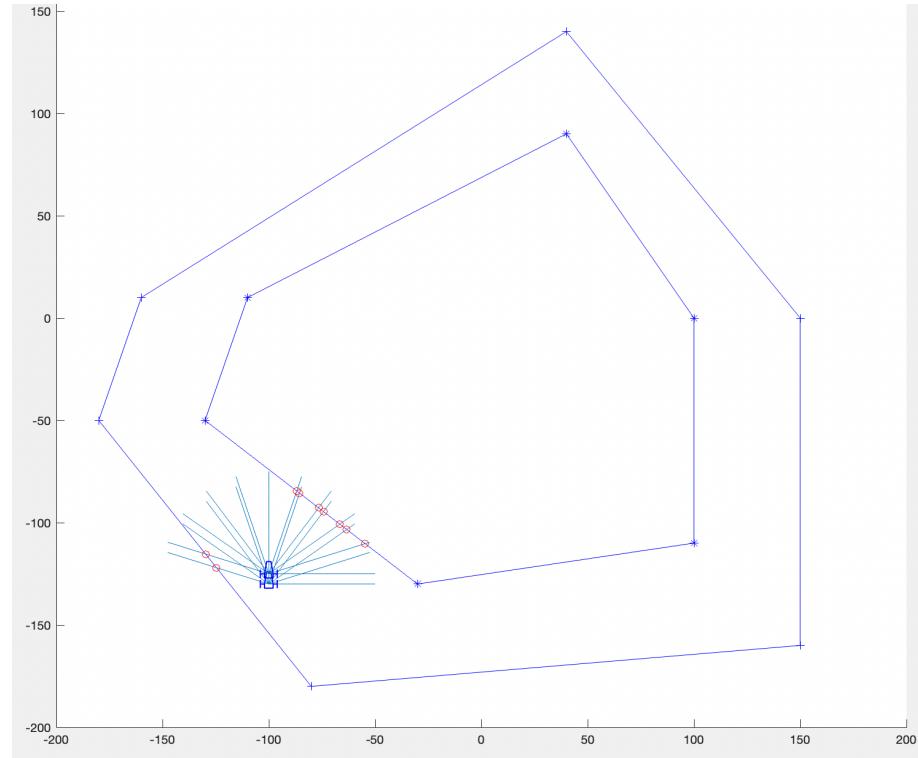


FIGURE 5 – Exemple de tir

La fonction à partir de ces arguments va calculer un nombre de points (dé-

pendant de la résolution), tous à une distance r de la voiture, espacés d'un pas_θ . Ces points représentent tous les tirs de LiDAR. Nous prenons en compte uniquement les points se situant devant la voiture. À partir des points obtenus la fonction calcule les intersections entre les tirs de LiDAR et les bords du circuit intérieur et extérieur grâce à la fonction `polyxpoly` de Matlab :

$$[xi, yi] = \text{polyxpoly}(x1, y1, x2, y2)$$

qui renvoie les points d'intersection de deux segments dans un système cartésien, avec les sommets définis par $x1, y1, x2$ et $y2$.

La fonction LiDAR renvoie 4 tableaux avec les :

- Coordonnées x des intersections avec le circuit intérieur x_{fint}
- Coordonnées y des intersections avec le circuit intérieur y_{fint}
- Coordonnées x des intersections avec le circuit extérieur x_{fext}
- Coordonnées y des intersections avec le circuit extérieur y_{fext}

Avec ces tableaux on a une fonction qui permet de calculer la distance de la voiture aux intersections.

Maintenant, avec toutes ces informations il faut que l'on arrive à modifier l'angle de braquage selon la position de la voiture dans le circuit.

5 Changement de l'angle de braquage

Après avoir déterminé les bordures intérieur et extérieur du circuit en une multitude de points grâce au LiDAR, il faut pouvoir changer l'angle de braquage à chaque pas de temps en fonction de la position de la voiture et des données mesurées avec le LiDAR.

Pour cela notre première idée était d'approximer les deux côtés du circuit par des fonctions polynomiales en résolvant le problème des moindres carrés. Pour cela on avait pour chaque côté du circuit une liste de n points : (x_i) et (y_i) avec $1 \leq i \leq n$

On calcule le polynôme de meilleur approximation de degré $m \leq n$ fixé, ces deux polynômes de degré m ont des coefficients (a_i) avec $1 \leq i \leq m+1$ caractérisé par :
 $tVVA = {}^tV y$ ou $V_{ij} = x_i^{(j-1)}$

Matlab inclut nativement un solveur pour ce type d'équation matricielle, la solution est donnée par : $a = (V'V) / (V'y)$

Par exemple pour un morceau de circuit, imaginons que le LiDAR nous renvoie cette liste de points pour le circuit extérieur :

$x : [-3.55, -3.05, -2.75, -2.49, -2.29, -2.01, -1.71, -1.29, -0.85, -0.37, 0.01, 0.49, 1.03]$

$y : [3.26, 3.04, 2.55, 1.99, 1.52, 0.86, 0.24, -0.36, -0.56, -0.35, 0.01, 0.43, 0.53]$

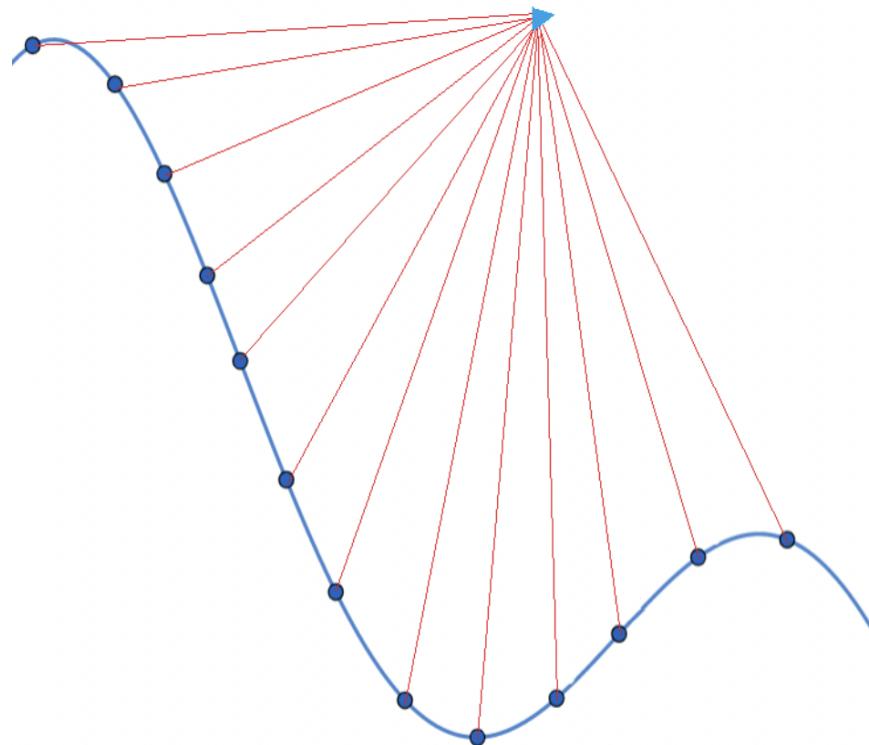


FIGURE 6 – Approximation du circuit

On peut approximer ce morceau par des polynômes de différents ordre et on voit que le polynôme d'ordre 4 donne la meilleure approximation comme on peut voir ci-dessous :

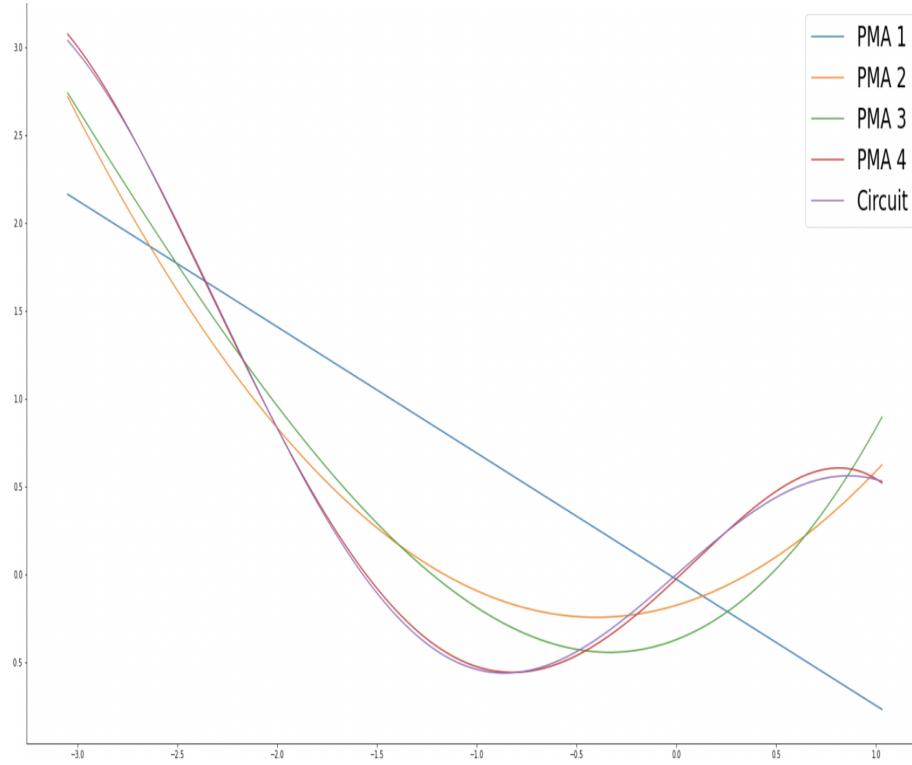


FIGURE 7 – Polynôme de meilleur approximation

Cependant même avec une bonne approximation du circuit par des polynômes, il est difficile d'en trouver la bonne trajectoire notamment en cas de circuits qui font des demis-tours (un virage en épingle par exemple), nous avons donc opté pour une autre solution.

L'autre solution pour modifier l'angle de braquage après que l'on ait déterminé les deux cotés du circuit à l'aide du LiDAR et comparé la moyenne des distances entre les points du circuit et la voiture avec le côté intérieur moyint et le côté extérieur moyext. Nous calculons donc le coefficient $A = \frac{moyint}{moyext}$:

-Si $A \approx 1$, alors la voiture se trouve approximativement au milieu du circuit, il faut qu'elle continue tout droit donc on change l'angle de braquage à 0

-Si $A \gg 1$, alors la voiture est trop à l'intérieur, elle doit se diriger vers l'extérieur donc on change l'angle de braquage à + l'angle de braquage qu'on avait définie en constante

-Si $A \ll 1$, alors la voiture est trop à l'extérieur, elle doit se diriger vers l'intérieur donc on change l'angle de braquage à - l'angle de braquage qu'on avait défini en constante

6 Analyse des résultats

Dans un premier temps, nous avons réussi à modéliser une trajectoire telle que la voiture fasse le tour du circuit pendant un certain temps à une vitesse constante. Cependant, cette trajectoire n'était ni optimale, ni réaliste. En effet, comme nous pouvons le voir sur la figure 8, la voiture n'a pas une trajectoire régulière. Notre véhicule effectue beaucoup trop de zigzags. Cela était dû au fait que notre programme comparait les distances toutes les 0.5 secondes et rectifiait la trajectoire avec un angle de braquage constant (0.9°). Le problème est qu'en réalité, un véhicule ne va pas chercher à modifier sa trajectoire à tout instant pour rester au centre du circuit mais plutôt à optimiser sa course en ayant la trajectoire la plus fluide possible.

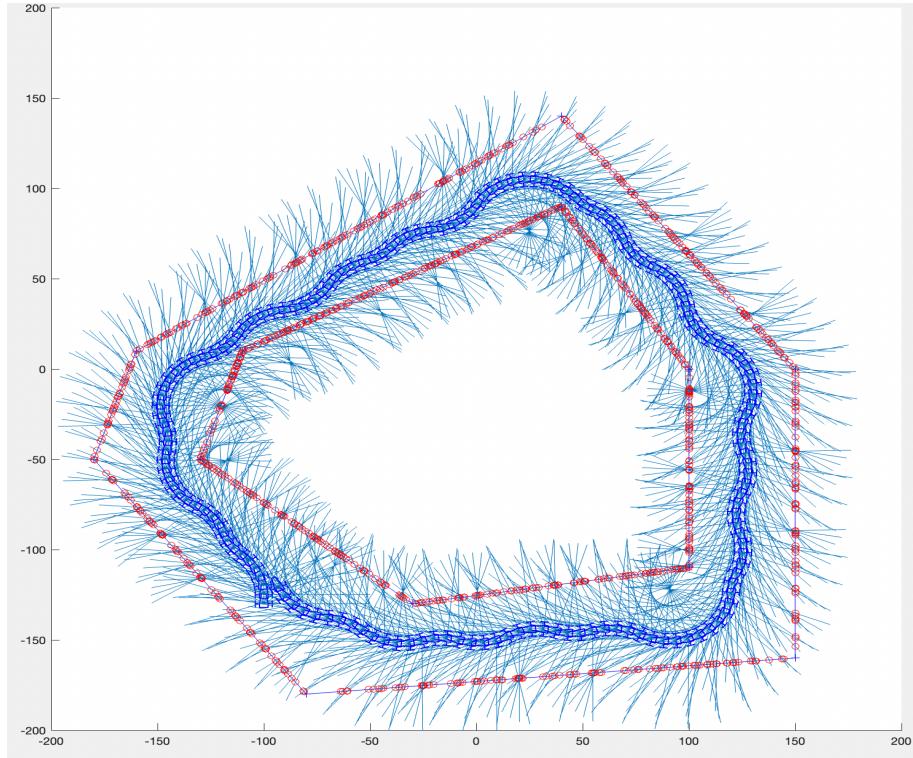


FIGURE 8 – Schéma sur circuit 1

Ainsi, nous avons opté pour une méthode visant à optimiser la trajectoire et la rendre plus réaliste. Cette fois ci, nous avons collecté, toujours toutes les 0.5 secondes, tous les points d'intersections du Lidar avec le circuit à droite et à gauche. Ensuite, en faisant la moyenne de ces points pour chaque côté, nous en avons fait le rapport (moyenne intérieure/moyenne extérieure). Si ce rapport est faible (entre 0.5 et 2), la voiture est assez centrée donc nous allons lui donner

un angle de braquage nul. Cependant, si ce rapport est trop faible (inférieur à 0.5) ou trop grand (supérieur à 2), c'est que la voiture va se heurter au circuit, nous allons donc lui donner un angle de braquage de 0.8 degré (à droite ou bien à gauche en fonction du rapport). Comme nous pouvons le voir sur la figure 9 ([video](#)), la trajectoire est beaucoup plus fluide et réaliste.

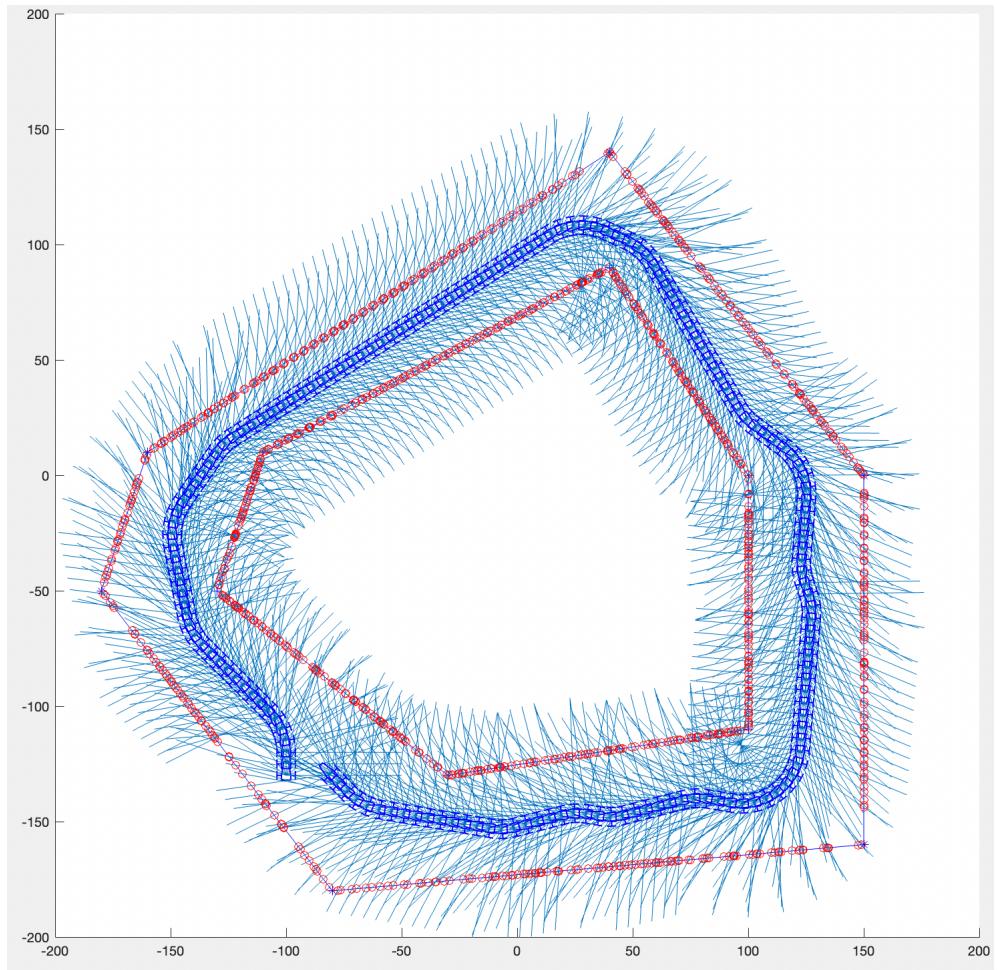


FIGURE 9 – Schéma sur circuit 1

On a testé notre programme sur deux autres circuits un peu différent notamment un circuit qui provient de la F1([vidéo](#)) comme vous pouvez le voir ci-dessous :

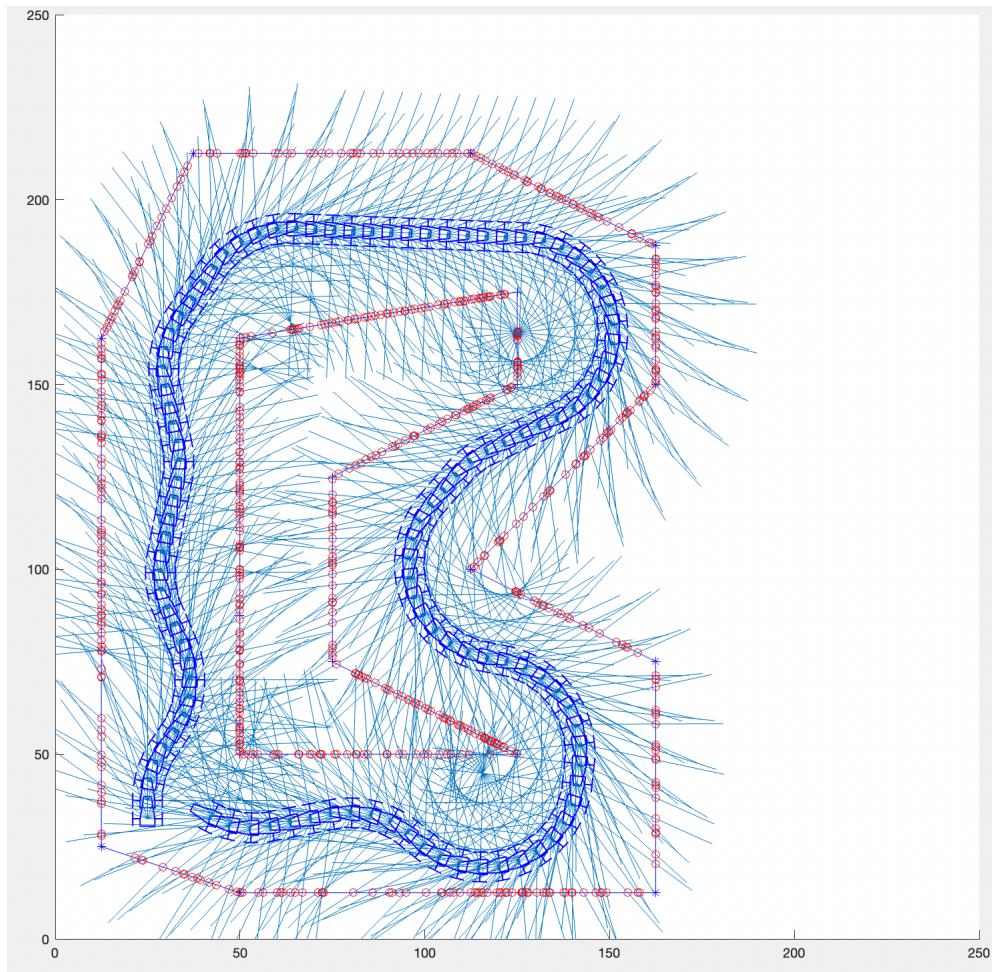


FIGURE 10 – Schéma sur circuit 2

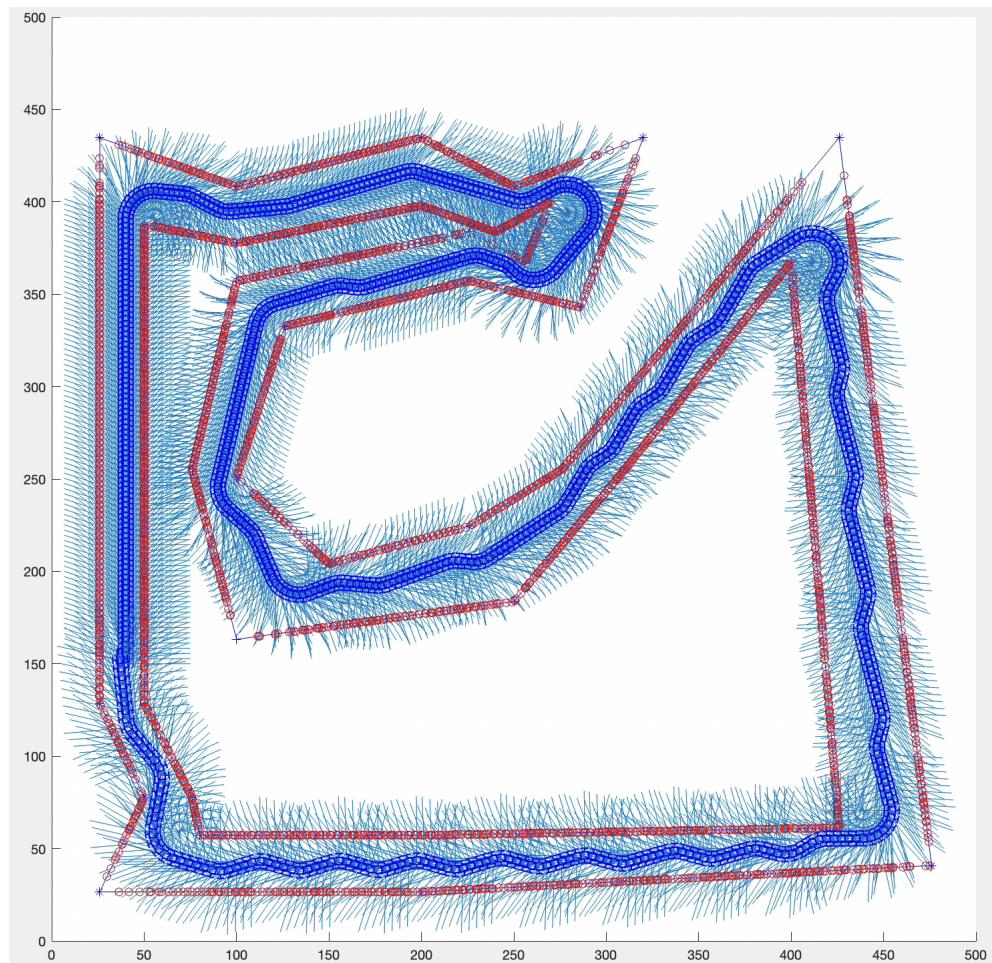


FIGURE 11 – Schéma sur un circuit de F1

7 Conclusions et Axes D'Améliorations

Au final, nous avons bien réussi à modéliser un modèle de véhicule autonome qui permet de finir un circuit sans toucher les bordures du circuit sur Matlab, avec l'étude du modèle qui nous a permis d'obtenir les équations différentielles qu'on devait résoudre, la construction du circuit, la modélisation du véhicule et du LiDAR et enfin le changement de l'angle de braquage en fonction de la position de la voiture.

Pour notre projet, à partir de ce que nous avons fait nous pourrions améliorer quelques axes. Dans un premier temps la voiture se déplace sur le circuit « sans le connaître », c'est à dire sans anticiper le reste du circuit. Cette approche peut être utile dans un premier temps pour cartographier le circuit qui selon les règles de la course est inconnu avant le jour j. Une fois le circuit connu il est donc nécessaire dans un premier temps de calculer la meilleure trajectoire pour cela nous pouvons nous appuyer sur le document : « A Sequential Two-Step Algorithm for Fast Generation of Vehicle Racing Trajectories » et implémenter une de leurs méthodes. Leur méthode consiste à d'extraire une ligne moyenne sur laquelle on peut calculer la courbure en fonction de l'abscisse curviligne. La première étape consiste à calculer le profil de la vitesse maximale de la voiture en fonction de la courbure de la ligne moyenne et de ses capacités d'accélération. La seconde étape calcule la trajectoire optimale du robot ainsi que l'angle de braquage en fonction de l'abscisse curviligne (distance parcourue). Cet article fait une optimisation sur tout le circuit, il sera tout à fait possible de l'adapter pour optimiser la trajectoire sur un horizon compatible avec la portée du capteur et exploiter le résultat dans une commande prédictive de l'angle de braquage. Après avoir obtenue la trajectoire optimale que nous devrions suivre comme base de départ il faut ensuite prendre en compte les autres voitures présentes sur le circuit et éviter les collisions tout en les dépassant et faire des tours de circuit le plus rapidement possible. Plusieurs options s'offrent à nous :

-De coder un programme dictant à la voiture son comportement en fonction de la position des voitures alentours, de notre vitesse, de l'adhérence. Nous pouvons voir qu'enormément de variables entre en jeu et rendent ainsi la tâche très complexe. Nous pensons que cette méthode serait fastidieuse et sans grand résultat du fait du très grand nombre de scénarios possibles.

-Nous pourrions aussi le faire en implémentant du Machine Learning, en sélectionnant à chaque fois les simulations qui ne se crash pas qui vont le plus loin et le plus vite.

Bibliographie et Annexe

Github avec tous les programmes, URL : https://github.com/Niiivek/Projet_voiture

- [1] Rajesh Rajamani, Vehicle Dynamics and Control,
URL : <https://frlib.org/book/2095593/8848e9>
- [2] Mohamed Fnadi, Commande prédictive et estimation des paramètres d'environnement pour un rover rapide,
URL : <https://cloud.isir.upmc.fr/owncloud/index.php/s/rTR71mmtOvXHIEi#pdfviewer>
- [3] Nitin R. Kapania, John Subosits et J. Christian Gerdès, Sequential Two-Step Algorithm for Fast Generation of Vehicle Racing Trajectories,
URL : <https://arxiv.org/pdf/1902.00606.pdf>