MASTER ÉLECTRONIQUE, ÉNERGIE ÉLECTRIQUE, AUTOMATIQUE
PARCOURS SYSTÈMES AUTOMATIQUES MOBILES

LOCALISATION, CARTOGRAPHIE ET PLANIFICATION
TRAVAIL PRATIQUE

Le filtrage de Kalman : Développement, Utilisation et Compréhension

---

Étudiants :
Otmane ATTOU
Badreddine SAFI

Enseignant :
Dominique GRUYER

Table des matières

Table des figures

1 Introduction :

Le filtre de Kalman en contexte discret est un estimateur récursif. Cela signifie que pour estimer l'état courant, seule l'estimation de l'état précédent et les mesures actuelles sont nécessaires. L'histoire des observations et des estimations n'est ainsi pas requise. Elle a deux phases : prédiction et mise à jour. La phase de prédiction utilise l'état estime de l'instant précèdent pour produire une estimation de l'état courant. Dans l'étape de mise à jour, les observations de l'instant courant sont utilisées pour corriger l'état prédit dans le but d'obtenir une estimation plus précise.

L'objectif principal de ce TP est de développer un filtre de Kalman étendu (avec un modèle non linéaire). Dans un premier temps, nous allons prendre un ensemble de codes sources dans Matlab et compléter toutes les fonctions utilisées pour implémenter le filtre de Kalman. Ensuite, l'algorithme sera testé sur des données réelles, une analyse plus détaillée de cette méthode sera également réalisée.

2 Filtre de Kalman :

2.1 Principe :

Dans ce TP le filtre de Kalman sera utilisé pour traiter les données d'un signal GPS , le signal de positionnement GPS serait utilisé comme signal d'observation du filtre de Kalman pour corriger la prédiction de la position de la voiture . la figure ci-dessous donne un aperçu globale sur le fonctionnement de ce filtre :

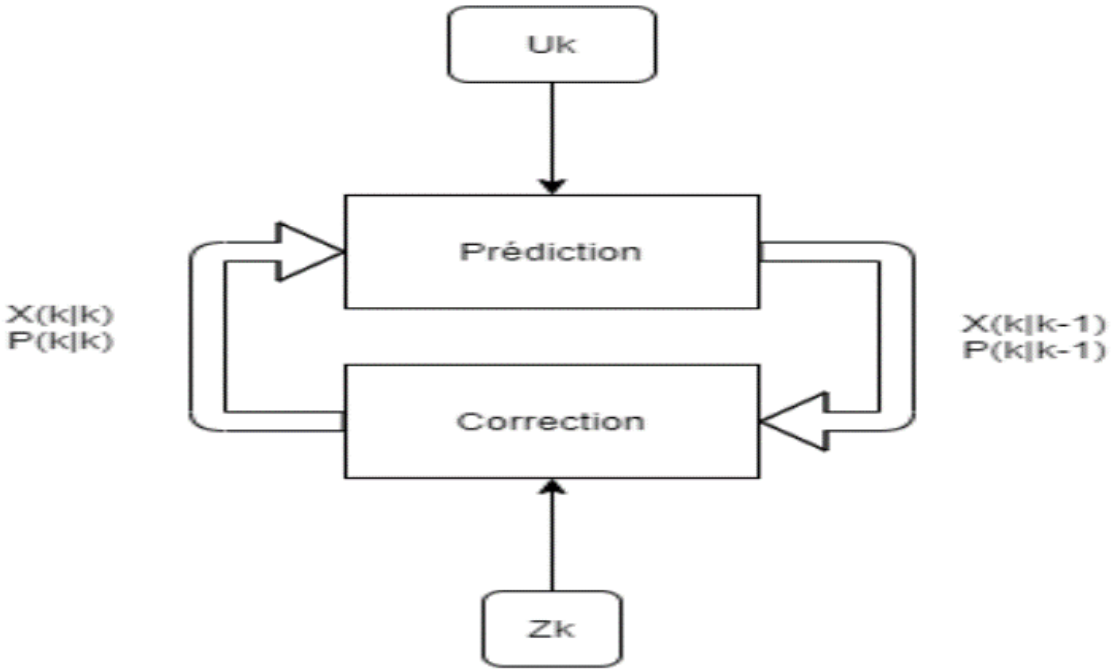

FIGURE 1 – Principe de filtre de Kalman

Avec :
— U_k : Matrice de commande
— Z_k : Matrice d'observation
— $X_{k/k-1}$: Prédiction de la position
— $X_{k/k}$: Position corrigée par les mesures

— $P_{k/k-1}$: Matrice de covariance prédite
— $P_{k/k}$: Matrice de covariance corrigée par les mesures

2.2 Développement des équations, estimateurs linéaires :

Soit les équations suivantes :

$$\begin{cases} x_k &= A_k x_{k-1} + B_{k-1} U_{k-1} + v_{k-1} \\ y_k &= C_k x_k + n_k \end{cases}$$

avec $v \in N(0, R_v)$ et $n \in N(0, R_n)$. La première équation est l'équation de l'évolution et la deuxième est celle de mesure.

L'algorithme ci-dessous illustre le filtre de Kalman :

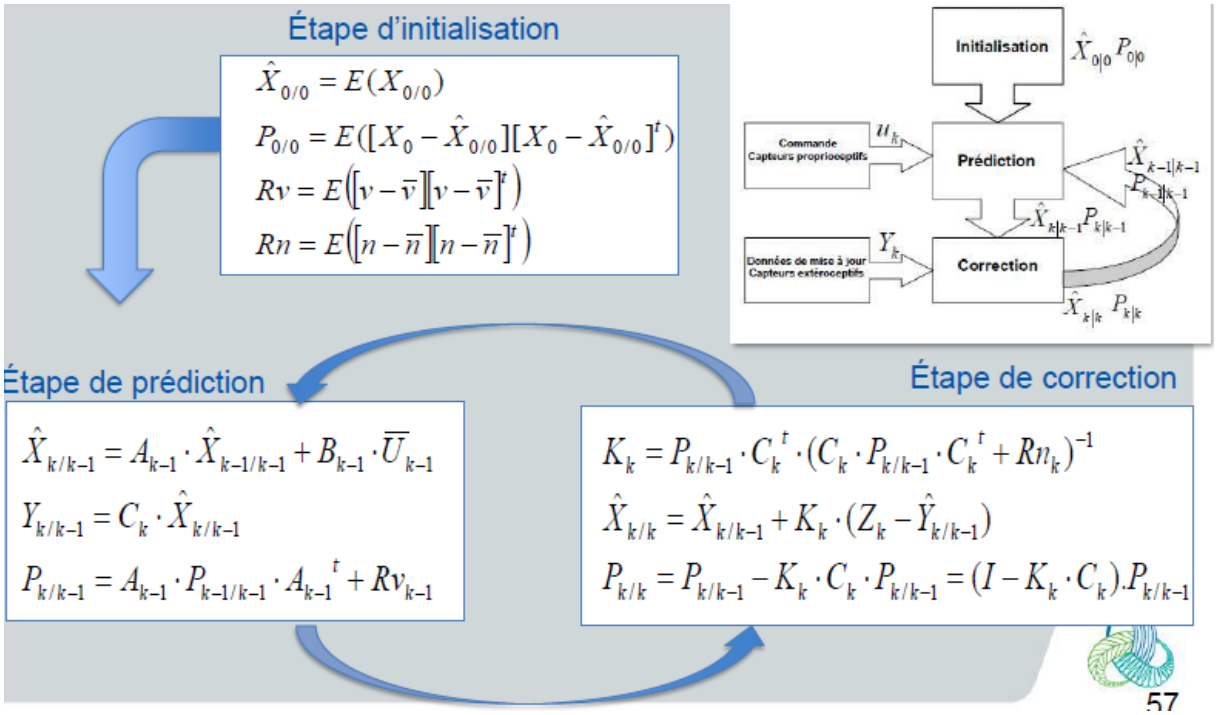

FIGURE 2 – Estimateur linéaire : filtre de Kalman

Cet algorithme est utilisé dans la cas linéaire. Cependant, Dans ce TP, nous abordons le cas où la fonction d'évolution f et la fonction de la correction g ne sont pas linéaires, d'où l'utilisation de filtre de Kalman EKF.

2.3 Développement des équations, estimateurs linéaires :

Comme les fonctions d'évolution et de correction ne sont pas linéaires, les étapes de filtre de Kalman étendu sont :

1. Étape de prédiction :
   D'abord, on calcule les matrices jacobiennes du systèmes :

$$\boxed{A_{k-1} = \nabla_X f(X, u_{k-1}, \bar{v})|_{X=\hat{X}_{k-1|k-1}}}$$

$$\boxed{D_v = \nabla_v f(\hat{X}_{k-1|k-1}, u_k, v)|_{v=\bar{v}}}$$

$$\boxed{B_{k-1} = \nabla_u f(\hat{X}_{k-1|k-1}, u, \bar{v})|_{u=u_{k-1}}}$$

Puis, Pour prendre en compte le bruit sur le processus on met à jour la fonction d'évolution :

$$\boxed{\hat{X}_{k|k-1} = f(\hat{X}_{k|k-1}, u_{k-1}, \bar{v})}$$

$$\boxed{P_{xk|k-1} = A_{k-1} P_{xk-1|k-1} A_{k-1}^t + D_v R_v D_v^t + B_{uk} R_\lambda B_{uk-1}^t}$$

2. Étape de la correction : On calcule les matrices Jacobiennes de l'observation :

$$\boxed{C_k = \nabla_x g(X, \bar{n} | X = \hat{X}_{k|k-1}}$$

$$\boxed{D_n = \nabla_n g(\hat{X}_{k|k-1}, \bar{n}) | n = \bar{n}}$$

Ensuite, les calculs de la correction :

$$\boxed{K_k = P_{xk|k-1} C_{xk}^t [C_{xk} P_{xk|k-1} C_{xk}^t + D_n R_n D_n^t]^{-1}}$$

$$\boxed{\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k [Z_k - g(\hat{X}_{k|k-1}, \bar{n})]}$$

$$\boxed{_{k/k} = [I - K_k C_{xk}] P_{k|k-1}}$$

En effet, l'algorithme de filtre de Kalman étendu utilise l'expansion d'ordre inférieur de la méthode de Taylor pour la formule de prédiction non linéaire, et atteint le calcul de la densité de probabilité du modèle par une méthode approximative. Ainsi, les formules A et B ci-dessus sont toutes des équations matricielles simplifiées.

3 Développement du filtrage de Kalman, estimateurs non linéaires :

L'évolution du système se fait selon le modèle de la bicyclette, où la fonction d'évolution f est défini de la façon suivante :

$$\begin{cases} x_{k+1} &= x_k + varS.cos(z_k + varEta/2)(cos(AngleRoue) \\ y_{k+1} &= y_k + varS.sin(z_k + varEta/2)(cos(AngleRoue) \\ z_{k+1} &= z_k + varEta \end{cases}$$

Le vecteur d'état est X_k telle que :

$$X_k = [x_k, y_k, z_k]^t$$

La matrice d'évolution de notre système :

$$A_k = \nabla_x f(X_k) = [\frac{\partial f(X_k)}{\partial x_k}, \frac{\partial f(X_k)}{\partial y_k}, \frac{\partial f(X_k)}{\partial z_k}]^t$$

$$A_k = \begin{pmatrix} 1 & 0 & -varS.cos(z_k + varEta/2).cos(AngleRoue) \\ 0 & 1 & varS.sin(z_k + varEta/2).cos(AngleRoue) \\ 0 & 1 & 0 \end{pmatrix}$$

Maintenant, nous calculons la matrice de la commande :

$$B_k = \nabla_u f(X_k)$$

$$B_k = \begin{pmatrix} cos(z_k + varEta/2).cos(Angleroue) & -varS/2.cos(z_k + varEta/2).cos(AngleRoue) \\ sin(z_k + varEta/2).cos(Angleroue) & varS/2.cos(z_k + varEta/2).cos(AngleRoue) \\ 0 & 1 \end{pmatrix}$$

Finalement, la matrice de mesure (avec et sans données de GPS) :

$$C_k = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$C_{kgps} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Passons à l'implémentation.

4 Mise en oeuvre du filtre sur données réelles :

Nous utilisons ces fonctions sur le jeu de données fournit dans ce TP.Dans un premier temps il sera testé sans dcorrection avec les données GPS afin d'avoir la dérive dans le temps.

4.1 Sans les données de GPS :

La figure ci-dessous illustre le comportement du système sans correction.

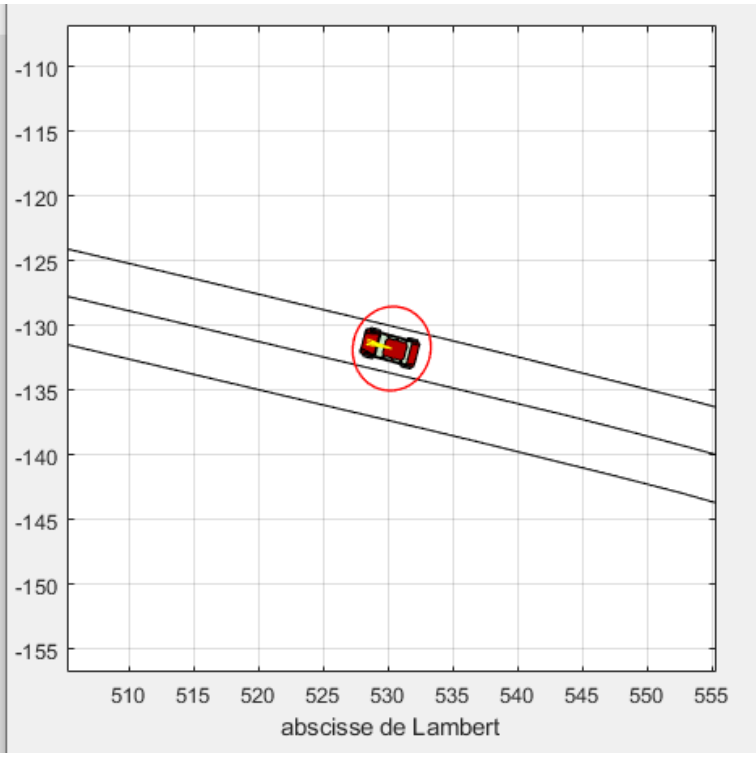

FIGURE 3 – Système sans correction

La figure ci-dessous illustre que la voiture s'éloigne de la piste au cours du temps. Donc, à l'instant initiale on remarque que la voiture est sortie carrément de la piste et elle n'a pas de données (GPS) pour corriger l'erreur qu'elle effectue.

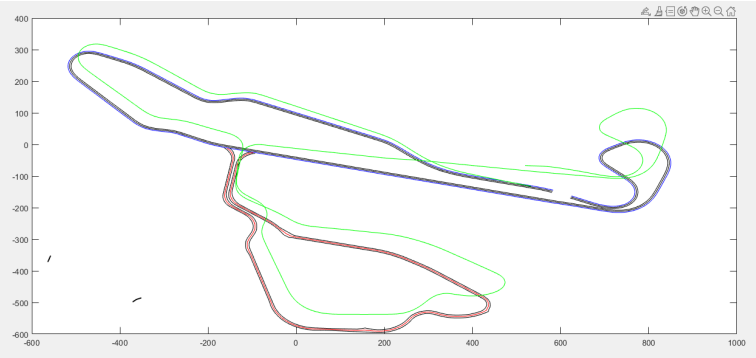

FIGURE 4 – Trajectoire de la voiture estimé sans données de GPS

Cela est tout à fait normal, car le filtre de Kalman n'exécute que la prédiction et non pas la correction.

4.2 Avec les données de GPS :

A ce niveau, nous ajoutons la partie de correction du filtre de Kalman. La figure ci-dessous illustre le comportement de la voiture ainsi l'ellipse d'incertitude après la correction(rouge).

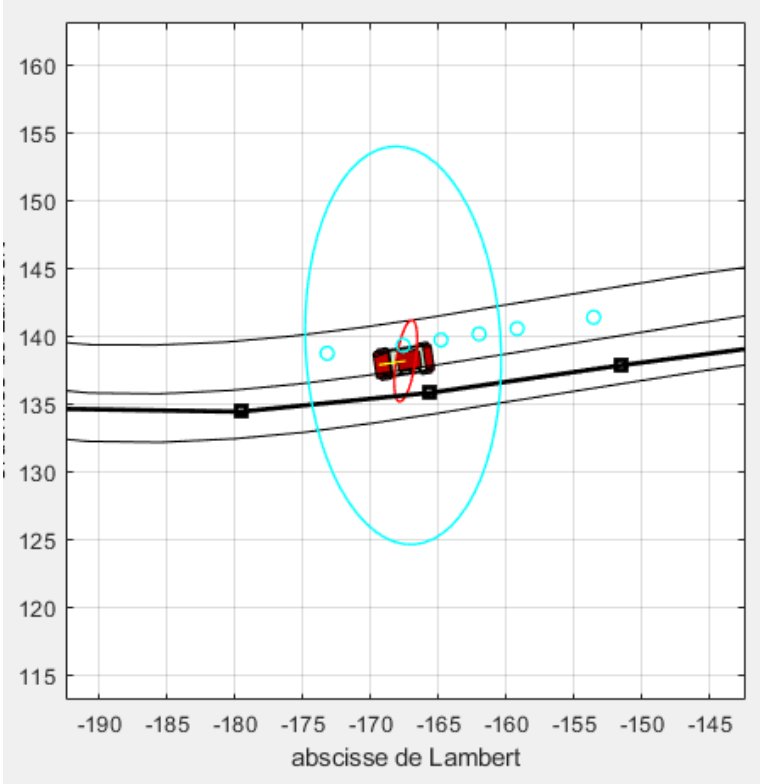

FIGURE 5 – Le comportement de la voiture après la correction

En effet, la voiture est localisée correctement après l'application de la correction, et les résultats sont améliorés car l'erreur est moins important sur la trajectoire finale de la voiture. L'aire de l'ellipse montre que la voiture a une meilleure intervalle de confiance que dans le cas précédent(ellipse en bleu ciel).

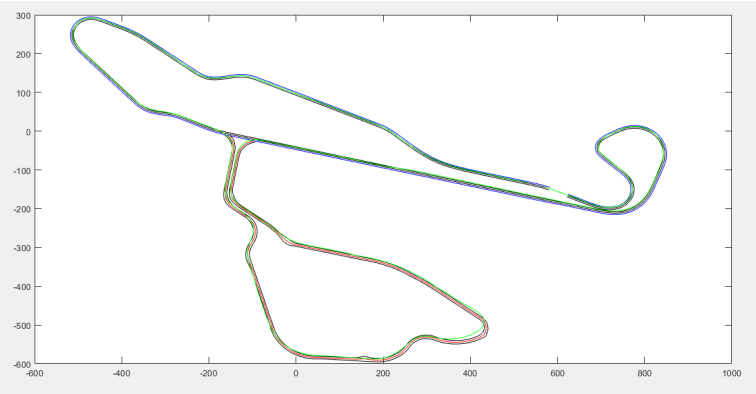

FIGURE 6 – Trajectoire de la voiture estimé avec données de GPS

5 Compréhension du fonctionnement :

5.1 Qualité de l'initialisation de l'état du système :

Afin d'étudier l'impact de la position initiale, nous allons le modifier pour voir l'effet. Tout d'abord, le système a été initialisé à partir d'un point situé sur la route. Ainsi, nous initialisons la position initiale en dehors de la piste et l'angle de direction initiale est choisi d'une manière aléatoire.

La figure ci-dessous montre cette initialisation. Après à un intervalle du temps et grâce à la correction GPS, l'estimation de la localisation du modèle est corrigé.

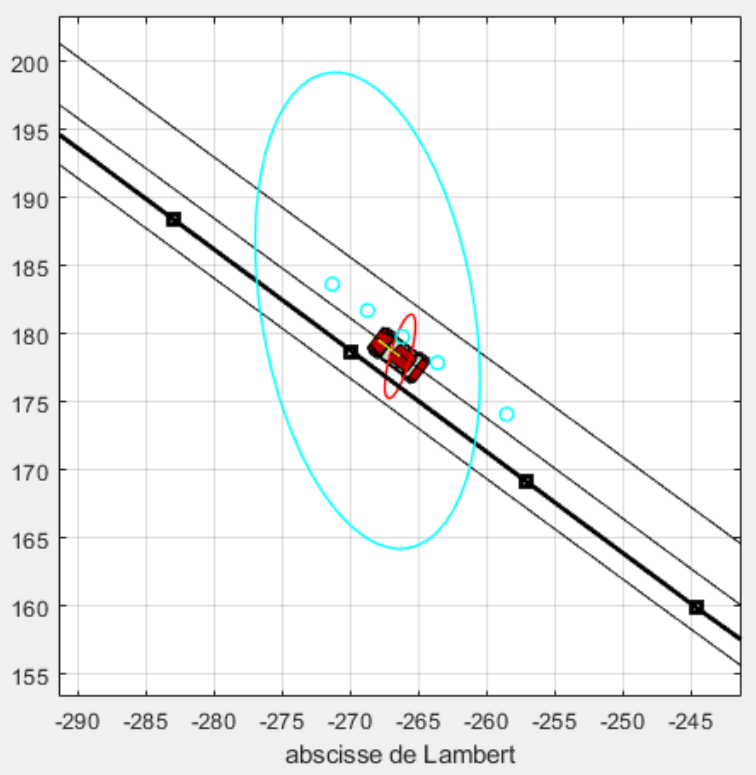

FIGURE 7 – Évolution de la simulation

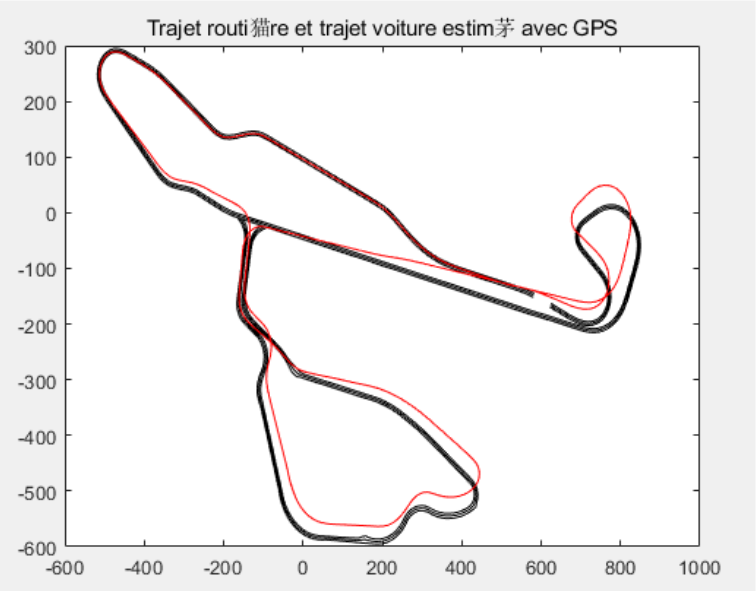

FIGURE 8 – Trajet routière et trajet estimé avec GPS (rouge)

On remarque que la trajectoire de la voiture se rejoint la piste après une certaine distance.

5.2 Qualité de l'initialisation de la matrice de covariance :

Afin d'étudier la qualité de l'initialisation de la matrice de covariance, nous allons changer les valeurs diagonales de la matrice P_e. Cette matrice est initialisée de la façon suivante : $P_e = diag([1, 1, 0.01])$. On remplace les deux premières valeurs par 100, c.a.d. que l'on a moins de confiance sur la prédiction sur la prédiction initiale.

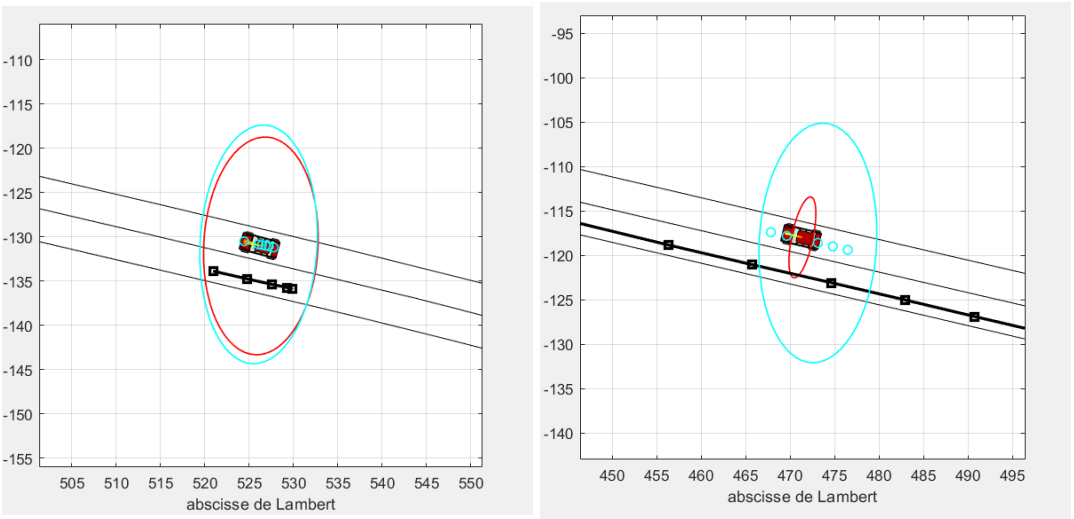

FIGURE 9 – Qualité de l'initialisation de la matrice covariance

Nous remarquons que au cours des itérations P_e converge vers les valeurs minimales alors que la nouvelle initialisation de P_e a augmenté l'incertitude de prédiction.

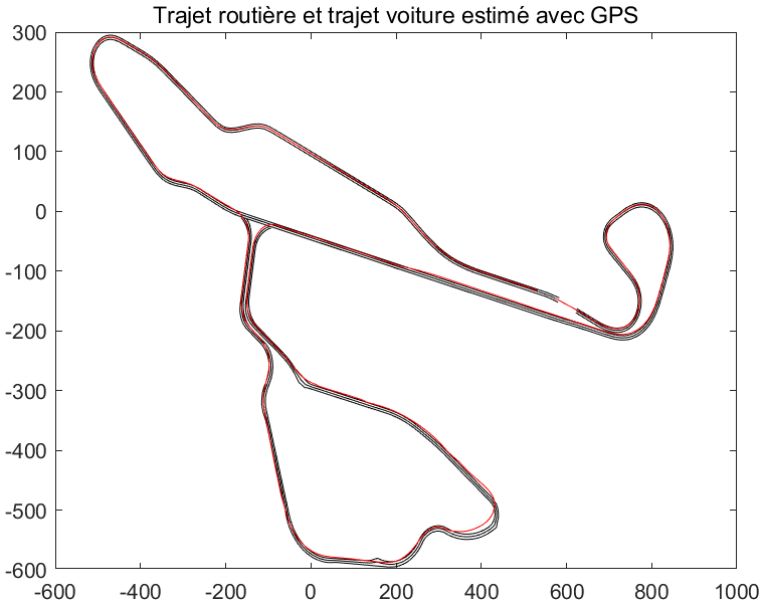

FIGURE 10 – La piste routière(noir) et le trajet de voiture estimée avec GPS

La trajectoire de la voiture est semblable à celle de l'étape précédente, cela démontre que P_e détermine la vitesse de convergence du système mais n'affecte pas le résultat d'estimation.

5.3 Impact de la matrice de bruit du système :

Dans cette étape, on va étudier l'influence de Bruits du système sur le filtre.Dans le programme, on va modifier «BruitsSystemeCapteurs »que manipule le bruit du capteur. Les deux premiers 'SigmaCodeur"SignaGyro' items correspondent à l'entrée du prédicateur. Si on augmente ou diminue ces deux options, on peut obtenir des différents résultats.
On prend $SigmaCodeur = 0.01$ et $SigmaGyro = 0.0015$ Premièrement, on annule le bruit de système, c.a.d. qu'il n'y a pas de variance dans la prédiction et la probabilité de l'équation de prédiction est proche de la fonction de réponse impulsionnelle.Ainsi, le filtre se base uniquement sur le prédicteur.Ainsi, nous remarquons dans la figure ci-dessous une erreur dans l'itinéraire de la voiture et le résultat est semblable à celui de la navigation sans GPS.

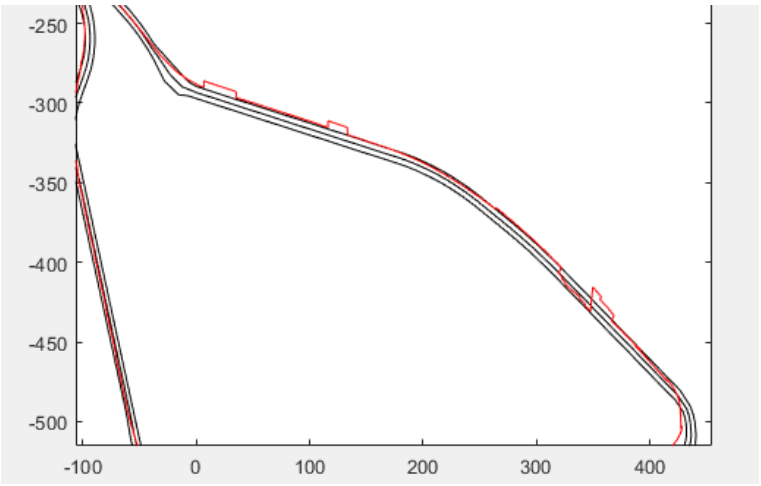

FIGURE 11 – Trajetoire estimée avec $Q_{system} = diag([100, 100, 15])$

Deuxièment, on multiplie l'erreur du système 10000 fois. Ce qui implique l'amplification de l'erreur du prédicteur.Ainsi, le filtre prend davantage le résultat du GPS.

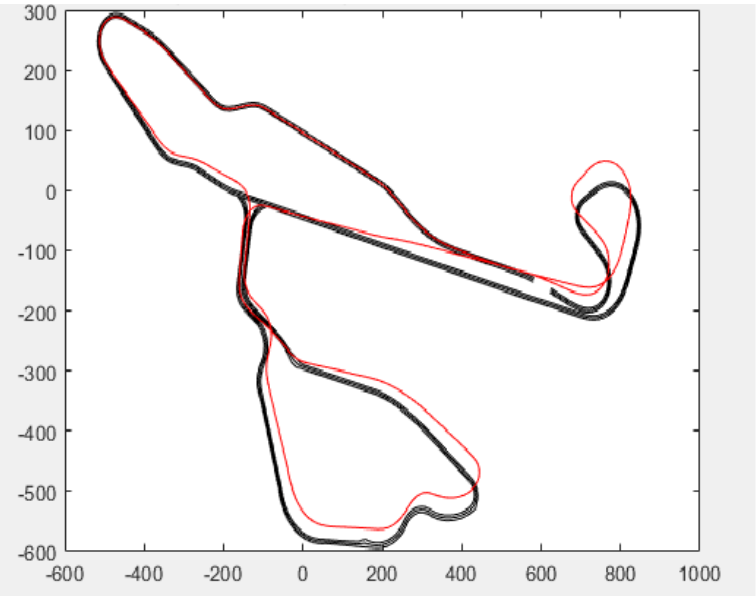

FIGURE 12 – Trajectoire estimée avec $Q_{system} = diag[0, 0, 0]$

6 Conclusion et perspective :

Le filtre de Kalman étendu permet donc de linéariser localement des systèmes non linéaire. Ceci assure donc la convergence locale de l'erreur, mais non celle globale. Un surcoût de calcul est constaté par rapport au filtre de Kalman classique. En effet, outre les opérations non linéaires introduit dans les équations d'états et de transitions, il faut recalculer à chaque étape les Jacobiennes de ces équations.

7 Annexe :

EstimationKalman

```
%*************************************************************************************
% EstimationKalmanNL
% Module de mise à jour et de correction du vecteur d'état
%
% Fichier           : $RCSfile: EstimationKalmanNL.m,v $
% Auteur            : Dominique Gruyer
%
% Auteurs :
%  Dominique Gruyer:          1) Developpement du code principal
%                                                  2) Calcul des étapes suivantes:
%                                Calcul du résidu
%                                                                  Calcul du gain
%                                                                  Calcul de la nouvelle est
%                                                                  Calcul de la nouvelle mat
%*************************************************************************************
function [X_E,P_E,Sk]=EstimationKalmanNL(P_P,Hk,X_P,Y_GPS,Q_GPS);

% P_P: Matrice de variance covariance sur la prediction
% Hk: Matrice de mesure. Cette matrice est une matrice de transfert entre la mesure et le
% X_P: vecteur d'état prédit
% Y_GPS: donnée GPS, (X,Y)
% Q_GPS: bruit de GPS
%==================================
% ETAPE D'ESTIMATION : instant k/k
%==================================
Sk = Hk * P_P * Hk' + Q_GPS; % Matrice de covariance de l'innovation
Kk = P_P * Hk' *inv(Sk); % Calcul du gain de kalman optimal
X_E = X_P + Kk * (Y_GPS - X_P(1:2)); % Nouvelle de l'estimation de l'etat
P_E = (eye(3) - Kk*Hk)*P_P*( eye(3) - Kk*Hk)'+Kk*Q_GPS*Kk'; % Nouvelle matrice de covarian
%=============================================================
```

ModeleEvolutionGPSNL

```
%*************************************************************************************
% ModeleEvolutionGPSNL
% Module de calcul des différentes matrices et structures utilisées par l'EKF
%
% Fichier           : $RCSfile: ModeleEvolutionGPSNL.m,v $
% Auteur            : Dominique Gruyer
%
% Auteurs :
%  Dominique Gruyer:          1) Developpement du code principal
%                                                  2) Calcul des étapes suivantes:
%                               matrice jacobienne du modele d'évolution: Fk = dfk/dxk
%                               matrice jacobienne du modele d'évolution: Bk = dfk/duk
%                               matrice de mesure: Hk
%                               matrice de bruit d'état et de mesure
%*************************************************************************************/
function [Fk,Hk,Bk,Q_systeme,Q_GPS]=ModeleEvolutionGPSNL(var_S,var_teta_c,teta_c,VarSystem
```

```
%=================================================================
% DEFINITION DU MODELE D'EVOLUTION ET DE LA PREMIERE PREDICTION
%=================================================================
     % calcul de la premiere position prédite du véhicule
%=================================================
R = 0.3;                                              % rayon de la roue
E = 1;                                               % longueur de l'essieu
pas_codeur=.1954;

     % matrice jacobienne du modele d'évolution: Fk = dfk/dxk
     %=========================================================
 Fk = [1 0 -var_S*sin(teta_c + (var_teta_c/2))*cos(Psi);...
  0 1 var_S*cos(teta_c + (var_teta_c/2))*cos(Psi); 0 0 1];

     % matrice jacobienne du modele d'évolution: Bk = dfk/duk
     %=========================================================
 Bk = [cos(teta_c + (var_teta_c/2))*cos(Psi) (-var_S/2)*sin(teta_c + (var_teta_c/2))*co
 sin(teta_c + (var_teta_c/2))*cos(Psi) (var_S/2)*cos(teta_c + (var_teta_c/2))*cos(Psi)

 % matrice de mesure: Hk
     %=========================================================
  Hk = [ 1 0 0 ;0 1 0 ];

%=========================================================
% CALCUL DE LA MATRICE DE BRUIT SUR L'ENTREE DU SYSTEME
%=========================================================
 Q_systeme=[ VarSysteme(1)^2 0 0;0 VarSysteme(2)^2 0;0 0 VarSysteme(3)^2];

%=================================================================
%                          Construction de la matrice de bruit du GPS
% Cette matrice est construite à partir des sigmas a et b et de la
% rotation fournie par le GPS dans le cas où l'indicateur de validité
% d'une mesure est à 1
%=================================================================
%  Mode GPS, 1 gps naturel, 2 différentiel EGNOS, 0 masquage
%  Valide, 0 et 1
%=================================================================

 Q_GPS = InitBruitGPS( Valide, ModeGPS, SigmaA, SigmaB, Phi);
```

ModeleEvolutionNL.m

```
%*************************************************************************************
% ModeleEvolutionNL
% Module de calcul des différentes matrices et structures utilisées par l'EKF
%
% Fichier           : $RCSfile: ModeleEvolutionNL.m,v $
% Auteur            : Dominique Gruyer
%
% Auteurs :
%  Dominique Gruyer:          1) Developpement du code principal
%                                                2) Calcul des étapes suivantes:
%                                matrice jacobienne du modele d'évolution: Fk = dfk/dxk
%                                matrice jacobienne du modele d'évolution: Bk = dfk/duk
%                                matrice de mesure: Hk
%                                matrice de bruit d'état et de mesure
%*************************************************************************************/
function [Fk,Hk,Bk,Q_systeme]=ModeleEvolutionNL(var_S,var_teta_c,teta_c,VarSysteme,Psi);
```

```
%================================================================
% DEFINITION DU MODELE D'EVOLUTION ET DE LA PREMIERE PREDICTION
%================================================================
     % calcul de la premiere position prédite du véhicule
%=====================================================
R = 0.3;                                                % rayon de la roue
E = 1;                                                 % longueur de l'essieu
pas_codeur=.1954;

     % matrice jacobienne du modele d'évolution: Fk = dfk/dxk
     %=========================================================
 Fk = [1 0 -var_S*sin(teta_c + (var_teta_c/2))*cos(Psi);
  0 1 var_S*cos(teta_c + (var_teta_c/2))*cos(Psi);
  0 0 1];

     % matrice jacobienne du modele d'évolution: Bk = dfk/duk
     %=========================================================
 Bk = [cos(teta_c + (var_teta_c/2))*cos(Psi) (-var_S/2)*sin(teta_c + (var_teta_c/2))*co
  sin(teta_c + (var_teta_c/2))*cos(Psi) (var_S/2)*cos(teta_c + (var_teta_c/2))*cos(Psi)

     % matrice de mesure: Hk
     %=========================================================
 Hk = [ 1 0 0; 0 1 0 ];

%=========================================================
% CALCUL DE LA MATRICE DE BRUIT SUR L'ENTREE DU SYSTEME
%=========================================================
   Q_systeme=[ VarSysteme(1)^2 0 0;0 VarSysteme(2)^2 0;0 0 VarSysteme(3)^2];
```

PredictionKalmanNL

```
%*************************************************************************************
% PredictionKalmanNL
% Module de prédiction de l'état du système
%
% Fichier           : $RCSfile: PredictionKalmanNL.m,v $
% Auteur            : Dominique Gruyer
%
% Auteurs :
%  Dominique Gruyer:          1) Developpement du code principal
%                                                  2) Calcul des étapes suivantes:
%                                Prédiction de l'état du système
%                                                              Calcul de la nouvelle mat
%*************************************************************************************
function [X_P,P_P]=PredictionKalmanNL(var_S,var_teta,Fk,X_E,P_E,Q_systeme,AngleRoue);


% calcul du vecteur d'état prédit X(k/k-1)
%=================================================
X_P  =   [        (X_E(1) + var_S*cos(X_E(3) + (var_teta/2))*cos(AngleRoue));
                (X_E(2) + var_S*sin(X_E(3) + (var_teta/2))*cos(AngleRoue));
                (X_E(3) + var_teta)];
```

```
% calcul de la variance de la prediction P(k/k-1)
%=================================================
P_P  =   Fk*P_E*Fk'+ Q_systeme;

end
```

Trajectoire

```
[XRef,YRef] = ConverterLL2XY(((48+47/60+11.07122/3600)*pi)/180, ((2+5/60+32.15779/3600)*pi
ZRef = 216.937;
ReferenceGPS = [ XRef YRef ZRef];
[pisteRoute, pisteValdOr] = ChargementPistesReferences(ReferenceGPS, 1);
load('saveData\sansResultatGPSINSTopoEKF3.mat');
count = size(DonneesLocalisationEKF,2);
X1_E = zeros(count, 3);
for i = 1:count
    x_E = DonneesLocalisationEKF(i).X_E(1);
    y_E = DonneesLocalisationEKF(i).X_E(2);
    z_E = DonneesLocalisationEKF(i).X_E(3);
    X1_E(i, :) = [x_E, y_E, z_E];
end

plot(pisteRoute.xd, pisteRoute.yd, 'black'); hold on;
plot(pisteRoute.xc, pisteRoute.yc, 'b'); hold on;
plot(pisteRoute.xg, pisteRoute.yg, 'black'); hold on;
plot(pisteValdOr.xd, pisteValdOr.yd, 'black'); hold on;
plot(pisteValdOr.xc, pisteValdOr.yc, 'r'); hold on;
plot(pisteValdOr.xg, pisteValdOr.yg, 'black'); hold on;
plot(X1_E(:,1), X1_E(:,2), 'r');

% % chargement de la matrice de position estimé sans les donée GPS :
%
% % load('ResultatFiltres\ResultatGPSINSTopoEKF3_1.mat');
% count = size(DonneesLocalisationEKF,2);
%
% % Extraction des coordonnées de la matrice de structure :
%
% X2_E = zeros(count, 3);
%
% for i = 1:count
%     x_E = DonneesLocalisationEKF(i).X_E(1);
%     y_E = DonneesLocalisationEKF(i).X_E(2);
%     z_E = DonneesLocalisationEKF(i).X_E(3);
%     X2_E(i, :) = [x_E, y_E, z_E];
% end
% figure;
%
% % tracer du trajet : carte routiere et cal d'Or :
%
% plot(pisteRoute.xd, pisteRoute.yd, 'g');
% hold on
% plot(pisteRoute.xc, pisteRoute.yc, 'r');
% hold on
% plot(pisteRoute.xg, pisteRoute.yg, 'g');
% hold on
% plot(pisteValdOr.xd,   pisteValdOr.yd, 'g');
% hold on
% plot(pisteValdOr.xc, pisteValdOr.yc, 'r');
% hold on
% plot(pisteValdOr.xg, pisteValdOr.yg, 'g');
% hold on
```

```
%
% % position de l'estimateur EKF sans GPS :
% plot(X2_E(:,1), X2_E(:,2), 'b');
```