Doc Laptime simulation

# Intro

Cette doc explique le fonctionnement global de la simulation, des docs spécifiques explique le fonctionnement des fonctions modélisant le comportement de la voiture en latérale et en longitudinale.

Toute cette simulation s’inspire de la vidéo de Matlab Racing Loung : <https://fr.mathworks.com/videos/matlab-and-simulink-racing-lounge-steady-state-lap-time-simulation-116109.html>

Globalement la simulation prend comme argument une liste de coordonnées X Y dessinant le circuit ainsi que tous les paramètres physiques de la nécessaires à la simulation véhicule.

L’idée est de calculer le diagramme GGV de la voiture, il contient les cercles de traction à différentes vitesses, dans le but ensuite de modéliser la voiture au maximum de ses performances à chaque instant sur le circuit. Cette méthode permet alors de s’affranchir d’un modèle pilote complexe à mettre en place.

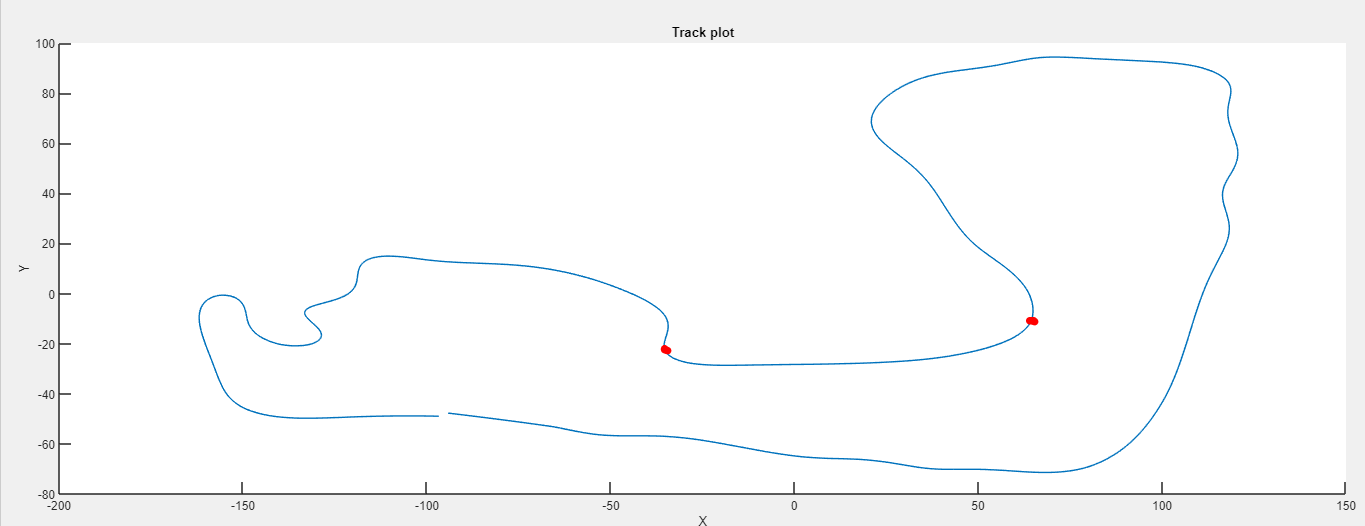
Un point important et le caractère **steady-state** de la simulation, on considère donc que la voiture est tout le temps à l’équilibre il n’y a pas de transitoire !

Dans le document le vert souligne de possibles améliorations à faire sur ce modèle.

# Import du circuit et segmentation

Pour la suite nous aurons besoin de la liste des rayons de courbure du circuit à chaque instant. Pour calculer ces rayons de courbure on utilise la fonction ‘LineCurvature2D’trouvé sur internet (Matlab Exchange) qui nous donne la courbure du tracé. La courbure correspond à l’inverse du rayon de courbure.

Pour la suite de la simulation l’emplacement des rayons de courbure minimaux doivent être connu. Cela nous permettra ensuite de découper le circuit en segment entre deux virages.



Sur la figure ci-dessus on peut voir deux point ayant des rayons de courbure correspondant à des minimums locaux.

Néanmoins comme on le voit sur la figure suivant la courbure du circuit est trop bruité pour obtenir une bonne segmentation :

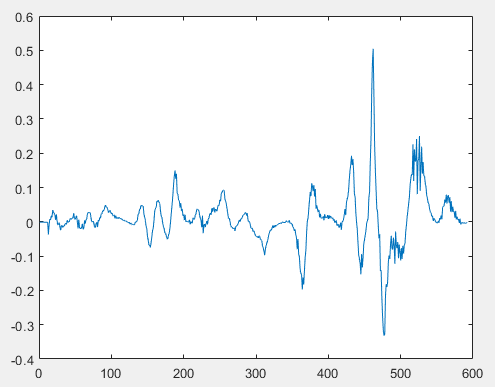
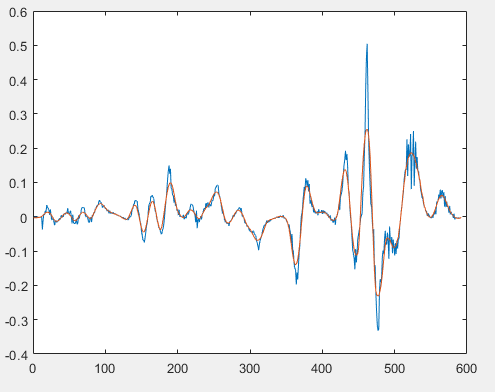


Figure 1 : Tracé de la courbure non filtrée

On lisse donc la courbe avec la fonction ‘sgolayfilt’(ce filtre est cool mais il y a peut-être mieux).



On peut voir en orange la courbe filtrée. On voit que les valeurs pic ne correspondent plus néanmoins le pic à 0.5 serait un rayon de 2m ce qui est impossible selon le règlement et pour la voiture. Il faudrait donc vérifier si ce rayon ainsi lissé correspond vraiment à la réalité.

Une fois la courbure lissée obtenue on peut obtenir les points de courbures maximaux avec la fonction *extrema.* Cette fonction renvois les indices des extrémaux locaux.

On calcul ensuite le rayon qui est l’inverse de la courbure, on le sature entre -150m et 150m et on remplace les NaN (cas ou la courbure est nulle) par 150m.

On calcul aussi la valeur absolue du rayon pour éviter les erreurs de calcul, on viendra à la fin de la simulation pondérer l’accélération latérale par le signe de R pour s’y retrouver.

On calcul finalement le vecteur D contenant les distances entre chaque point du circuit.

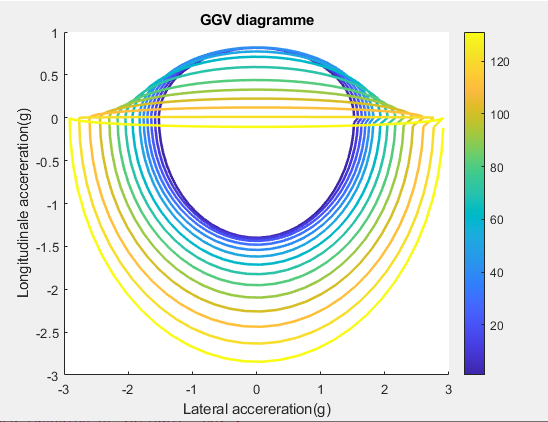
# Calcul du diagramme GGV

Avant de commencer notre boucle sur le circuit on calcule le diagramme GGV pour avoir le potentiel de la voiture. C’est la fonction *makeGGV* qui remplit cette fonction. La variable GGV est constituée de 5 colonnes permettant d’avoir accès aux performances maximale de la voiture. L’idée est de pouvoir connaitre les accélérations maximales que la voiture peut fournir à une vitesse donnée

1. La colonne 1 contient l’accélération longitudinale positive
2. La colonne 2 contient l’accélération longitudinale négative
3. La colonne 3 contient l’accélération latérale max
4. La colonne 4 contient les vitesses de calculs des précédentes valeurs
5. La colonne 5 contient le rayon de courbure associé aux accélération latérale

On fait ici des calculs pour une accélération longitudinale ou latérale pur. Dans la suite on fera une approximation elliptique pour avoir un mix entre les deux. Une amélioration pourrait être de simuler des cas mix !

La fonction *plotGGV* permet d’afficher rapidement le diagramme GGV.



Vitesse (km/h)

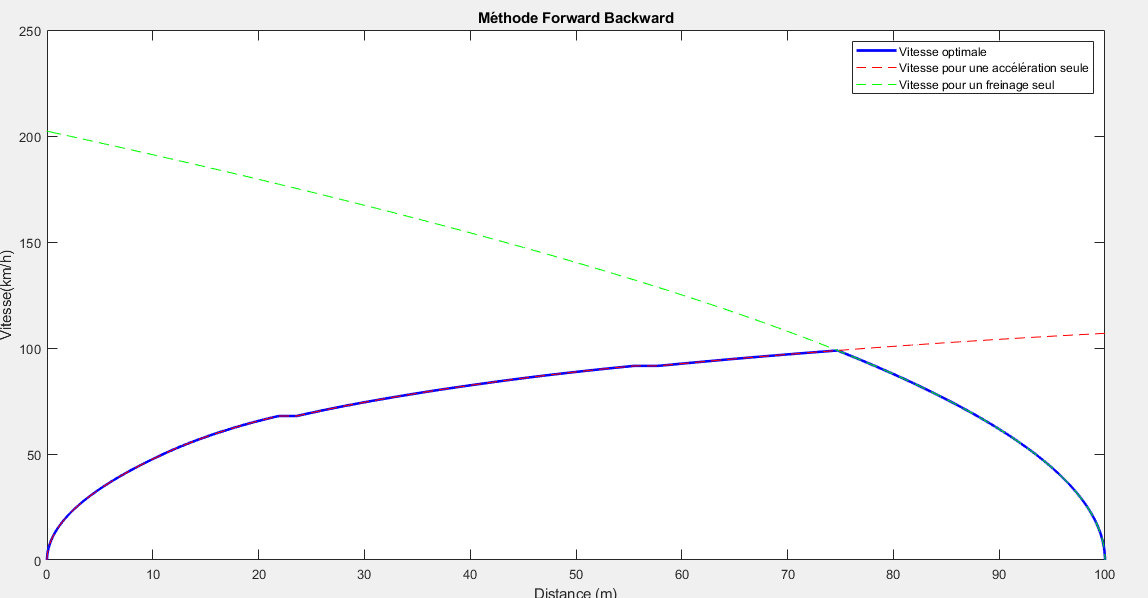
# Boucle sur les segments

On parcourt ensuite chaque segment du circuit en déterminant la vitesse optimale sur chaque segment. Pour cela avec une initialisation on connait toujours la vitesse de départ du segment (0 pour le premier la vitesse finale du segment précédent sinon) on calcul ensuite la vitesse maximale de fin de segment en considérant une accélération purement latérale lors de la corde du virage (ie la fin du segment).

On utilise alors la méthode Forward Backward qui consiste à réaliser deux simulation :

* Une simulation en partant de la vitesse initiale et en accélérant au maximum sans se soucier de la vitesse finale 🡪 Simulation Forward
* Une simulation à l’envers en partant de la vitesse finale et en considérant que l’on a freiné un maximum pour arriver à cette vitesse 🡪 Simulation Backward

On garde ensuite la vitesse minimale du véhicule entre ces deux méthodes en chaque point. Cela permet de trouver la distance de freinage pour arriver avec la bonne vitesse.



La figure si dessus permet d’illustrer cette méthode pour des vitesses initiale et finale nulle.

Cas particulier :

* Si en accélérant au maximum on atteint pas la vitesse finale alors la vitesse finale devient celle atteignable.
* Si en freinant au maximum on atteint pas la vitesse initial (cas d’un segment trop court), Pour l’instant la vitesse initiale n’est pas modifiée et il y a donc une discontinuité de la vitesse entre le premier et le deuxième point du segment.

## Détail Forward

La simulation Forward est réalisé par la fonction du même nom. C’est une boucle sur tous les points du segment, pour chaque nouveau point « i » :

* On estime le temps mit par la voiture pour se déplacer entre le point i-1 et i ce temps est noté dt. Pour estimer ce temps on prend en compte la vitesse et l’accélération en i-1 en considérant que l’accélération reste constante entre les deux points.

On alors : et par intégration : avec Di la distance entre les deux points. On a une équation du second ordre on distingue 3 cas pour la résoudre :

* + et sont nul : C’est normalement impossible car à V=0 l’accélération produite doit être positive néanmoins on fixe dt=0.1s et on affiche un message pour l’utilisateur.
  + = 0 et Dans ce cas-là le problème est simple :
  + > 0 et
* Une fois dt connu on calcul la nouvelle vitesse en i avec .
* Ensuite, avec notre hypothèse steady state on calcul l’accélération latérale subit par la voiture : . Dans le cas ou Ri = 150m alors Gyi = 0g car on a saturé R au début.
* Avec le GGV on peut ensuite connaitre l’accélération longitudinale que peut produire la voiture dans ces conditions de vitesse et d’accélération latéral, c’est la fonction findGxmax qui permet de réaliser cette fonction

Une amélioration possible serait de stopper la boucle après avoir dépassé la vitesse finale attendu pour gagner en temps de calcul.

## Détail Backward

La simulation Backward reprend exactement le même fonctionnement que Forward à l’envers.

Petite subtilité : la formule pour calculer le pas de temps est le même avec une valeur absolue sur l’accélération car la situation est symétrique.

# Sortie de la simulation

En sortie de la fonction *LapTime* plusieurs variables sont disponible :

* La matrice GGV
* La vitesse
* Le temps
* Les deux accélérations
* La distance cumulée

A partir de ces données on peut alors en créer de nouvelles telle que les RPM, la vitesse engagée, les charges sur chaque roue, l’angle du volant, le TPS, la conso d’essence etc…