ROB312 - TD Filtrage Particulaire

BRAMBILLA Davide Luigi

19 Décembre 2019

Le Filtre Particulaire

Le filtrage Particulaire représente une technique d'estimation des systèmes et est basé sur la **simulation**. Cette technique se distingue des filtres de Kalman pour le fait que, avec un numéro suffisamment grand de particules, peut atteindre une plus haute précision.

L'objectif du filtre particulaire est d'estimer les états réels du système en tenant en compte des variables d'observation.

Comme avantages il a le fait d'utiliser une dynamique non linéaire et des modèles de mesure non linéaires, le fait de pouvoir commencer avec une haute incertitude, d'être facile à implémenter et de prendre en compte des modèles non linéarisé avec des incertitudes non gaussienne.
Comme désavantage il a le fait que il n'est pas facile choisir les paramètres, il peut converger vers maxima locaux et qu'il est une méthode qui demande beaucoup des calculs.

1 Le code et les paramètres

Dans cette section je vais présenter les paramètres utilisés dans le code. Au début de notre algorithme il est possible de voir la définition des paramètres initiaux et lesquels qui nous allons utiliser et modifier pendant l'exécution du programme. On trouve:

- X_r qui représente les états effectifs du système qui ne sont pas connus et sur lesquels on devra faire l'estimation. Initialement, ils sont définis à partir des états estimés initiaux au travers de la variable *erreurInitiale* définie en utilisant la matrice de covariance initiale.
- $\hat{X}$ qu'indique l'estimation des états. Elle sera composé par les états estimés par le filtre qui devront se rapprocher le plus possible des états réels.
- **N** qui représente le nombre de particules du système.
- $\hat{P}$ qui représente la matrice de covariance et qui sera mise à jour à chaque boucle dans l'exécution de notre algorithme.
- Q_f qui donne la matrice de covariance du bruit de dynamique du filtre. Ce type de bruit influence les valeurs estimés des états du système.
- R qui donne la matrice de covariance du bruit de mesure réelle. Ce type de bruit, qui est présente sur le capteur utilisé, influence directement les états mesurés du système.
- R_f qui donne la matrice de covariance du bruit de mesure du filtre. Ce type de bruit influence directement les états mesurés du système.
- **Seuil de ré-échantillonnage** N_{thr} qui permet de choisir s'il est nécessaire effectuer l'opération de ré-échantillonage. En particulier, si le nombre effectif de particules N_{eff} est plus petit de cette valeur, on effectue le ré-échantillonage.
- **Y** représente les mesures faits sur le système.
- **Xp** qui est composé par les particules de filtre qui, au début, sont prises au tour de $\hat{X}$.
- **wp** qui représente les poids associés aux particules.

En suite, pour étudier la structure du code, nous avons reporté ci-dessous un schéma qui décrit son fonctionnement.

Nous avons choisi de représenter trois grands parties: la première partie qui est composé par l'initialisation des paramètres du filtre et du système, le boucle vrai et propre qui termine à la fin du temps de simulation et le plot des résultats:

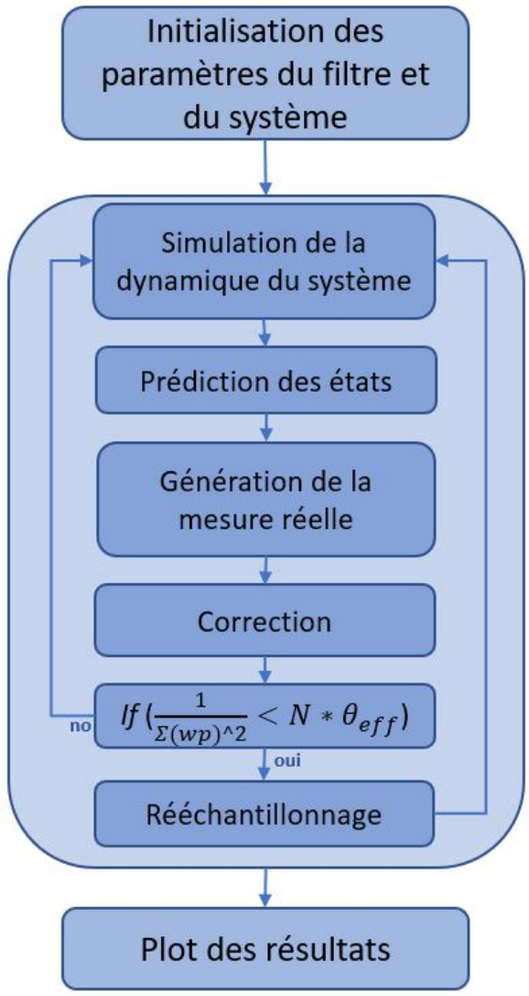


Figure 1: Schéma qui décrit le fonctionnement du code

2 Remplissage des parties manquantes de code

Dans cette section nous allons reporter la façon dont on a complète le code dans ses différents parties.
Au début nous avons reporté la dynamique du système en se basant sur le problème donné qui considère un avion qui avance à *vitesse* constante et qui varie sa trajectoire avec un *omega* constante.
Nous allons considérer les états du système comme $X_k = [x_k, y_k, z_k, \theta_k]$ et la commande $u_k = [V, \omega_k]$ Les équations utilisées sont:

$$X_{k+1} = f(X_k, U_k) = \begin{bmatrix} x_k + V\Delta t cos(\theta_k) \\ y_k + V\Delta t sin(\theta_k) \\ z_k \\ \theta_k + \Delta t \omega_k \end{bmatrix}$$

Pour ce que concerne les mesures, l'équation utilisé est:

$$Y_k = h(X_k) = z_k - hobs(x_k, y_k) + v_k$$

où v_k est un bruit gaussien basé sur le bruit qui affecte le capteur qui effectue la mesure.

Je vais reporter ci-dessus la dynamique du système dans le code:

2.1 La dynamique du système

```
t = dt*(tk-1); % temps courant
X_reel = [X_reel(1)+V*dt*cos(X_reel(4)); X_reel(2)+V*dt*sin(X_reel(4)); X_reel(3);
    X_reel(4)+dt*omega]; % propagation de l'etat reel (a completer)
X_reel(4) = mod(X_reel(4), 2*pi); % modulo 2pi sur le cap
```

Pour les parties suivantes (prédiction, correction et ré-échantillonage), nous nous sommes basés sur les équations décrites sur les transparents et les codes ont été reportés ci-dessus.

2.2 La prédiction

La partie de prédiction se base sur l'utilisation des particules qui, combinées avec les poids *wp*, donnent les estimation des états réels comme reporté dans la suite:

```
Xp = [Xp(1,:)+V*dt*cos(Xp(4,:)); Xp(2,:)+V*dt*sin(Xp(4,:)); Xp(3,:);
    Xp(4,:)+dt*omega*ones(1,N)] + sqrt(Qf)*randn(d,N); % particules predites
X_hat = Xp*wp'; % tat  estim  predit
X_hat(4,:) = mod(X_hat(4,:), 2*pi); % modulo 2pi sur le cap
P_hat = ((Xp-X_hat(:,1*ones(1,N))).*wp(ones(d,1),:) )*(Xp-X_hat(:,ones(1,N)))';
```

2.3 La correction

Dans cette partie du code nous allons utiliser la mesure Y basée sur la connaissance des états donnée par exemple par un capteur afin de corriger les poids qui on donne à chaque particule et on visualisera dans la suite que elle est très utile afin de réduire l'incertitude su les états et récalibrer les poids afin de rapprocher les états estimés aux états réels.

```
if is_measurementValid
    for i = 1 : N
        wp(i) = wp(i) * exp(-0.5 * (Y-(Xp(3,i)-hobs(Xp(:,i),params)))' * inv(Rf) *
            (Y-(Xp(3,i)-hobs(Xp(:,i),params))));
    end
    wp = wp./sum(wp);
end
```

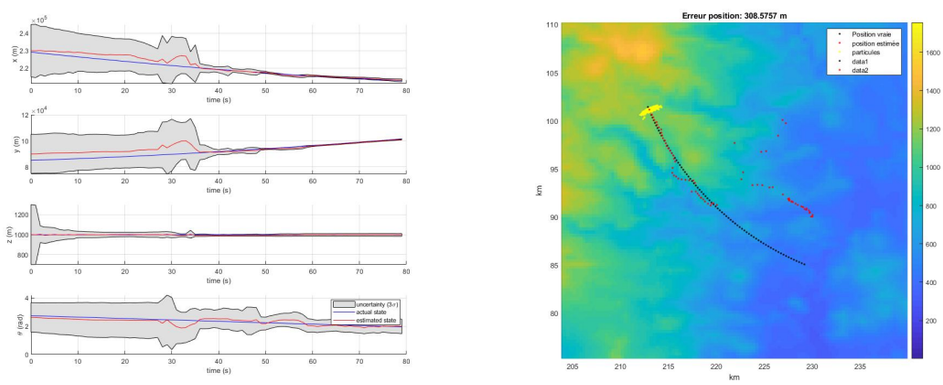
2.4 Le ré-échantillonage

Pour la partie de ré-échantillonage nous allons vérifier que la condition soit satisfaite et nous allons sélectionner des nouvelles particules et réinitialiser les poids comme présenté dans la suite:

```
criterionResampling = 1/(sum(wp.^2)); % critre de reechantillonnage "N efficace"  complter
if criterionResampling < N*threshold_resampling
    Xp = Xp(:,select(wp)); % slection des nouvelles particules selon l'algorithme de
        r-chantillonnage multinomial
    wp = 1/N*ones(1,N); % re-initialisation des poids
end
```

2.5 Les résultats

Ce que nous allons obtenir c'est des états estimés qui convergent vers les états réels avec une incertitude qui diminue toujours de plus.



(a) États vrais et estimés

(b) Erreur de position sur le schéma

Figure 2: Résultat pour l'estimation avec le filtrage particulaire

À partir de la figure de droite, il est possible de voir que, au début, les états réels et lesquels estimés sont plutôt loin l'un de l'autre(avec un erreur d'environ 5000m) mais, à la fin de notre simulation, l'algorithme arrive à beaucoup réduire l'erreur de position jusqu'à la valeur de 308.57m.

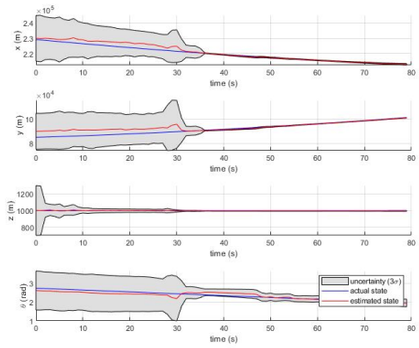
3 La variation du bruit dynamique Q_f

Dans cette section, nous allons nous baser sur la présence de bruit sur la dynamique interne de notre filtre. Ce type de bruit on va l'appliquer dans le calcul des particules pour calculer les états internes du filtre et donc il aura un effet directe sur l'estimation. J'ai reporté trois différents cas: un cas où la valeur du bruit

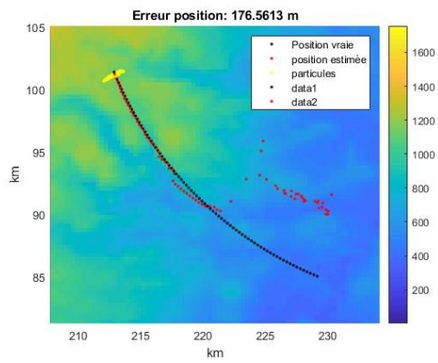
est divisé par un facteur 10, un cas où elle est multiplié par un facteur 10 et un dernier cas où la valeur est multiplié par un facteur 100. Nous allons analyser chaque cas ci-dessous.

3.1 Q_f divisé par 10

Dans le cas où nous diminuons le bruit sur le système nous attendons une amélioration des performances du filtre et c'est effectivement comme ça:



(a) États vrais et estimés



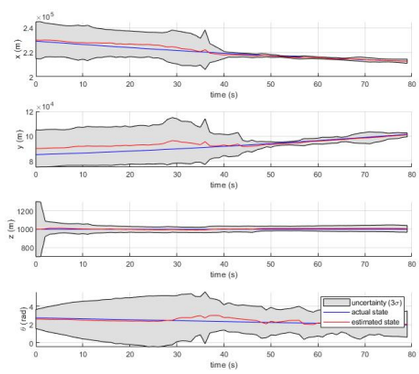
(b) Erreur de position sur le schéma

Figure 3: Résultat pour l'estimation avec le filtrage particulaire avec Q_f divisé par 10

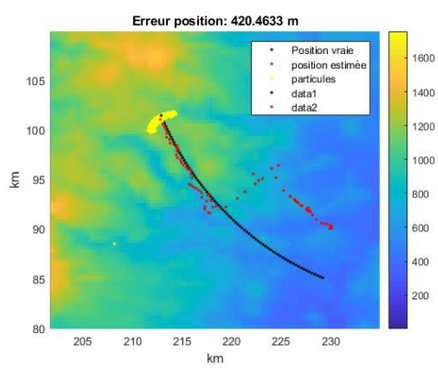
À partir des figures, il est possible de voir que l'incertitude se réduit plus en avance par rapport au cas précèdent et que les états estimés sont vraiment proches des états réels avec un erreur finale de 176.56m qui est beaucoup plus petit par rapport à l'erreur précèdent. De plus, les particules sont directes tous dans le sense de la trajectoire et elles ne sont pas trop disperses.

3.2 Q_f multiplié par 10

Dans ce cas, nous attendons des performances pires par rapport au cas précèdent.



(a) États vrais et estimés



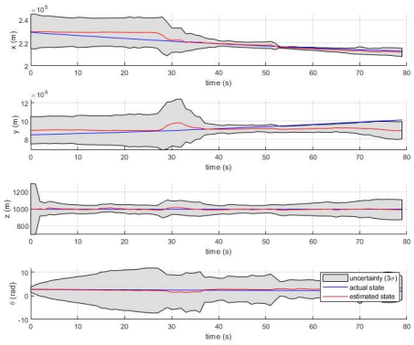
(b) Erreur de position sur le schéma

Figure 4: Résultat pour l'estimation avec le filtrage particulaire avec Q_f multiplié par 10

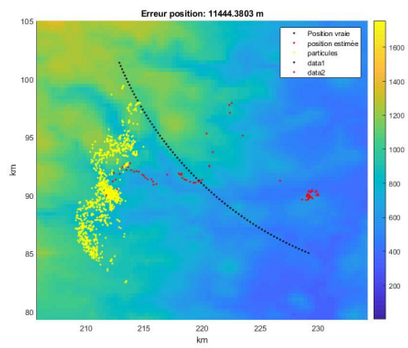
C'est possible de voir que, par rapport au cas précèdent, l'incertitude est plus grande et que donc les particules sont plus dispersées. On peut quand même dire que l'estimation reste correcte et que le résultat au moment de la fin de la simulation est de 420m qui n'est pas trop plus grand par rapport au cas dans la figure 2.

3.3 Q_f multiplié par 100

Comme dernier cas, nous avons essayé de augmenter fortement la valeur du bruit associé au système et nous n'avons pas eu des résultats satisfaisantes:



(a) États vrais et estimés



(b) Erreur de position sur le schéma

Figure 5: Résultat pour l'estimation avec le filtrage particulaire avec Q_f multiplié par 100

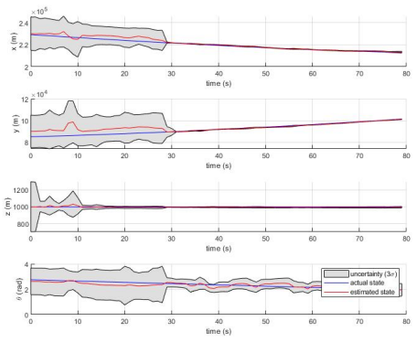
Dans les figures il est possible de voir que les estimation des états ne sont pas cohérents avec les estimation réels et les points des particules sont très disperses qui causent une très grande incertitude pendant la durée de la simulation.
Cela il nous ramène à dire que nous ne pouvons pas avoir un bruit de dynamique trop grande à l'intérieure du filtre car, autrement, nous allons avoir une phase de prédiction mauvaise des états qui nous ramène à avoir une estimation trompé du vrai état: l'erreur de position, à la fin de la simulation, est en fait de 1144.38m qui est une valeur totalement incorrect.

4 La variation du bruit de mesure du filtre R_f

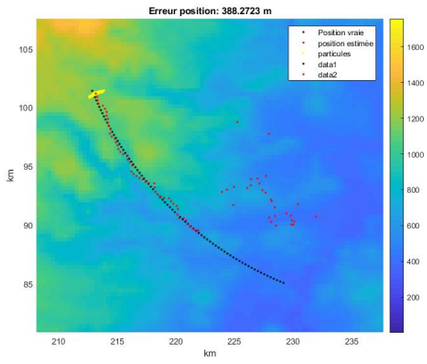
Dans cette section, il nous est demandé de jouer avec le bruit de mesure du filtre et de le faire varier entre la valeur de 10^2 et de 100^2: nous avons pris quatre as différents qui sont 10^2, 20^2, 50^2 et 100^2. Cette bruit est lié à la matrice R_f qui est utilisé dans la section de *correction* des poids. Ce qu'on attend c'est une modification et une plus grande dispersion des particules.
Ce que l'on peut observer dans les graphiques suivants c'est exactement une plus grande dispersion des particules et une plus lente convergence vers la valeur correcte de l'état réel.

4.1 $R_f = 10^2$

Dans cette cas, nous sommes en train de réduire le bruit de mesure du filtre. Les résultat sont reportés ci-dessous:



(a) États vrais et estimés



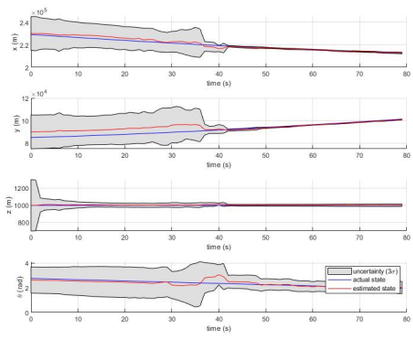
(b) Erreur de position sur le schéma

Figure 6: Résultat pour l'estimation avec le filtrage particulaire avec $R_f = 10^2$

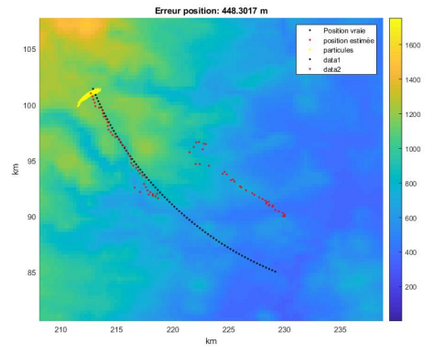
Il est possible de voir que le filtre arrive en moins de $30s$ à avoir une incertitude petite et une valeur correcte pour les estimation comme l'on peut voir dans le graphique de droite.

4.2 $R_f = 20^2$

Nous allons passer à une valeur de 20^2 et il possible de voir comme le filtre prend plus de temps à réduire l'incertitude sur les mesures:



(a) États vrais et estimés



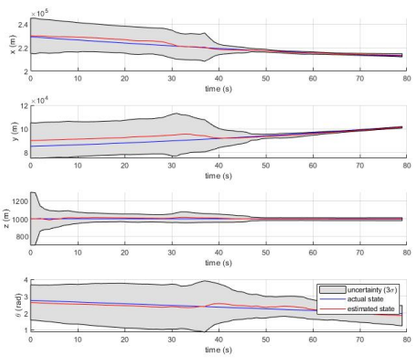
(b) Erreur de position sur le schéma

Figure 7: Résultat pour l'estimation avec le filtrage particulaire avec $R_f = 20^2$

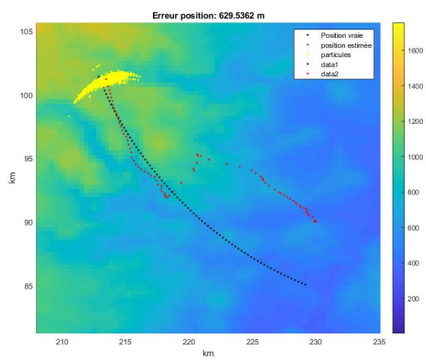
De plus, il est possible d'observer comme les particules commencent à être plus disperses par rapport à avant. Nous pouvons conclure que l'estimation est encore correcte par le fait que l'erreur finale est encore acceptable.

4.3 $R_f = 50^2$

Si on passe en suite à une valeur de 50^2, nous pouvons commencer à observer le fait que la dispersion des particules deviennent encore plus accentuée et que le filtre prend encore plus de temps à converger.



(a) États vrais et estimés



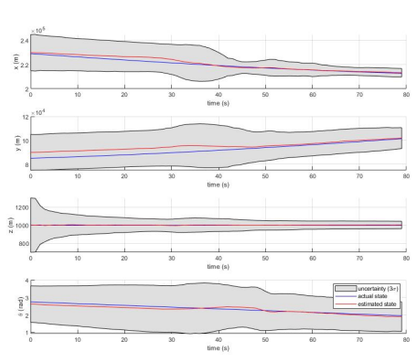
(b) Erreur de position sur le schéma

Figure 8: Résultat pour l'estimation avec le filtrage particulaire avec $R_f = 50^2$

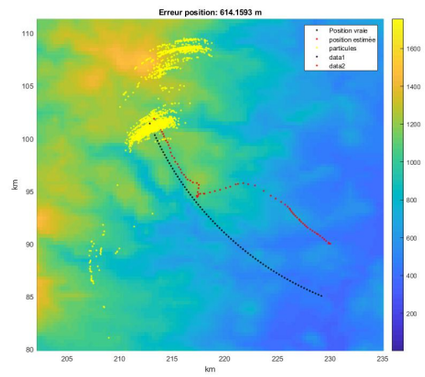
Même sur l'erreur de position finale nous avons encore une détérioration et cela est dû au fait que les particules sont plus sparses dans le plan.

4.4 $R_f = 100^2$

Enfin, nous arrivons dans le cas où nous considérons la valeur plus grande pour le bruit de mesure parmi notre intervalle.



(a) États vrais et estimés



(b) Erreur de position sur le schéma

Figure 9: Résultat pour l'estimation avec le filtrage particulaire avec $R_f = 100^2$

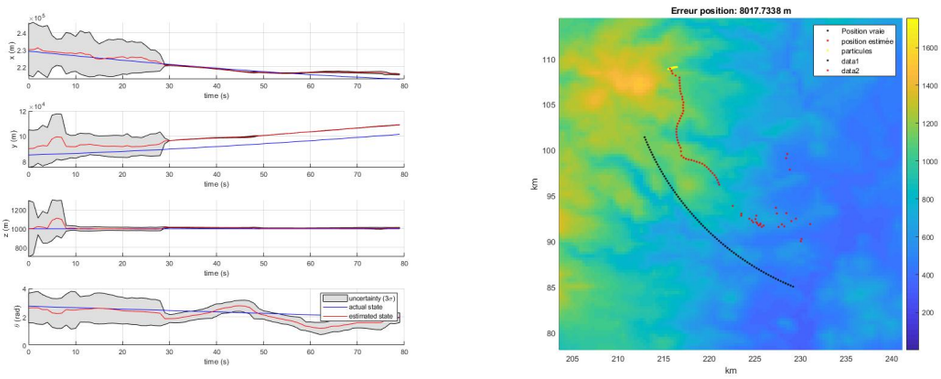
Il est évident que, dans ce cas, la dispersion des particules est plus accentuée et nous allons avoir une incertitude grande pour toute la durée de la simulation.

5 La variation du nombre de particules N

Dans cette section nous allons modifier le nombre de particules que l'algorithme pourra utiliser dans l'estimation des états. Nous attendons que, le plus grand ce nombre, le plus précis sera le résultat obtenu. Nous avons choisi d'étudier deux cas différents: le premier où on divise par un facteur 10 la valeur de N (300) et la deuxième où on multiplie par le même facteur (30000).

5.1 $N = 300$

Dans ce première cas, il est possible de voir que le résultat que nous allons obtenir n'est pas optimale à niveau d'erreur de position et il vaut 8017.73m. Cela est du au fait que les particules à disposition ne sont pas suffisant pour effectuer une bonne approximation.



(a) États vrais et estimés

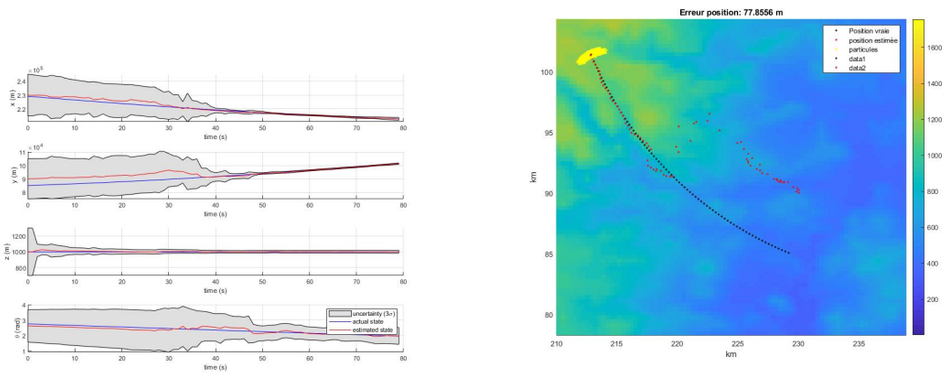
(b) Erreur de position sur le schéma

Figure 10: Résultat pour l'estimation avec le filtrage particulaire avec avec $N = \frac{3000}{10}$

Dans le graphique de gauche, il est possible de voir que, même si l'incertitude devient petite rapidement, l'estimation ne converge pas à la bonne valeur: cet effet est visible surtout dans l'état y où il y a un *bias* constante entre la valeur estimé et la valeur réelle.

5.2 $N = 30000$

Dans ce deuxième cas, nous avons un résultat beaucoup meilleur même si l'incertitude prend plus de temps à se réduire.



(a) États vrais et estimés

(b) Erreur de position sur le schéma

Figure 11: Résultat pour l'estimation avec le filtrage particulaire avec $N = 3000 * 10$

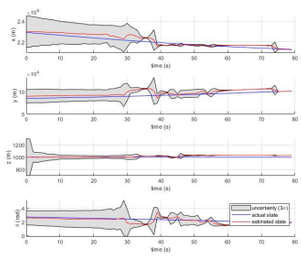
En fait, l'erreur sur la position finale est de seulement 77.85m et les particules ne sont pas trop disperses. Le fait d'avoir un plus grand nombre de particules ralentit les calculs mais permet d'avoir une estimation plus précis des états. Enfin, on peut conclure que si on a à disposition une grande nombre de particules le comportement du filtre sera plus satisfaisante et arrivera à mieux estimer les états réels.

6 La variation du seuil de ré-échantillonage

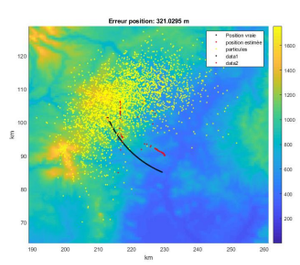
Dans ce cas, nous allons faire varier le seuil de ré-échantillonage et nous allons lui donner trois différents valeurs 0, 0.5 et 1. Cela nous permettra de voir l'effet de ré-échantillonage sur le résultat. Le ré-échantillonnage permet d'éviter le problème de la **dégénérescence** de l'algorithme où tous les poids sauf lesquels liés à un parcour sont proches de zéro. Son but est lequel de redistribuer les particules en dupliquant celles avec des poids grand et en éliminant celles avec des poids faibles.

6.1 La valeur de 0

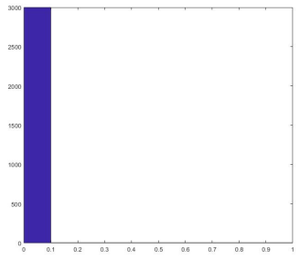
Avec une valeur de 0 nous n'allons jamais avoir le ré-échantillonage pour le fait que la condition ne sera jamais satisfaite. En fait, nous allons comparer une valeur positive à la valeur zéro et le ré-échantillonage ne sera jamais effectué.



(a) États vrais et estimés



(b) Erreur de position sur le schéma



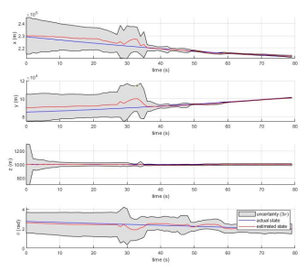
(c) Histogramme des poids

Figure 12: Résultat pour l'estimation avec le filtrage particulaire avec le seuil de ré-échantillonage de 0

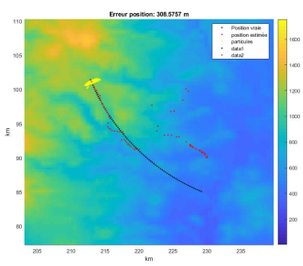
Il est possible de voir dans ce cas que nous avons le problème de **dégénérescence** car tous les poids ont la valeur de zéro.

6.2 La valeur de 0.5

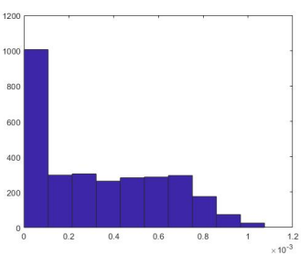
Ce cas est la valeur utilisé par défaut dans notre algorithme. Dans ce cas, il est possible d'observer que le ré-échantillonage est bien effectué.



(a) États vrais et estimés



(b) Erreur de position sur le schéma



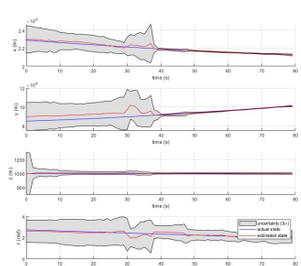
(c) Histogramme des poids

Figure 13: Résultat pour l'estimation avec le filtrage particulaire avec le seuil de ré-échantillonage de 0.5

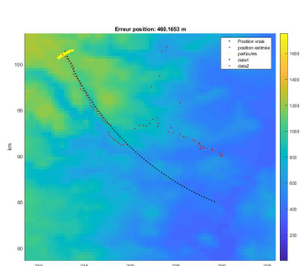
Le résultat de notre algorithme est lequel désiré avec un erreur sur la position petit et avec des poids distribués et non plus tous concentrés en zéro.

6.3 La valeur de 1

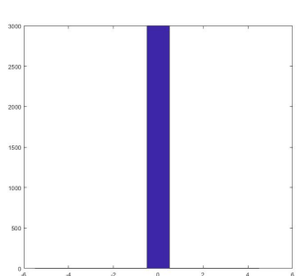
Dans ce cas aussi nous allons avoir que la grande majorité des poids vaut zéro et donc nous allons avoir une autre fois le problème de la dégénérescence et nous allons faire disparaître plus rapidement des poids car considères trop petits par rapport aux trajectoires considerés interessants.



(a) États vrais et estimés



(b) Erreur de position sur le schéma



(c) Histogramme des poids

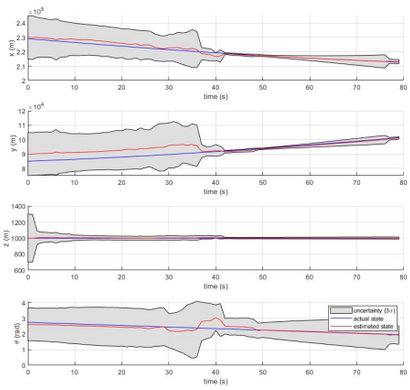
Figure 14: Résultat pour l'estimation avec le filtrage particulaire avec le seuil de ré-échantillonage de 1

7 Simulation avec un trou de mesures entre $t = 50s$ et $t = 75s$

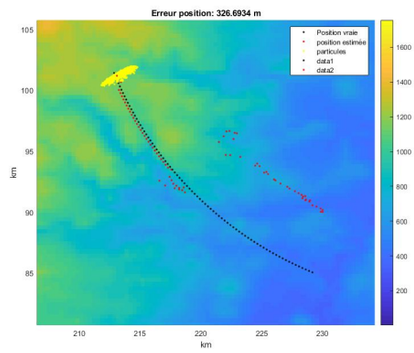
Pour désactiver la disponibilité des mesures, nous avons utilisé la variable *isMeasurementValid* et nous l'avons laissé à *false* dans l'intervalle entre $50s$ et $75s$. Cette opération nous permet de voir le comportement du filtre dans le cas où le capteur que nous donne les mesures ne fonctionne pas pour une certaine période.

```
is_measurementValid = false;
if t < 50 || t > 75
    is_measurementValid = true;
end
```

Ci-dessous, je vais reporter les graphiques obtenues:



(a) États vrais et estimés



(b) Erreur de position sur le schéma

Figure 15: Résultat pour l'estimation avec le filtrage particulaire avec le trou des mesures

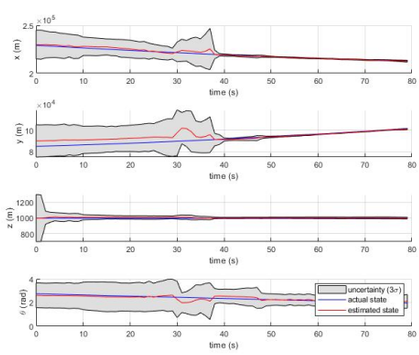
Dans le graphique il est possible de voir que l'incertitude augmente dans l'intervalle où les mesures sont absentes. Toutefois, l'estimation n'est pas mauvaises et le filtre continue à bien estimer les poids même si les mesures ne sont plus disponibles pour un certain période.

8 La modification de la fréquence des mesures à $10Hz$

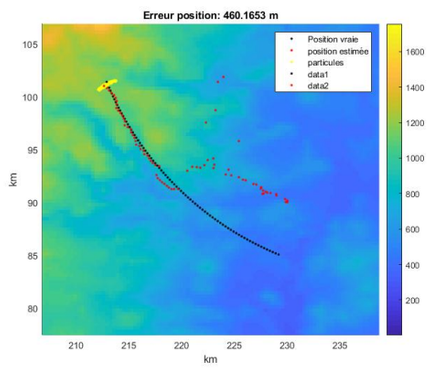
Dans cette section, nous allons analyser la cas où on diminue la fréquence de disponibilité des mesures.

```
is_measurementValid = false;
if mod(t, 0.1) == 0
    is_measurementValid = true;
end
```

Au début, nous allons considérer une fréquence de *10 Hz* qui correspond à avoir les mesures disponibles chaque $0.1s$ et nous avons obtenu:



(a) États vrais et estimés

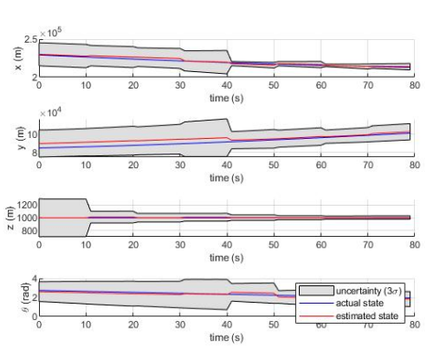


(b) Erreur de position sur le schéma

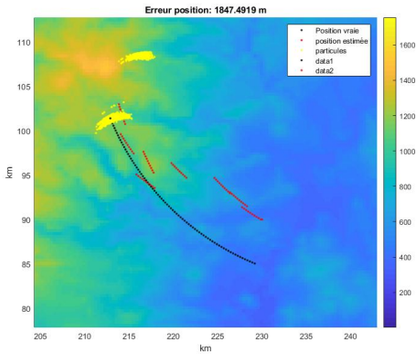
Figure 16: Résultat pour l'estimation avec le filtrage particulaire avec les mesures disponible chaque $0.01s$

Dans ce cas, nous n'avons pas noté des différences par rapport au cas où les mesures sont toujours disponibles et c'est pour cela que nous avons choisi de diminuer la fréquence à laquelle les mesures sont disponibles.

Par conséquence, nous avons considéré une fréquence de *0.1Hz* qui correspond à avoir une mesure disponible chaque $10s$ et nous avons obtenu:



(a) États vrais et estimés



(b) Erreur de position sur le schéma

Figure 17: Résultat pour l'estimation avec le filtrage particulaire avec les mesures disponible chaque $10s$

Dans ce deuxième cas, il est possible de voir plus facilement comment la présence des mesures nous permet d'avoir un incertitude mineure sur la prédiction donnée par le filtre: où les mesures ne sont pas présents l'incertitude a tendance à augmenter et l'estimation pourrait devenir pire.

9 Modification de la façon de ré-échantillonner les poids

Dans cette dernière section, il nous est demandé de proposer une autre façon de ré-échantillonner les poids et de la coder: pour cela il est suggéré de se baser sur l'article proposé.

Dans cette article, il est proposé un pseudo-code qui nous avons reporté ci-dessous et qui implémente trois différents types de ré-échantillonage: **multinomial**, **stratifié** et **systématique**.

Code 3: Multinomial/stratified/systematic resampling.

$[\{\tilde{x}_t^{(n)}\}_{n=1}^N] = \text{Resample}[\{x_t^{(m)}, w_t^{(m)}\}_{m=1}^M, N]$

$[\{Q_t^{(m)}\}_{m=1}^M] = \text{CumulativeSum}[\{w_t^{(m)}\}_{m=1}^M]$

$n = 0$

/Systematic/stratified choice runs:

  $m = 1$

/Systematic choice runs

  $u_0 \sim U(0, 1/N]$

WHILE $(n \leq N)$

/Stratified choice runs

  $u_0 \sim U(0, 1/N]$

/Systematic/stratified choice runs

  $u = u_0 + n/N$

/Multinomial choice runs

  $u \sim U(0, 1]$; $m = 1$

  WHILE $(Q_t^{(m)} < u)$

    $m = m + 1$

  END

  $n = n + 1$

  $\tilde{x}_t^{(n)} = x_t^{(m)}$

END

Figure 18: Pseudo-code pour le ré.échantillonage

Les trois codes ont été, en suite, implémentés dans *Matlab* et nous avons reporté dans la suite les résultats. Nous pouvons observer que, même si les résultats sont proches l'un de l'autre, il y a quelque différence par exemple sur l'incertitude associé à l'estimation ou sur l'erreur obtenu dans la position finale.

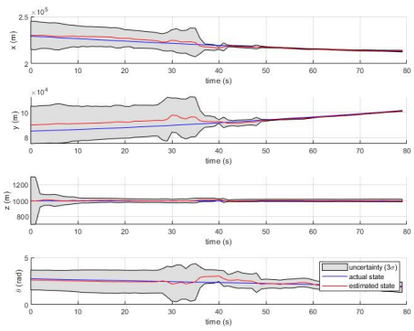
9.1 Ré-échantillonage multinomial

Je vais reporter ci-dessous le code implémenté pour une type de échantillonage multinomial:

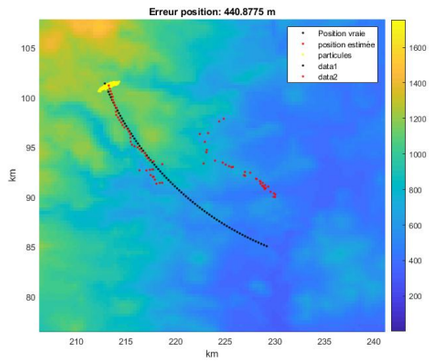
```
function [ind] = multinomial(wp)
N = length(wp);
M = length(wp);
Q = cumsum(wp);
n = 1;
while n <= N
    U = rand;
    m = 1;
    while Q(m) < U
        m = m + 1;
```

```
    end
    ind(n) = m;
    n = n + 1;
end
end
```

Les résultats que j'ai obtenu sont:



(a) États vrais et estimés



(b) Erreur de position sur le schéma

Figure 19: Résultats en utilisant le ré-échantillonage multinomial

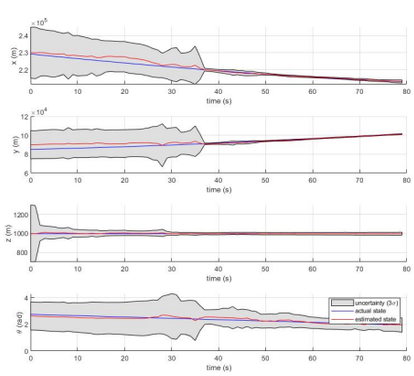
Ce qu'il est possible d'observer c'est que il n'y a pas un forte changement dans ce cas. L'erreur est similaire auquel que l'on obtient avec le code classique même si, pour des valeurs avant la fin de l'algorithme l'estimation atteint des valeurs d'erreur plus bas.

9.2 Ré-échantillonage stratifié

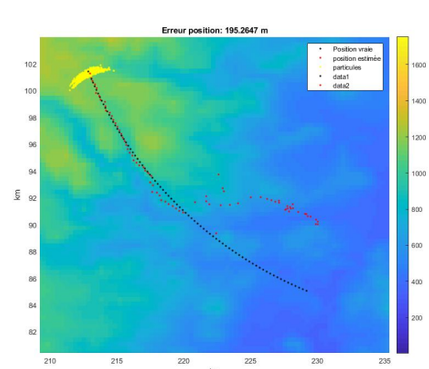
Je vais reporter ci-dessous le code implémenté pour une type de échantillonage stratifié:

```
function [ind] = stratified(wp)
N = length(wp);
M = length(wp);
Q = cumsum(wp);
ind = zeros(1, N);
n = 1;
m = 1;
while(n <= N && m <= M)
    U0 = randn/N;
    U = U0 + n/N;
    while (Q(m) < U && m < M)
        m = m + 1;
    end
    ind(n) = m;
    n = n + 1;
end
end
```

Les résultats que j'ai obtenu sont:



(a) États vrais et estimés



(b) Erreur de position sur le schéma

Figure 20: Résultats en utilisant le ré-échantillonage stratifié

Ce cas c'est le cas avec le résultats meilleurs: on peut observer que l'incertitude se réduit plus rapidement par rapport aux autres deux cases et que l'erreur final est plus petit. On peut dire que, pour notre modèle,

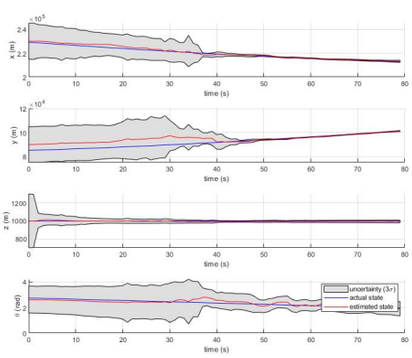
c'est le ré-échantillonage stratifié que nous fait obtenir les résultats meilleurs. Une autre chose qui est visible de la figure 20 c'est que les particules sont plus redistribués par rapport au cas précèdent (figure 19).

9.3 Ré-échantillonage systématique

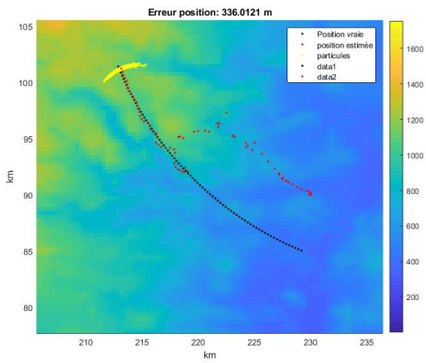
Je vais reporter ci-dessous le code implémenté pour une type de échantillonage systématique:

```
function [ind] = systematic(wp)
N = length(wp);
M = length(wp);
Q = cumsum(wp);
ind = zeros(1, N);
n = 1;
m = 1;
U0 = randn/N;
while(n <= N && m <= M)
    U = U0 + n/N;
    while (Q(m) < U && m < M)
        m = m + 1;
    end
    ind(n) = m;
    n = n + 1;
end
end
```

Les résultats que j'ai obtenu sont:



(a) États vrais et estimés



(b) Erreur de position sur le schéma

Figure 21: Résultats en utilisant le ré-échantillonage systématique

Ici il est possible de voir que l'erreur sur la position dans le moment final est meilleur par rapport à l'échantillonnage multinomial mais pire par rapport auquel stratifié. Pour ce que concerne l'incertitude on peut observer que elle est plus lente à se réduire par rapport au cas précèdent.

Le code

```
clear % effacer toutes les variables
close all % fermer toutes les fentres
clc % effacer la ligne de commande
rng(123457) % imposer la graine de gnration de nombres pseudo-alatoire pour la  rptabilit

% Paramtres  du filtre
N = 3000; % nombre de particules
P_hat = diag([5000, 5000, 100, 20*pi/180].^2); % matrice de covariance initiale
X_hat = [230000, 90000, 1000, 150*pi/180]'; % estim initial
d = size(X_hat,1); % dimension de l'tat
Qf = diag([3, 3, 0.6, 0.001*180/pi].^2); % matrice de covariance de bruit de dynamique
Rf = 20.^2; % matrice de covariance du bruit de mesure du filtre
threshold_resampling = 1; % seuil de r-chantillonnage (theta_eff)

% Paramtres  initiaux et de simulation
erreurInitiale = sqrt(P_hat)*randn(d,1);
X_reel = X_hat + erreurInitiale; % tat vrai initial (inconnu du filtre)
dt = 1; % pas de temps
R = 20.^2; % matrice de covariance du bruit de mesure relle (inconnu du filtre)
dm = size(R,1); % dimension du vecteur de mesures
is_temporalPlot = true;

% Chargement des donnes
load carte.mat
params.pasx_reel = pasx_reel;
params.pasy_reel = pasy_reel;
params.nrow_h = size(h_MNT,1);
params.dxh_MNT = diff(h_MNT,1,2)/pasx_reel;
params.dyh_MNT = diff(h_MNT)/pasy_reel;
params.h_MNT = h_MNT;
params.x_MNT = x_MNT;
params.y_MNT = y_MNT;

% Initialisation du filtre
Xp = X_hat*ones(1,N) + sqrt(P_hat)*randn(d,N); % Tirage des particules autours de X_hat initial
wp = 1/N*ones(1,N); % poids initiaux

% Initialisation des variables de stockage des donnes
tk=1;
t_sim(tk) = 0;
P_sim(:,:,tk) = P_hat;
Pdiag_sim(:,:,tk) = diag(P_hat);
X_reel_sim(:,tk) = X_reel;
X_hat_sim(:,tk) = X_hat;

% Boucle de simulation
T = 80; % dure  (s)
for tk = 2:(T/dt)
    % commande  (dfinit la trajectoire)
    V = 300; % Vitesse
    omega = -0.01; % vitesse angulaire (rad/s)

    % simulation de l'tat vrai (attention, inconnu du filtre)
    t = dt*(tk-1); % temps courant
    X_reel = [X_reel(1)+V*dt*cos(X_reel(4)); X_reel(2)+V*dt*sin(X_reel(4)); X_reel(3);
        X_reel(4)+dt*omega]; % propagation de l'tat rel ( complter)
    X_reel(4) = mod(X_reel(4), 2*pi); % modulo 2pi sur le cap

    % prediction (  complter : variables Xp, X_hat et P_hat)
    Xp = [Xp(1,:)+V*dt*cos(Xp(4,:)); Xp(2,:)+V*dt*sin(Xp(4,:)); Xp(3,:);
        Xp(4,:)+dt*omega*ones(1,N)] + sqrt(Qf)*randn(d,N); % particules prdites
    X_hat = Xp*wp'; % tat  estim  prdit
    X_hat(4,:) = mod(X_hat(4,:), 2*pi); % modulo 2pi sur le cap
    P_hat = ((Xp-X_hat(:,1*ones(1,N))).*wp(ones(d,1),:) )*(Xp-X_hat(:,ones(1,N)))';

    %  gnration  de la mesure relle (   complter )
    Y = X_reel(3) - hobs(X_reel, params) + sqrt(R)*randn(dm,1);

    % validit  de la mesure relle
    is_measurementValid = false;
%    if t < 50 || t > 75
    if mod(t, 0.1) == 0
      is_measurementValid = true;
    end
    % correction (  complter )
```

```
    if is_measurementValid
        for i = 1 : N
            wp(i) = wp(i) *
                exp(-0.5*(Y-(Xp(3,i)-hobs(Xp(:,i),params)))'*inv(Rf)*(Y-(Xp(3,i)-hobs(Xp(:,i),params))));
        end
        wp = wp./sum(wp);
    end

    %  R -chantillonnage
    criterionResampling = 1/(sum(wp.^2)); % critre de rchantillonnage "N efficace"  complter
    if criterionResampling < N*threshold_resampling
        Xp = Xp(:,select(wp)); % slection des nouvelles particules selon l'algorithme de
            r-chantillonnage multinomial
        wp = 1/N*ones(1,N); % r-initialisation des poids
    end

    % enregistrement des variables (pour plot)
    t_sim(tk) = t;
    P_sim(:,:,tk) = P_hat;
    Pdiag_sim(:,:,tk) = diag(P_hat);
    X_reel_sim(:,tk) = X_reel;
    X_hat_sim(:,tk) = X_hat;

    % plot instantan
    if is_temporalPlot
        figure(2)
        clf
        hold on
        imagesc(params.x_MNT*params.pasx_reel/1000, params.y_MNT*params.pasx_reel/1000, h_MNT)
        xlabel('km'); ylabel('km');
        title(['Erreur position: ', num2str(norm(X_hat(1:3) - X_reel(1:3))), ' m'])
        grid
        colorbar
        hold on
        plot(X_reel_sim(1,:)./1000,X_reel_sim(2,:)./1000,'.k')
        plot(X_hat_sim(1,:)./1000,X_hat_sim(2,:)/1000,'.r')
        scatter(Xp(1,:)./1000, Xp(2,:)./1000, '.y')
        legend('Position vraie', 'position estime','particules')
        scatter(X_reel_sim(1,tk)./1000, X_reel_sim(2,tk)./1000, '.k')
        scatter(X_hat_sim(1,tk)./1000, X_hat_sim(2,tk)./1000, '.r')
        grid on
        axis equal
        drawnow
    end
end

% Plot des rsultats
figure(1)
labels = {'x (m)','y (m)','z (m)','\theta (rad)'};
for i = 1:d
    subplot(4,1,i)
    hold on
    fill([t_sim flip(t_sim,2)],[X_hat_sim(i,:) - 3*sqrt(Pdiag_sim(i,:)), X_hat_sim(i,end:-1:1) +
        3*sqrt(Pdiag_sim(i,end:-1:1))], [7 7 7]/8);
    plot(t_sim, X_reel_sim(i,:), 'b')
    plot(t_sim, X_hat_sim(i,:), 'r')
    grid on
    xlabel('time (s)')
    ylabel(labels(i))
end
legend('uncertainty (3\sigma)', 'actual state', 'estimated state')
```