Porsuite de cible filtre de Kalman Étendu

Arthur Coelho Ruback
Matheus Santos Sano
Leonardo Greco Picoli

Estimation et identification statistique MA201

Palaiseau, France

30 octobre 2022

Table des matières

1 Exercice 1 **2**
1.1 . . . 2
1.2 . . . 3
1.3 . . . 3
1.4 . . . 4

2 Exercice 2 **5**
2.1 . . . 5
2.2 . . . 6
2.3 . . . 7

3 Exercice 3 **8**
3.1 . . . 8
3.2 . . . 9
3.3 . . . 10
3.4 . . . 11
3.5 . . . 13

4 Exercice 4 **13**

5 Exercice 5 **13**
5.1 . . . 13
5.2 . . . 14
5.3 . . . 14

6 Exercise 6 **15**

7 Anexes **16**
7.1 EX2 . . . 16
7.2 EX3 . . . 19
7.3 EX5 . . . 22

1 Exercice 1

1.1

Objectif :
Montrer qu'un processus $a(k) = (a_x(k), a_y(k))$ peut être mis sous la forme $a_x(k+1) = \beta a_x(k) + w_x(k)$ et $a_y(k+1) = \beta a_y(k) + w_y(k)$ où $w_x(k)$ et $w_y(k)$ sont représentations des bruits blancs de variance σ_w^2.
Développement :
Le composant successive est représenté par :

$$r(l) = E[a_x(k).a_x(k+l)] = \sigma_a^2 exp(-\mu.l) \tag{1}$$

Pour obtenir β et σ_w^2, on développe :

$$a_x(k+1) = \beta a_x(k) + w_x(k)$$
$$E[(a_x(k+1))^2] = E[(\beta a_x(k) + w_x(k))^2]$$

Comme $E[a_x(k+1).a_x(k+1)] = E[a_x(k).a_x(k)]$ et $E[a_x(k).a_x(k)] = \sigma_a^2.exp(-\mu.0) = \sigma_a^2$:

$$E[(a_x(k+1))^2] = E[(\beta a_x(k) + w_x(k))^2]$$
$$\sigma_a^2 = E[\beta^2 (a_x(k))^2 + 2\beta a_x(k).w_x(k)] + (w_x(k))^2]$$
$$\sigma_a^2 = \beta^2 E[(a_x(k))^2] + 2\beta E[a_x(k).w_x(k)] + E[(w_x(k))^2] \tag{2}$$

Comme $w_x(k)$ est une composant d'un vecteur bruit blanc centré, $E[w_x(k)] = 0$. Par ailleurs, on considère que $a_x(k)$ et $w_x(k)$ sont indépendants. Donc :

$$E[a_x(k).w_x(k)] = E[a_x(k)].E[w_x(k)] = 0$$
$$et$$
$$Var[w(k)] = E[(w_x(k))^2] - (E[w_x(k)])^2 = \sigma_w^2$$
$$E[(w_x(k))^2] = \sigma_w^2$$

Après ces considérations, l'équation (2) peut être reprise et, finalement, σ_w est trouvé :

$$\sigma_a^2 = \beta^2 E[(a_x(k))^2] + 2\beta E[a_x(k).w_x(k)] + E[(w_x(k))^2]$$
$$\sigma_a^2 = \beta^2.\sigma_+\sigma_w^2$$
$$\sigma_w^2 = \sigma_a^2(1 - \beta^2)$$

Pour trouver β on utilise l'équation (1) pour $l = 1$:

$$r(1) = E[a_x(k).a_x(k+1)]$$
$$r(1) = E[a_x(k).(\beta a_x(k) + w_x(k))] = E[\beta a_x(k)^2 + a_x(k).w_x(k))]$$
$$r(1) = \beta\sigma_a^2$$

Comme $r(1) = \sigma_a^2 exp(-\mu)$, on trouve β :

$$\beta\sigma_a^2 = \sigma_a^2 exp(-\mu)$$
$$\beta = exp(-\mu)$$
$$\sigma_w^2 = \sigma_a^2(1 - exp(-2\mu))$$

Conclusion :

$$\beta = exp(-\mu)$$
$$\sigma_w^2 = \sigma_a^2(1 - exp(-2\mu))$$

1.2

Objectif :
Modéliser le système donné par une équation d'état ayant comme état la position, vitesse et accélération sur les directions X et Y.
Développement :
On utilise le principe de l'intégration pour lier les états dans une même direction contre le principe de la dérivation lorsque celle-la a moins de bruit.
Pour bien modéliser l'accélération, on a séparé l'accélération par rapport au axe y en une accélération constante, pesanteur, une accélération liée au processus stochastique ($\ddot{y}_e(k+1)$) et une accélération résultant ($\ddot{y}(k+1)$).

$$\begin{aligned} \ddot{y}(k) &= \ddot{y}_e(k) - g \\ \ddot{y}_e(k+1) &= \beta \ddot{y}_e(k) + w_y(k) \end{aligned} \tag{3}$$

En substituant $\ddot{y}_e(k)$ dans l´équation 7, on trouve :

$$\ddot{y}(k+1) = \beta \ddot{y}(k) + w_y(k) + g(\beta - 1)$$

Vu qu'on travaille sur un système discret :

$$\begin{cases} x_{k+1} = x_k + T\dot{x}_k + \frac{T^2}{2}\ddot{x}_k \\ \dot{x}_{k+1} = \dot{x}_k + T\ddot{x}_k \\ \ddot{x}_{k+1} = \beta \ddot{x}_k + w_{xk} \end{cases} \qquad \begin{cases} y_{k+1} = y_k + T\dot{y}_k + \frac{T^2}{2}\ddot{y}_k \\ \dot{y}_{k+1} = \dot{y}_k + T\ddot{y}_k \\ \ddot{y}_{k+1} = \beta \ddot{y}_k + w_{yk} + g(\beta - 1) \end{cases}$$

L'équation de la dynamique de l'état est défini sous la forme $X_{k+1} = F_k X_k + G_k u_k$, où :

$$F_k = \begin{pmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & 0 & 0 & \beta \end{pmatrix} \quad X(k) = \begin{pmatrix} x(kT) \\ y(kT) \\ \dot{x}(kT) \\ \dot{y}(kT) \\ \ddot{x}(kT) \\ \ddot{y}(kT) \end{pmatrix} \quad G(k) = I_{6x6} \quad u(k) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ w_x(k) \\ w_y(k) + g(\beta - 1) \end{pmatrix}$$

1.3

Objectif :
Exprimer le vecteur de mesures (D_k, α_k) en fonction des positions de la cible $(x(t), y(t))$ et du dispositif d'observation (x_0, y_0).
Développement :
Considérant la cible et le radar comme deux points sur un plan cartésien, la distance entre eux (D_k) et l'angle entre la droite cible-radar et l'horizontale (α_k) s'expriment comme :

$$\begin{aligned} D_k &= \sqrt{(x_k - x_0)^2 + (y_k - y_0)^2} + n_D(k) \\ \alpha_k &= \arctan((y_k - y_0)/(x_k - x_0)) + n_s(k) \end{aligned}$$

Il faut en souligner que la distance D_k et l'angle α_k sont des mesures bruitées, dont les bruits sont représentés, respectivement, par $n_D(k)$ et $n_s(k)$.
Conclusion :

$$D_k = \sqrt{(x_k - x_0)^2 + (y_k - y_0)^2} + n_D(k) \tag{4}$$

$$\alpha_k = \arctan((y_k - y_0)/(x_k - x_0)) + n_s(k) \tag{5}$$

1.4

Objectif :
Montrer que, a partir de pseudo mesures, nous ne pouvons pas écrire [x(t), y(t)] comme un somme d'un fonction de l'état plus un bruit. Ça veut dire que l'équation :

$$[x_k, y_k] = [f_x(k) + b_x(k), f_y(k) + b_y(k)] \tag{6}$$

ce n'est pas possible à écrire en sachant que D_k et α_k sont bruités.
Développement :
Si D_k et α_k sont bruités, alors à partir de (5) et (6) on peut écrire :

$$D_k = \sqrt{x_k^2 + y_k^2} + B_d$$
$$\alpha_k = \arctan(y_k/x_k) + B_\alpha$$

où B_d et B_α sont les bruits associés aux variables D_k et α_k, respectivement. Après, on a que :

$$\tilde{x} = D_k \cos\alpha_k$$
$$\tilde{y} = D_k \sin\alpha_k$$
$$\tilde{x} = (\sqrt{x_k^2 + y_k^2} + B_d)\cos(\arctan(y_k/x_k) + B_\alpha)$$
$$\tilde{y} = (\sqrt{x_k^2 + y_k^2} + B_d)\sin(\arctan(y_k/x_k) + B_\alpha)$$

Si on développe, on a que :

$$\tilde{x} = x_k\cos(B_\alpha) - y_k\sin(B_\alpha) + \frac{B_d}{\sqrt{x_k^2 + y_k^2}}(x_k\cos(B_\alpha) - y_k\sin(B_\alpha))$$
$$\tilde{y} = y_k\cos(B_\alpha) + x_k\sin(B_\alpha) + \frac{B_d}{\sqrt{x_k^2 + y_k^2}}(y_k\cos(B_\alpha) + x_k\sin(B_\alpha))$$

Enfin, on peut voir que les équations représentent un système non linéaire qui dépend des bruits de distance et sinus ou cosinus du bruit de l'angle, donc les pseudo mesures ne peuvent pas être modélisées sous la forme d'un somme d'un fonction et un bruit gaussien.

2 Exercice 2

2.1

Objectif :
Écrire les équations du filtre de Kalman simple correspondant aux équations dynamiques de l'état et l'équation d'observation correspondant aux pseudo-mesures.
Développement :
Le problème est modélisé par le système d'équations suivantes :

$$\begin{cases} x_{k+1} = F_k x_k + G_k u_k \\ z_k = H x_k + v_k \end{cases}$$

où x_k représente l'état du système ; z_k représente le vecteur de mesures. Dans ce cas ici, on mesure la position $\tilde{x}$ et $\tilde{y}$, correspondants au pseudo-mesures. ; u_k représente le bruit de modélisation qui a été déjà définit ; et v_k représente le bruit de mesure et suit un loi normale $\sim \mathcal{N}(0, V)$.
Les équations de filtre de Kalman sont séparés en 2 parties : prédiction et correction.
Prédiction :

$$\begin{cases} x_{k+1} = F_k x_k \\ P_{k+1} = F_k P_k F_k^T + W_k \end{cases}$$

où W_k c'est la matrice qui représente le bruit de modélisation ; P_k c'est la matrice de covariance avec les paramètres arbitrairement grands :

$$F_k = \begin{pmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & 0 & 0 & \beta \end{pmatrix} \qquad P_k = \gamma I_{6x6} \qquad W_k = T \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_w^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_w^2 \end{pmatrix}$$

où γ c'est une constante arbitrairemente grande.
Correction - Calcul du gain de Kalman :

$$K_{k+1} = P_{k+1} H_{k+1}^T (H_{k+1} P_{k+1} H_{k+1}^T + V_{k+1})^{-1}$$

Correction - Mise à jour :

$$\begin{cases} x_{k+1} = x_{k+1} + K_{k+1}(z_{k+1} - H_{k+1} x_{k+1}) \\ P_{k+1} = (I - K_{k+1} H_{k+1}) P_{k+1} \end{cases}$$

Le problème ici c'est que, pour la matrice de covariance V, on ne peut pas utiliser les variances du bruit blancs associés aux paramètres D_k et α_k, i.e. on ne peut pas supposer que :

$$V_k = \begin{pmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_s^2 \end{pmatrix}$$

lorsqu'on calcule les pseudo-mesures, qui sont $\tilde{x}$ et $\tilde{y}$, on doit linéariser cette matrice pour contenir une approximation des vraies variances associés aux pseudo-mesures, par exemple, $\sigma_{\tilde{x}}^2$ et $\sigma_{\tilde{y}}^2$. L'exercice 2.2 donne un possible solution pour cette problème.

2.2

Objectif :
Suggérer un moyen de déterminer les matrices de covariances associées aux pseudo-mesures
Développement :
Pour déterminer les matrices de covariances associées aux pseudo-mesures, on calcule la matrice jacobienne associée aux équations des pseudo-mesures :

$$\begin{cases} \tilde{x} = D_k \cos \alpha_k \\ \tilde{y} = D_k \sin \alpha_k \end{cases}$$

et pour la matrice jacobienne :

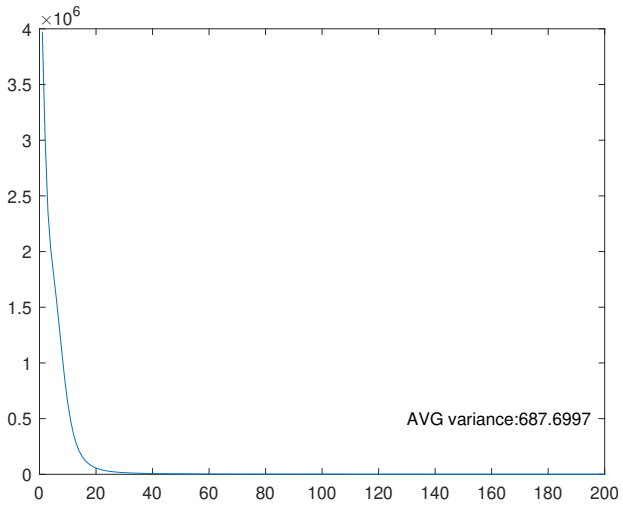
$$J = \begin{pmatrix} \frac{\partial \tilde{x}}{\partial D_k} & \frac{\partial \tilde{x}}{\partial \alpha_k} \\ \frac{\partial \tilde{y}}{\partial D_k} & \frac{\partial \tilde{y}}{\partial \alpha_k} \end{pmatrix} = \begin{pmatrix} \cos \alpha_k & -D_k \sin \alpha_k \\ \sin \alpha_k & D_k \cos \alpha_k \end{pmatrix}$$

Pour finir, pour calculer la nouvelle matrice de covariance V_n, on fait la multiplication de matrices pour linéariser :

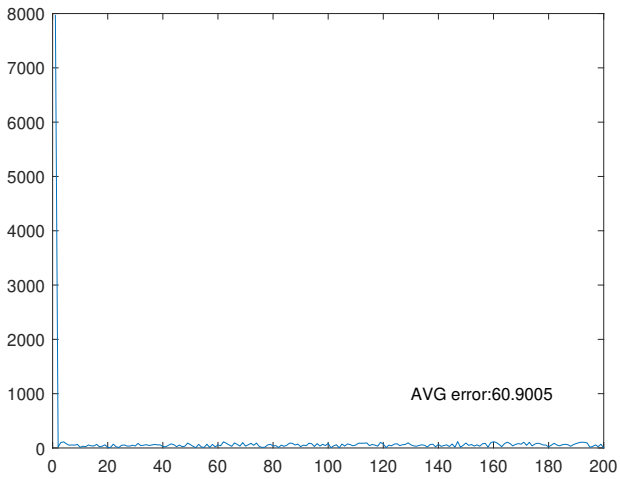
$$V_n = JVJ^T = \begin{pmatrix} \cos \alpha_k & -D_k \sin \alpha_k \\ \sin \alpha_k & D_k \cos \alpha_k \end{pmatrix} \begin{pmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_s^2 \end{pmatrix} \begin{pmatrix} \cos \alpha_k & \sin \alpha_k \\ -D_k \sin \alpha_k & D_k \cos \alpha_k \end{pmatrix}$$

2.3

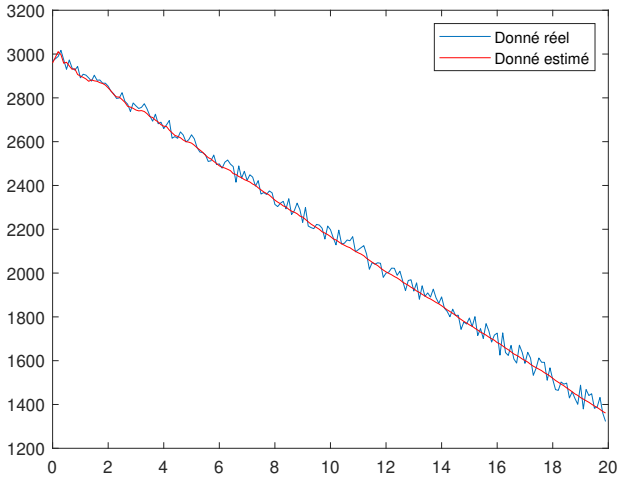
Objectif :
Implémenter l'algorithme résultant sous Matlab
Développement : L'implémentassion du code est compris dans l'annexe 7.1.
Conclusion : On voit que avec les deux jeux de donnés le système a été capable de fournir une réponse cohérent avec une faible variance et un erreur aussi faible.
On se voit aussi que le comportement du filtre a été meilleur pour les donnés sans accélération vu que c'est un mouvement plus simple.



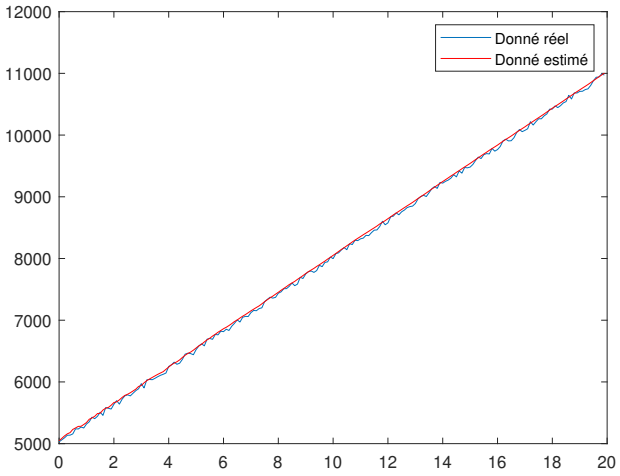
(a) Trace de la matrice de variance



(b) Erreur quadratique

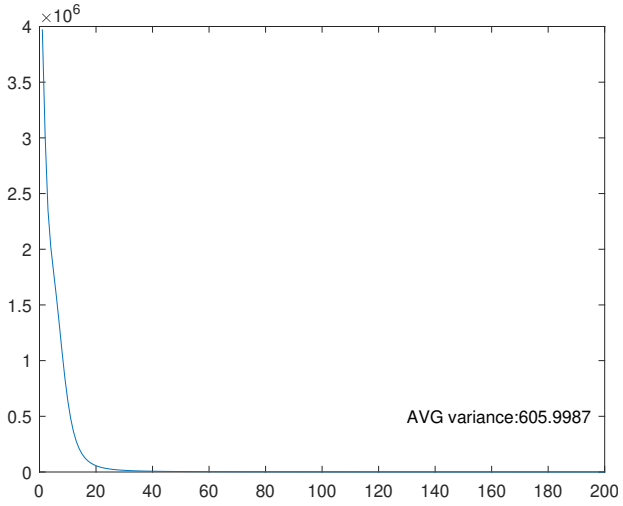


(c) Position X

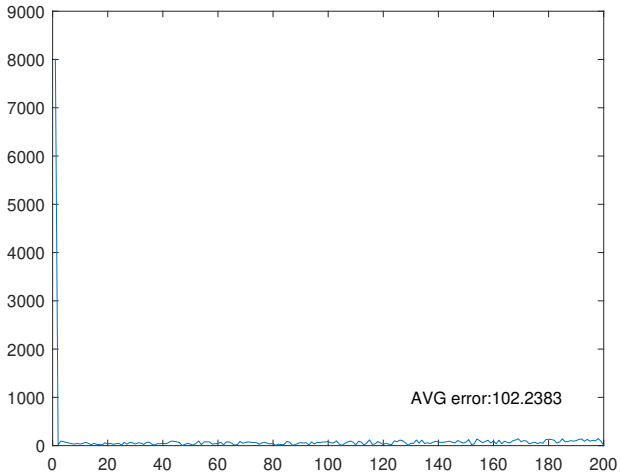


(d) Position Y

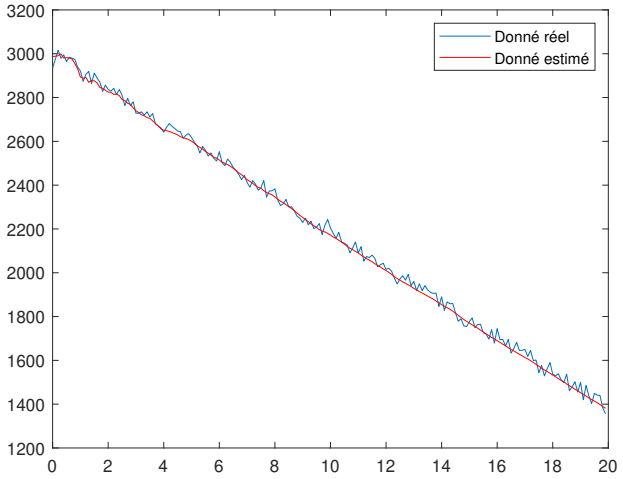
FIGURE 1 – Résultats Filtre de Kalman Simple avec de jeu de donnés 1.



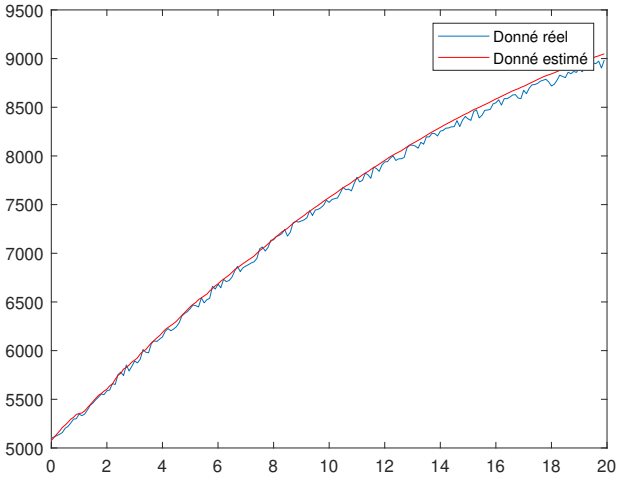
(a) Trace de la matrice de variance



(b) Erreur quadratique



(c) Position X



(d) Position Y

FIGURE 2 – Résultats Filtre de Kalman Simple avec de jeu de donnés 2.

3 Exercice 3

3.1

Objectif :
Écrire l'équation d'observation linéarisée autour de $\hat{X}_{k|k-1}$ sous la forme $z_k = H_k X(k) + b(k)$. Vérifier si $b(k)$ est centré.
Développement :
Soit une équation de la tangente au tour de x_0 :

$$z_0 = f'(x_0).(x - x_0) + f(x_0)$$

Pour trouver l'équation d'observation sous la forme

$$z_k = H_k x_k + b_k \tag{7}$$

il faut linéariser cette équation d'observation au tour de $x_0 = x_{k|k-1}$ et $y_0 = y_{k|k-1}$ par l'équation tangente.

Pour trouver la matrice d'observation H_k on calcule la jacobienne au tour de x_0 et y_0 :

$$H_k = \begin{pmatrix} \frac{\partial D}{\partial x} & \frac{\partial D}{\partial y} \\ \frac{\partial \alpha}{\partial x} & \frac{\partial \alpha}{\partial y} \end{pmatrix}$$

où :

$$D = \sqrt{x^2 + y^2}$$
$$\alpha = \arctan(y/x)$$

où on trouve :

$$\frac{\partial D}{\partial x} = \frac{x}{\sqrt{x^2+y^2}} \qquad \frac{\partial D}{\partial y} = \frac{y}{\sqrt{x^2+y^2}} \qquad \frac{\partial \alpha}{\partial x} = \frac{-y}{x^2+y^2} \qquad \frac{\partial \alpha}{\partial y} = \frac{x}{x^2+y^2}$$

et, donc, l'équation 7 dans le point x_0, y_0 on rappelle ici que $x_0 = x_{k|k-1}$ et que $y_0 = y_{k|k-1}$ devient :

$$z_k = \begin{pmatrix} \frac{x_0}{\sqrt{x_0^2+y_0^2}} & \frac{y_0}{\sqrt{x_0^2+y_0^2}} \\ \frac{-y_0}{x_0^2+y_0^2} & \frac{x_0}{x_0^2+y_0^2} \end{pmatrix} \begin{pmatrix} x_k - x_0 \\ y_k - y_0 \end{pmatrix} + \begin{pmatrix} \sqrt{x_0^2+y_0^2} \\ arctan(\frac{y_0}{x_0}) \end{pmatrix} + \begin{pmatrix} n_D(k) \\ n_s(k) \end{pmatrix} \tag{8}$$

A partir de l'équation linéarisée ci-dessus, on trouve la nouvelle équation du bruit :

$$b(k) = \begin{pmatrix} \sqrt{x_0^2+y_0^2} \\ arctan(\frac{y_0}{x_0}) \end{pmatrix} - H_k \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} n_D(k) \\ n_s(k) \end{pmatrix} \tag{9}$$

Comme on peut le voir dans l'équation 8, le bruit a une moyenne $\begin{pmatrix} \sqrt{x_0^2+y_0^2} \\ arctan(\frac{y_0}{x_0}) \end{pmatrix} - H_k \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$
Donc, il n'est pas centré.
Conclusion :

$$z_k = \begin{pmatrix} \frac{x_0}{\sqrt{x_0^2+y_0^2}} & \frac{y_0}{\sqrt{x_0^2+y_0^2}} \\ \frac{-y_0}{x_0^2+y_0^2} & \frac{x_0}{x_0^2+y_0^2} \end{pmatrix} \begin{pmatrix} x_k - x_0 \\ y_k - y_0 \end{pmatrix} + \begin{pmatrix} \sqrt{x_0^2+y_0^2} \\ arctan(\frac{y_0}{x_0}) \end{pmatrix} + \begin{pmatrix} n_D(k) \\ n_s(k) \end{pmatrix} \tag{10}$$

3.2

Objectif :
Modifier les équations du filtre de Kalman pour un bruit de mesure non centré.
Développement :
Pour ternir compte d'un bruit de mesure non centré, les équations du filtre de Kalman doivent être représentées comme suit :

$$z_k = H_k(x - x_{k|k-1}) + \begin{pmatrix} \sqrt{x_{k|k-1}^2 + y_{k|k-1}^2} \\ arctan(\frac{y_{k|k-1}}{x_{k|k-1}}) \end{pmatrix}$$

$$\begin{cases} z_k = H_k x + M_k \\ M_k = -H_k x_{k|k-1} + \begin{pmatrix} \sqrt{x_{k|k-1}^2 + y_{k|k-1}^2} \\ arctan(\frac{y_{k|k-1}}{x_{k|k-1}}) \end{pmatrix} \end{cases}$$

Enfin, les équations du filtre de Kalman pour un bruit de mesure non centré sont représentées sous la forme matricielle suivante :

$$z_k = \begin{pmatrix} H_k & I_2 \end{pmatrix} \begin{pmatrix} X_k \\ M_k \end{pmatrix} + \begin{pmatrix} n_D(k) \\ n_s(k) \end{pmatrix}$$

Les états ajoutés au système pour tenir en compte de bruit gaussian non centré, M_k, doivent être calculés a chaque itération. Normalement ça est fait pour la matrice F
Conclusion :

$$\begin{cases} z_k = \begin{pmatrix} H_k & I_2 \end{pmatrix} \begin{pmatrix} X_k \\ M_k \end{pmatrix} + \begin{pmatrix} n_D(k) \\ n_s(k) \end{pmatrix} \\ M_k = -H_k x_{k|k+1} + \begin{pmatrix} \sqrt{(x_{k|k+1})^2 + (y_{k|k+1})^2} \\ arctan(\frac{y_{k|k+1}}{x_{k|k+1}}) \end{pmatrix} \end{cases}$$

3.3

Objectif :
Déduire les équations du filtre de Kalman correspondant à ce modèle linéarisé, en faisant apparaître les paramètres nécessaires à l'initialisation et au fonctionnement du filtre d'une récurrence à l'autre
Dévelopment :
Le problème est modelisé par le système d'équations suivantes :

$$\begin{cases} x_{k+1} = F_k x_k + G_k u_k \\ z_k = H x_k + v_k \end{cases}$$

où x_k représent l'état du système ; z_k représent le vecteur de mésures. Dans ce cas ici, on mésure la distance D_k et l'angle α_k ; u_k représent le bruit de modélisation qui a été déjà definit ; et v_k représente le bruit de mésure et suit un loi normale $\sim \mathcal{N}(0, V)$.
Dans ce cas, avec un bruit de mesure non centré, on doit ajouter 2 nouveaux états à matrice d'états x_k pour qu'ils soient traités.
Les équations de filtre de Kalman sont séparés en 2 parties : prédiction et correction, comme dans le cas précédente.
Prédiction :

$$\begin{cases} x_{k+1} = F_k x_k + A_k \\ P_{k+1} = F_k P_k F_k^T + W_k \end{cases}$$

où W_k c'est la matrice qui représente le bruit de modélisation ; P_k c'est la matrice de covariance avec les paramètres arbitrairement grande ; et A_k c'est la matrice :

$$A_{8x1} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ M_k \end{pmatrix}$$

où M_k c'est la matrice M_{2x1} qui ajuste le bruit de mesure non centré, qui est décrit comme :

$$\begin{cases} M_k = -H_k x_{k|k-1} + \begin{pmatrix} \sqrt{x_{k|k-1}^2 + y_{k|k-1}^2} \\ arctan(\frac{y_{k|k-1}}{x_{k|k-1}}) \end{pmatrix} \end{cases}$$

En tenant compte des 2 nouveaux états de la matrice x_k, nous devons compléter par des zéros les matrices F_k, P_k, W_k afin qu'elles supportent les opérations matricielles de multiplication. Donc, on a :

$$F_k = \begin{pmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 & T & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & T & 0 & 0 \\ 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad P_k = \gamma I_{8x8} \qquad W_k = I_{8x8} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ T\sigma_w^2 \\ T\sigma_w^2 \\ 0 \\ 0 \end{pmatrix}^T$$

où γ c'est un constant arbitrairement grande.
Correction - Calcul du gain de Kalman :

$$K_{k+1} = P_{k+1} H_{k+1}^T (H_{k+1} P_{k+1} H_{k+1}^T + V_{k+1})^{-1}$$

Correction - Mise à jour :

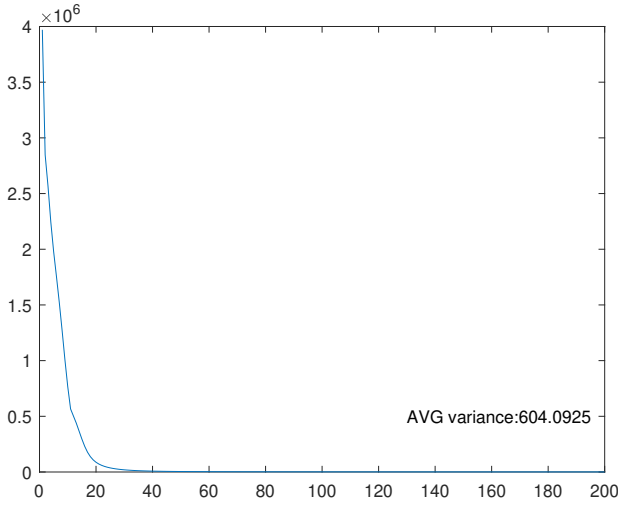
$$\begin{cases} x_{k+1} = x_{k+1} + K_{k+1}(z_{k+1} - H_{k+1} x_{k+1}) \\ P_{k+1} = (I - K_{k+1} H_{k+1}) P_{k+1} \end{cases}$$

La matrice de covariance V, on utilise les variances du bruit blancs associés aux paramètres D_k et α_k :

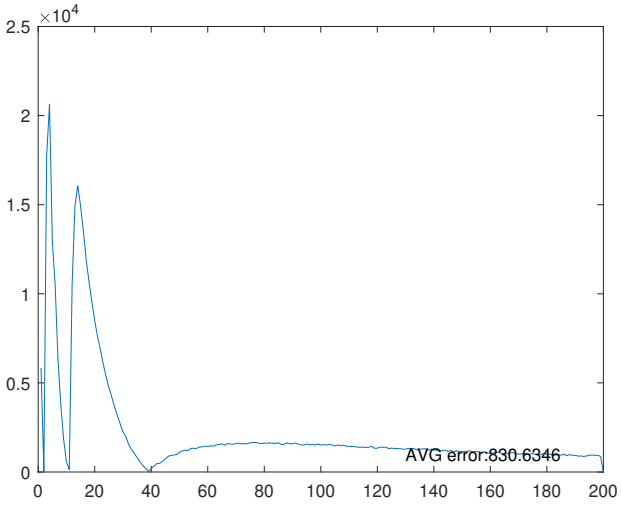
$$V_k = \begin{pmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_s^2 \end{pmatrix}$$

3.4

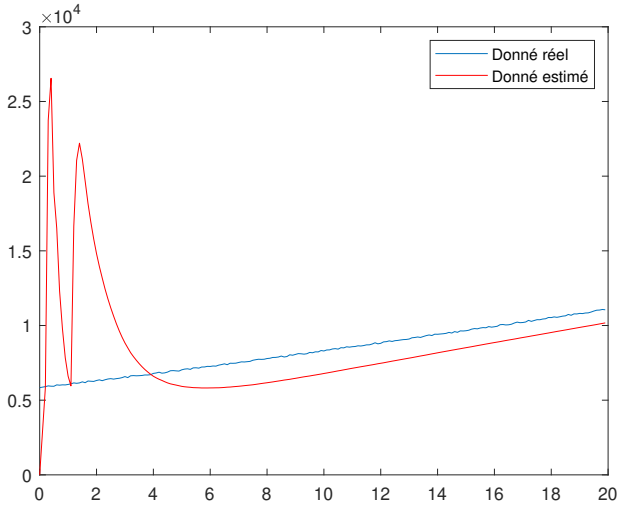
Objectif : Expliquer les influences de la initialisation des états du système et des valeur de la matrice de covariance pour la convergence du algorithme.
Développent : Après avoir implémenter l'algorithme, on a exploré plusieurs scénarios de initialisation des états du système et de la matrice de covariance.
On observe que pour une initialisation égal à celle-la de la question précédant, le filtre se porte de façon sub-optimal même si converge. Il reste toujours biasé dans l'estimation des états. On attribut ça ai processus de linéarisation de la matrice H_k.



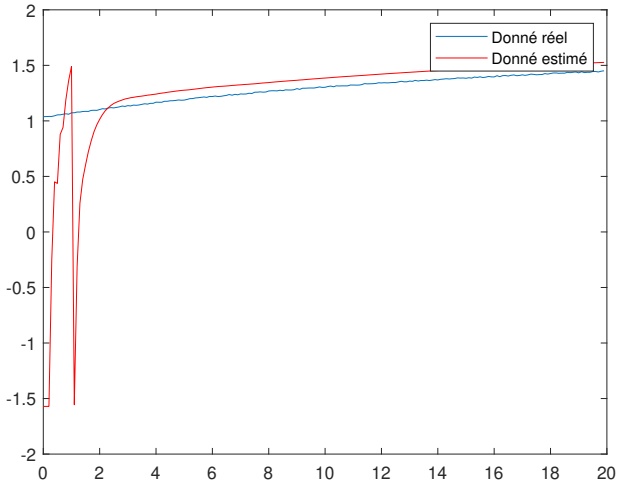
(a) Trace de la matrice de variance



(b) Erreur quadratique



(c) Position X



(d) Position Y

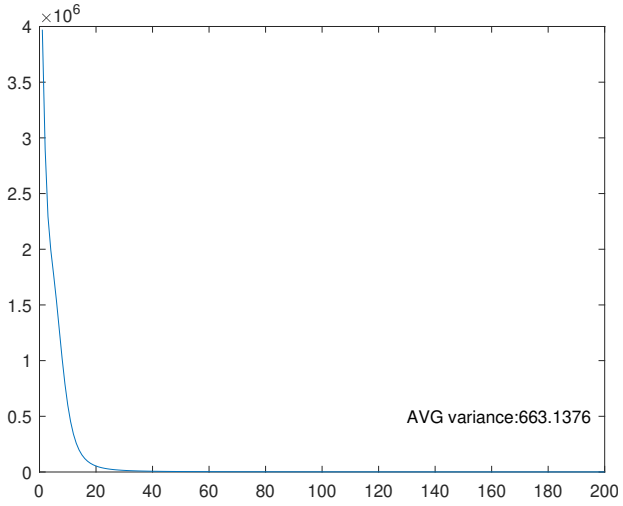
FIGURE 3 – Résultats Filtre de Kalman Étendue avec de jeu de donnés 1, initialisation des états = 0 et variance initial égal a 10^6.

Par contre si on l'initialise avec les premier pair de donnés, la position, le système marche très bien, avec un erreur moins significatif que filtre de Kalman simple.

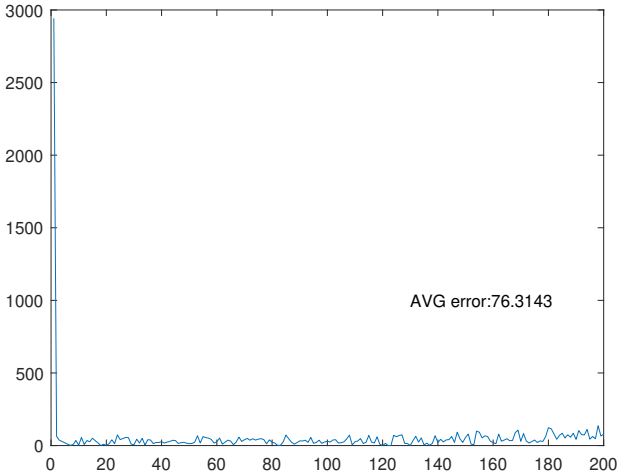
Conclusion :

Le filtre de Kalman Étendue n'as pas les mêmes garanties de convergence théoriques que le filtre de Kalman simple. Par contre il marche bien pour les systèmes bien comportés, où la dérive de la courbe ne change pas beaucoup entre interactions.

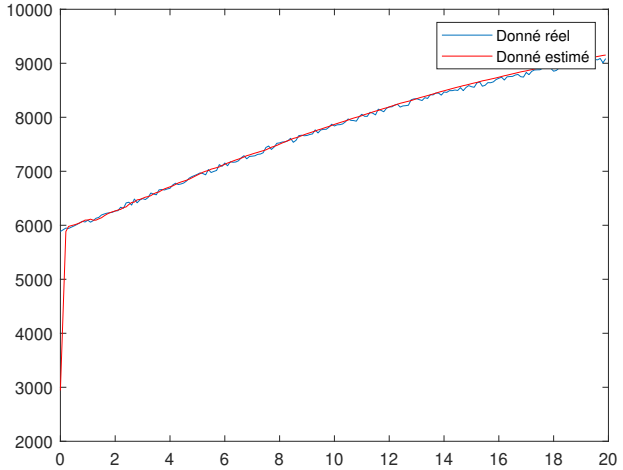
Il a un problème de convergence si on prend en compte les mêmes conditions de initialisations utilisées dans le filtre de Kalman simple. C'est pas a dire qu'il ne pourrait jamais marcher bien mais qu'il faut bien analyser les conditions initiales. On suggère démarrer l'algorithme avec les donnés initiales du capteur. Un teste a été fait avec les donnés initiales du capteur déformés en 50% et le résultat se suivre. Comme on peut voir, le système marche comme attendue.



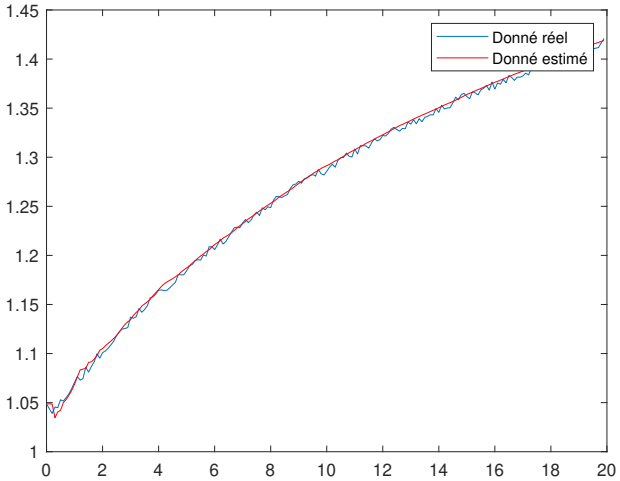
(a) Trace de la matrice de variance



(b) Erreur quadratique



(c) Position X



(d) Position Y

FIGURE 4 – Résultats Filtre de Kalman Étendue avec de jeu de donnés 2, initialisation des états de position avec les premiers donnés du capteur et variance initial égal a 10^6.

3.5

L'algorithme comme il a été définit a été implémente et le code peut être trouvé sur la section 7.2. Les résultats on été déjà donnés sur la question précèdent.

4 Exercice 4

Rien a faire.

5 Exercice 5

5.1

Montré dans la session 3.3.

5.2

Utilisant $(z_i(k+1|k)$ et $x(k+1|k)$ comme matrices 2x1 qui contiennent les donnés polaires du capteur et les ses estimés respectivement.

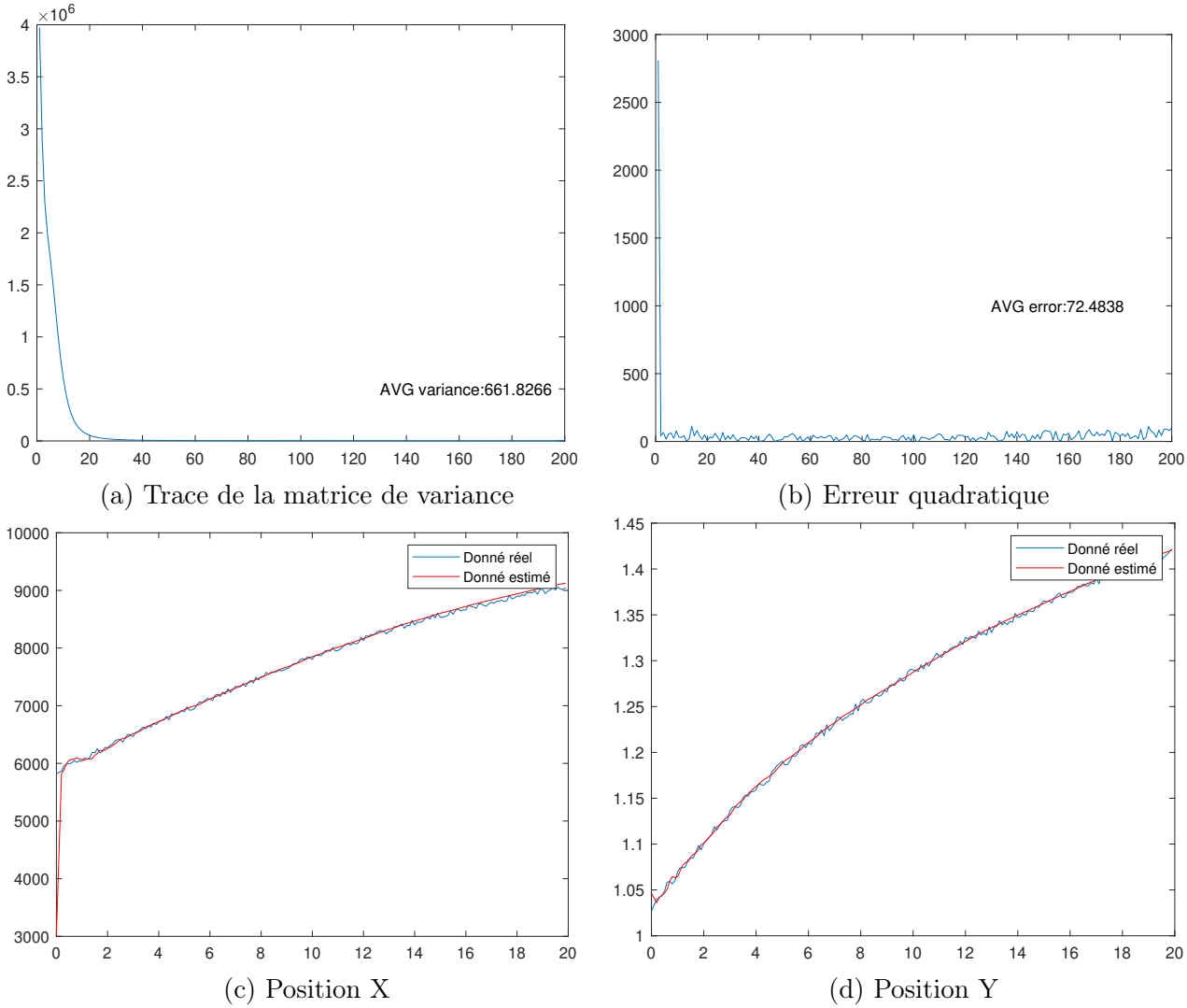
$$EQM_i = sum\left(\frac{(z_i(k+1|k) - x(k+1|k) * (z_i(k+1|k) - x(k+1|k)^t}{2}\right)$$

C'est important noter que les donnés $x(k+1|k)$ ont été calculés des estimations cartesiennes des états, c'est a dire avec le produit $H_k * x_{k|k-1}$.

5.3

```
% NNSF - selection des mesures les plus probables
 xk = X_est_polar(i,1); % donnés estimés
 yk = X_est_polar(i,2);
 for j = 1:5
     eqm(j) = sum(sqrt(([xk;yk]-[mesures(i,j);mesures(i,j+5)]).^2));
 end
 [erreur(1,i), j_best] = min(eqm);
 goodMesure(i,:) = [mesures(i,j_best) mesures(i,j_best+5)];
 % suite du algorithme avec "goodMesure"...
```

Comme on attendait, on a eu un résultat similaire auquel on a eu dans la question précèdent.



(a) Trace de la matrice de variance

(b) Erreur quadratique

(c) Position X

(d) Position Y

FIGURE 5 – Résultats Filtre de Kalman Étendue avec de jeu de donnés 3, initialisation des états de position avec les premiers donnés du capteur divisé par 2 et variance initial égal a 10^6.

6 Exercise 6

Dejá fait dans le dévelopment du rapport.

7 Anexes

7.1 EX2

```
% Filtre de kalman pour l'estimation de position

clear variables;
close all;
clc;

load('mesurestrajKalm2.mat');
% mesures polaires
% sigmesurayon
% simesuang
% Tmesu
Z(:,1) = mesures(:,1) .* cos(mesures(:,2));
Z(:,2) = mesures(:,1) .* sin(mesures(:,2));

N = size(Z,1);

% echantillonage de 0.1s
T = 0.1;

g = 9.81;

sigma_a = 0.05;
mu = 0.01;
beta = exp(-mu);
sigma_w = sigma_a * sqrt(1-exp(-2*mu));

% FILTRE DE KALMAN
% initialization
X0 = [0;0;0;0;0;g*(beta - 1)];
Pi = zeros(6,6,N);
P0 = [10^6    0    0    0  0    0;
         0 10^6    0    0  0    0;
         0    0 10^6    0  0    0;
         0    0    0 10^6  0    0;
         0    0    0    0  10^6 0;
         0    0    0    0  0    10^6];
F = [1 0 T 0 T^2/2 0         ;
     0 1 0 T 0     T^2/2     ;
     0 0 1 0 T     0         ;
     0 0 0 1 0     T         ;
     0 0 0 0 beta  0         ;
     0 0 0 0 0     beta     ];
H = [1 0 0 0 0 0 ;
     0 1 0 0 0 0 ];
W = T * [0 0 0 0 0     0         ;
         0 0 0 0 0     0         ;
         0 0 0 0 0     0         ;
```

```
         0 0 0 0 0     0          ;
         0 0 0 0 sigma_w^2     0          ;
         0 0 0 0      0     sigma_w^2];

X_est = zeros(6,N);
var = zeros(1,N);
erreur = zeros(1,N);

% ---------
% main loop
% ---------
for i = 1:N-1
    % prediction
    if i > 1
        X_est(:,i) = F*X_est(:,i-1);
        Pi(:,:,i) = F*Pi(:,:,i-1)*F' + W;
    else
        X_est(:,1) = F*X0;
        Pi(:,:,1) = F*P0*F' + W;
    end

    erreur(1,i) = sum(sqrt((H*X_est(:,i)-Z(i,:)').*(H*X_est(:,i)-Z(i,:)')));
    % correction
    % gain de Kalman
    D = mesures(i,1);
    alpha = mesures(i,2);
    J = [cos(alpha) -D*sin(alpha) ;
         sin(alpha) D*cos(alpha)];
    V = [sigmesurayon^2 0 ;
          0 simesuang^2];
    Vn = J*V*J';
    Ki = Pi(:,:,i)*H'*inv(H*Pi(:,:,i)*H'+Vn);
    % mise a jour
    X_est(:,i) = X_est(:,i) + Ki*(Z(i+1,:)'-H*X_est(:,i));
    Pi(:,:,i) = (eye(6)-Ki*H)*Pi(:,:,i);
    var(1,i) = trace(Pi(:,:,i));
end

% plot
n = 1:N;

figure(1)
plot(n,var);
moyenne = mean(var(end-10:end));
str = strcat('AVG variance: ',num2str(moyenne));
text(130,500000,str);

figure(2)
plot(n,erreur)
moyenne = mean(erreur(end-10:end));
```

```
str = strcat('AVG error: ',num2str(moyenne));
text(130,1000,str);

figure(3)
plot(Tmesu(1:end-1),Z(1:end-1,1),Tmesu(1:end-1),X_est(1,1:end-1),'r')
legend('Donné réel','Donné estimé')

figure(4)
plot(Tmesu(1:end-1),Z(1:end-1,2),Tmesu(1:end-1),X_est(2,1:end-1),'r')
legend('Donné réel','Donné estimé')
```

7.2 EX3

```
% Filtre de kalman pour l'estimation de position

clear variables;
close all;
clc;

load('mesurestrajKalm2.mat');
% mesures polaires
% sigmesurayon
% simesuang
% Tmesu

N = size(mesures,1);

% echantillonage de 0.1s
T = 0.1;

g = 9.81;

sigma_a = 0.05;
mu = 0.01;
beta = exp(-mu);
sigma_w = sigma_a * sqrt(1-exp(-2*mu));

% FILTRE DE KALMAN
% initialisation
x_0 = mesures(1,1) * cos(mesures(1,2));
y_0 = mesures(1,1) * sin(mesures(1,2));
X0 = [x_0/2;y_0/2;0;0;0;-g;0;0];
%X0 = [0;0;0;0;0;-g;0;0];
Pi = zeros(8,8,N);
P0 = 10^6 * eye(8);
F = [1 0 T 0 T^2/2 0     0 0     ;
     0 1 0 T 0     T^2/2 0 0     ;
     0 0 1 0 T     0     0 0     ;
     0 0 0 1 0     T     0 0     ;
     0 0 0 0 beta  0     0 0     ;
     0 0 0 0 0     beta  0 0     ;
     0 0 0 0 0     0     0 0     ;
     0 0 0 0 0     0     0 0    ];
H = [0 0 0 0 0 0 1 0;
     0 0 0 0 0 0 0 1];
W = T * [0 0 0 0 0         0         0 0;
         0 0 0 0 0         0         0 0;
         0 0 0 0 0         0         0 0;
         0 0 0 0 0         0         0 0;
         0 0 0 0 sigma_w^2 0         0 0;
         0 0 0 0 0         sigma_w^2 0 0;
         0 0 0 0 0         0         0 0;
```

```
         0 0 0 0 0         0         0 0];
V = [sigmesurayon^2 0 ;
          0 simesuang^2];
X_est = zeros(8,N);
var = zeros(1,N);
erreur = zeros(1,N);

X_est_polar = zeros(200,2);

% ---------
% main loop
% ---------
for i = 1:N
    % prediction
        if i > 1
            X_est(:,i) = F*X_est(:,i-1);
            Pi(:,:,i) = F*Pi(:,:,i-1)*F' + W;
        else
            X_est(:,1) = F*X0;
            Pi(:,:,1) = F*P0*F' + W;
        end

        x = X_est(1,i);
        y = X_est(2,i);

        % linearization matrice H

        Hk = [x/sqrt(x^2+y^2)  y/sqrt(x^2+y^2)
              -y/(x^2+y^2)     x/(x^2+y^2)];
        H(1:2,1:2) = Hk;

        X_est_polar(i,:) = [sqrt(x^2+y^2); atan(y/x)];

        % mise a jour mk_x et mk_y
        X_est(7:8,i) = -H*X_est(:,i) + [sqrt(x^2+y^2); atan(y/x)];

    erreur(1,i) = sum(sqrt((H*X_est(:,i)-mesures(i,:)').*(H*X_est(:,i)-mesures(i,:)')));

    % correction
        % gain de Kalman
        Ki = Pi(:,:,i)*H'*inv(H*Pi(:,:,i)*H'+V);
        % mise a jour
        X_est(:,i) = X_est(:,i) + Ki*(mesures(i,:)'-H*X_est(:,i));
        Pi(:,:,i) = (eye(8)-Ki*H)*Pi(:,:,i);
        var(1,i) = trace(Pi(:,:,i));
end

% plot
n = 1:N;
```

```
figure(1)
plot(n,var);
moyenne = mean(var(end-10:end));
str = strcat('AVG variance: ',num2str(moyenne));
text(130,500000,str);

figure(2)
plot(n,erreur)
moyenne = mean(erreur(end-10:end));
str = strcat('AVG error: ',num2str(moyenne));
text(130,1000,str);

figure(3)
plot(Tmesu(1:end-1),mesures(1:end-1,1),Tmesu(1:end-1),X_est_polar(1:end-1,1),'r')
legend('Donné réel','Donné estimé')

figure(4)
plot(Tmesu(1:end-1),mesures(1:end-1,2),Tmesu(1:end-1),X_est_polar(1:end-1,2),'r')
legend('Donné réel','Donné estimé')
```

7.3 EX5

```
% Filtre de kalman pour l'estimation de position et vitesse

clear variables;
close all;
clc;

load('mesurestrajKalm3.mat');
% mesures polaires
% sigmesurayon
% simesuang
% Tmesu

N = size(mesures,1);

% echantillonage de 0.1s
T = 0.1;

g = 9.81;

sigma_a = 0.05;
mu = 0.01;
beta = exp(-mu);
sigma_w = sigma_a * sqrt(1-exp(-2*mu));

% FILTRE DE KALMAN
% initialisation
x_0 = mesures(1,1) * cos(mesures(1,6));
y_0 = mesures(1,1) * sin(mesures(1,6));
X0 = [x_0/2;y_0/2;0;0;0;-g;0;0];
%X0 = [0;0;0;0;0;-g;0;0];
Pi = zeros(8,8,N);
P0 = 10^6 * eye(8);
F = [1 0 T 0 T^2/2 0     0 0     ;
     0 1 0 T 0     T^2/2 0 0     ;
     0 0 1 0 T     0     0 0     ;
     0 0 0 1 0     T     0 0     ;
     0 0 0 0 beta  0     0 0     ;
     0 0 0 0 0     beta  0 0     ;
     0 0 0 0 0     0     0 0     ;
     0 0 0 0 0     0     0 0    ];
H = [0 0 0 0 0 0 1 0;
     0 0 0 0 0 0 0 1];
% TODO
W = T * [0 0 0 0 0         0         0 0;
         0 0 0 0 0         0         0 0;
         0 0 0 0 0         0         0 0;
         0 0 0 0 0         0         0 0;
         0 0 0 0 sigma_w^2 0         0 0;
         0 0 0 0 0         sigma_w^2 0 0;
```

```
         0 0 0 0 0         0         0 0;
         0 0 0 0 0         0         0 0];
V = [sigmesurayon^2 0 ;
          0 simesuang^2];
X_est = zeros(8,N);
var = zeros(1,N);
erreur = zeros(1,N);

X_est_polar = zeros(200,2);
eqm = zeros(5,1);
goodMesure = zeros(200,2);

% ---------
% main loop
% ---------
for i = 1:N
    % prediction
        if i > 1
            X_est(:,i) = F*X_est(:,i-1);
            Pi(:,:,i) = F*Pi(:,:,i-1)*F' + W;
        else
            X_est(:,1) = F*X0;
            Pi(:,:,1) = F*P0*F' + W;
        end

        x = X_est(1,i);
        y = X_est(2,i);

        % linearization matrice H

        Hk = [x/sqrt(x^2+y^2)  y/sqrt(x^2+y^2)
              -y/(x^2+y^2)     x/(x^2+y^2)];
        H(1:2,1:2) = Hk;

        X_est_polar(i,:) = [sqrt(x^2+y^2); atan(y/x)];

        % mise a jour mk_x et mk_y
        X_est(7:8,i) = -H*X_est(:,i) + [sqrt(x^2+y^2); atan(y/x)];

     % NNSF - selection des mesures les plus probables
     xk = X_est_polar(i,1);
     yk = X_est_polar(i,2);
     for j = 1:5
     eqm(j) = sum(sqrt(([xk;yk]-[mesures(i,j);mesures(i,j+5)]).^2));
     end
     [erreur(1,i), j_best] = min(eqm);
     goodMesure(i,:) = [mesures(i,j_best) mesures(i,j_best+5)];


    % correction
```

```
        % gain de Kalman
        Ki = Pi(:,:,i)*H'*inv(H*Pi(:,:,i)*H'+V);
        % mise a jour
        X_est(:,i) = X_est(:,i) + Ki*(goodMesure(i,:)'-H*X_est(:,i));
        Pi(:,:,i) = (eye(8)-Ki*H)*Pi(:,:,i);
        var(1,i) = trace(Pi(:,:,i));
end

% plot
n = 1:N;

figure(1)
plot(n,var);
moyenne = mean(var(end-10:end));
str = strcat('AVG variance: ',num2str(moyenne));
text(130,500000,str);

figure(2)
plot(n,erreur)
moyenne = mean(erreur(end-10:end));
str = strcat('AVG error: ',num2str(moyenne));
text(130,1000,str);

figure(3)
plot(Tmesu(1:end-1),goodMesure(1:end-1,1),Tmesu(1:end-1),X_est_polar(1:end-1,1),'r')
legend('Donné réel','Donné estimé')

figure(4)
plot(Tmesu(1:end-1),goodMesure(1:end-1,2),Tmesu(1:end-1),X_est_polar(1:end-1,2),'r')
legend('Donné réel','Donné estimé')
```