ÉCOLE NATIONALE SUPÉRIEURE DE TECHNIQUES AVANCÉES

TP3 : Filtrage Particulaire

Auteur :
Vanessa LOPEZ

Encadrants :
M. Nicolas MERLINGE

ROB312 - Navigation pour les systèmes autonomes

31 octobre 2023

Table des matières

1 Introduction

On va appliquer **le filtrage particulaire étendu pour des systèmes non gaussiens et non linéaires** sur un programme représentant la trajectoire d'un véhicule, lequel suivre les étapes ci-dessous.

En bref, ce filtre place les particules dans les zones où le véhicule est le plus susceptible de se trouver sur la carte, en augmentant le poids de ces particules et en diminuant ou en éliminant celles qui sont peu probables. Lorsque l'on trouve un arrangement qui modélise la forte probabilité de localisation du véhicule, il s'agit d'une probabilité arbitraire, généralement non gaussienne, d'où l'utilisation de ce type de filtrage.

Étape de prédiction :

— À l'aide du modèle de l'espace d'états du système, on va prédire l'estimation de l'état des particules à l'instant k en fonction de l'estimation à l'instant k-1 et de l'entrée de commande appliquée à l'instant k avec une incertitude (bruit de processus W_k).

— Prédire l'observation à partir de la position de la particule

Étape de mise à jour (correction) : Au cours de cette étape, le poids des particules sera corrigé.

— Calculer la différence entre les mesures réelles du capteur à l'instant k et les prédictions de la position de la particule pour le temps k actuel (Innovation).

— Mise à jour des poids avec l'innovation et les poids précédents.

— Normaliser les poids

Étape d'estimation :

— Calculer une estimation de position actualisée et l'estimation de la covariance pour le temps k.

Resampling :

— Comparer la taille effective de l'échantillon avec le seuil de resampling.

Globalement, le programme crée un environnement de simulation virtuel avec les mesures d'odométrie que le véhicule calcule où une référence et les valeurs de position et d'angle sont entrées avec un peu de bruit pour recréer les mesures qu'un capteur réel prendrait. Ensuite, des points aléatoires (amers) sont générés sur la carte pour simuler les capteurs qui mesurent la position réelle du robot et enfin, l'algorithme du filtre PF sera appliqué pour corriger la trajectoire.

2 Développement du code

2.1 Prediction

Evolution théorique de l'état (modèle dynamique) : On commence à définir le modèle dynamique sur la fonction `motion_model` pour le mouvement du véhicule suivant la configuration décrite ci-dessous en appliquant la prédiction de la position pour chaque particule. On s'appuiera sur la fonction `tcomp` déjà présente dans le programme, qui fournit la dynamique du système en prenant les vecteurs de la position, de la vitesse avec une incertitude représentée par un bruit w_k et de la différence de temps.

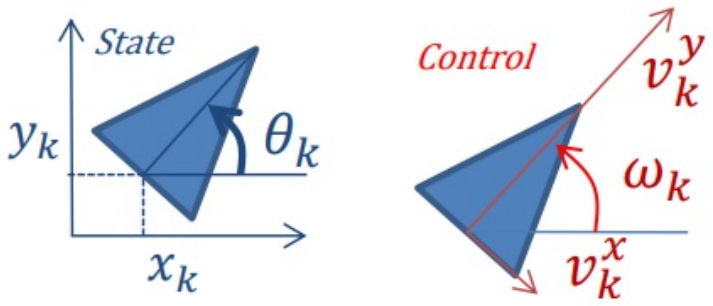


FIGURE 1 – Modèle dynamique

$$
\begin{aligned}
x_k &= f(x_{k-1}, \tilde{u}_k, w_k) \\
x_k &= \begin{bmatrix} x_{k-1} + ((v_k^x + w_k^{V_x}) \cdot cos(\theta_{k-1}) - (v_k^y + w_k^{V_y}) \cdot sin(\theta_{k-1})) \cdot \Delta t \\ y_{k-1} + ((v_k^x + w_k^{V_x}) \cdot sin(\theta_{k-1}) - (v_k^y + w_k^{V_y}) \cdot cos(\theta_{k-1})) \cdot \Delta t \\ \theta_{k-1} + w_k \cdot \Delta t \end{bmatrix}
\end{aligned} \tag{1}
$$

Équation d'observation théorique (modèle de mesure) : Ensuite, avec le modèle d'observation, on utilise l'état pour estimer les mesures des capteurs pour chaque particule qui représentent un vecteur de mesures prédites des capteurs y_k en ajoutant un peu de bruit. La carte des points de amers (représentés par les accords x^P, y^P) est connue a priori. À chaque temps k, un amers est choisi au hasard pour simuler une mesure.

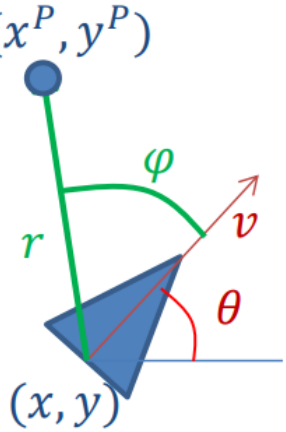


FIGURE 2 – Modèle de capteur

$$y_k = h(x_k) = \begin{bmatrix} r_k \\ \phi_k \end{bmatrix} = \begin{bmatrix} \sqrt{(x_k^P - x_k)^2 + (y_k^P - y_k)^2} \\ atan\frac{y_k^P - y_k}{x_k^P - x_k} - \theta_k \end{bmatrix} \tag{2}$$

2.2 Mise a jour

Lors de la mise à jour, les poids des particules seront renouvelés en fonction de l'équation suivante :

$$\hat{w}_k^i = w_{k-1}^i \cdot \exp\left(-\frac{1}{2}(y_k - h(x_k^i))^T \cdot R_k^{-1} \cdot (y_k - h(x_k^i))\right) \tag{3}$$

On calcule l'innovation qui est la différence entre les observations réelles du capteur et les observations prédites du particules (erreur de prédiction) en se concentrant sur le fait d'augmenter ou diminuer les poids des particules pour se focaliser sur les premiers et ignorer les seconds.

Ces poids sont ensuite normalisés et on calcule la position moyenne et covariance moyenne, en accordant une plus grande priorité aux particules dont les poids sont les plus élevés. Pour cela, les équations suivantes sont utilisées :

$$\hat{x}_k = \sum_{i=1}^{N} w_k^i \cdot x_k^i \tag{4}$$

$$\hat{p}_k = \sum_{i=1}^{N} w_k^i \cdot (x_k^i - \hat{x}_k) \cdot (x_k^i - \hat{x}_k)^T \tag{5}$$

Enfin, si la condition ci-dessous est remplie, on réalise un rééchantillonnage multinomial où N nombres aléatoires indépendants sont générés pour choisir une particule dans l'ancien ensemble comme le montre la figure 3. Sinon, les particules prennent les valeurs des prédictions réalisées pour chacune d'entre elles.

$$N_{eff} = \frac{1}{\sum_i (w_i)^2} < \theta_{eff} \cdot N \tag{6}$$

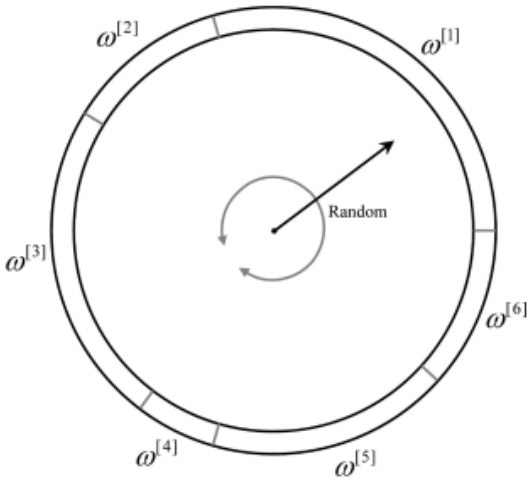


FIGURE 3 – Multinomial resampling, source : [3]

3 Analyses de resultat

Enfin, on peut constater que le PF (filtre particulaire) calcule une moyenne pondérée des mesures des particules prédites afin de converger vers une estimation de l'état réel du système, où chaque particule représentant une hypothèse possible sur l'état du système.

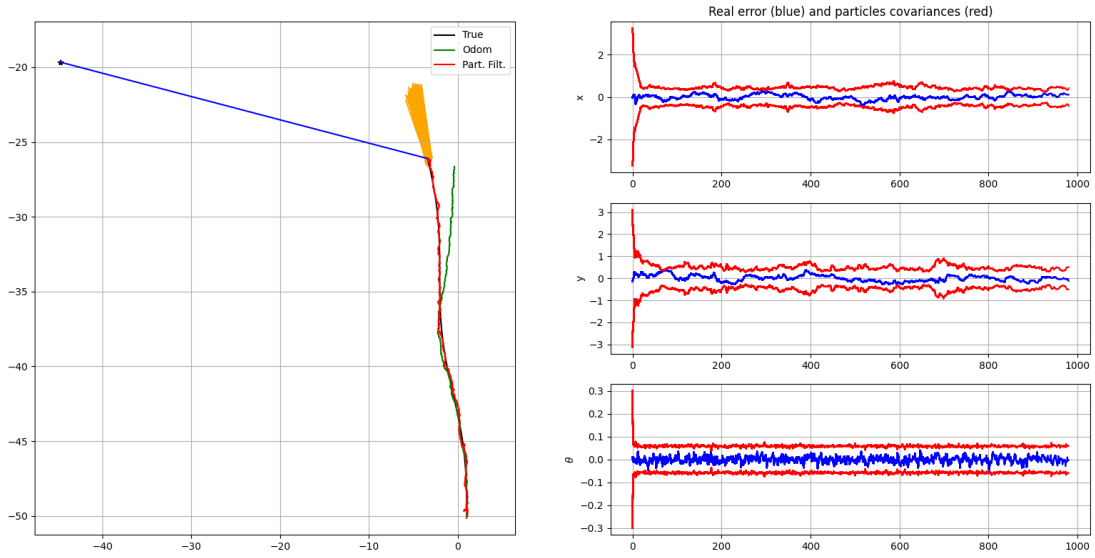
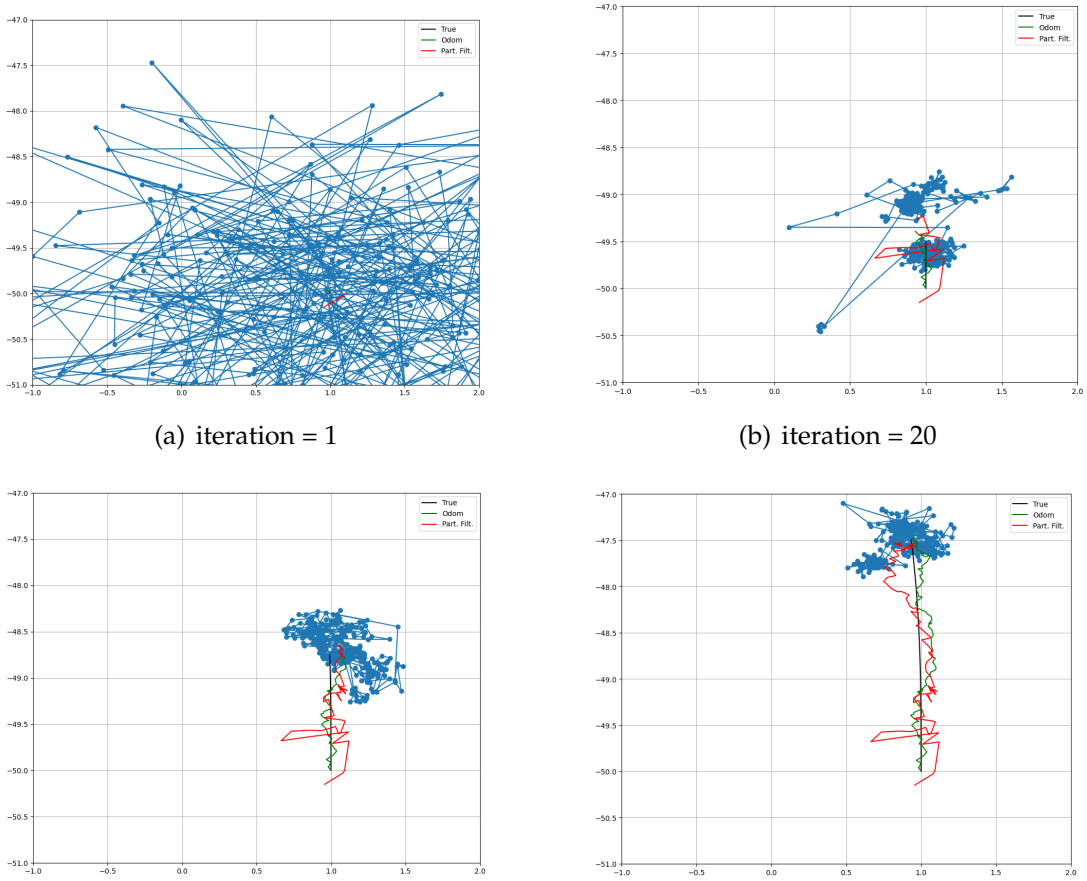


FIGURE 4 – Résultat filtre PF

Il est évident que le filtre particulaire accorde une plus grande priorité à la localisation des amers qu'à l'odométrie. Sur la droite de la figure 4 se trouve l'erreur entre l'état estimé et l'état réel (courbe bleue) et sa covariance (courbe rouge) qui varie de $\pm\ 3\sigma$ et on peut observer que ces courbes sont très bruyantes.

Pour mieux comprendre ce filtre, on a fait un rapprochement sur les itérations initiales pour voir ce que les particules faisaient le long de la trajectoire. Ce processus est illustré à la figure 5.

L'encadré (a) montre que la première itération génère des particules aléatoires. Dans la figure suivante, une centralisation des particules autour de la trajectoire est évidente, mais elles sont encore un peu dispersées. Dans les cases inférieures, les particules sont beaucoup plus concentrées sur la prédiction de la trajectoire.



(a) iteration = 1

(b) iteration = 20

(c) iteration = 50

(d) iteration = 100

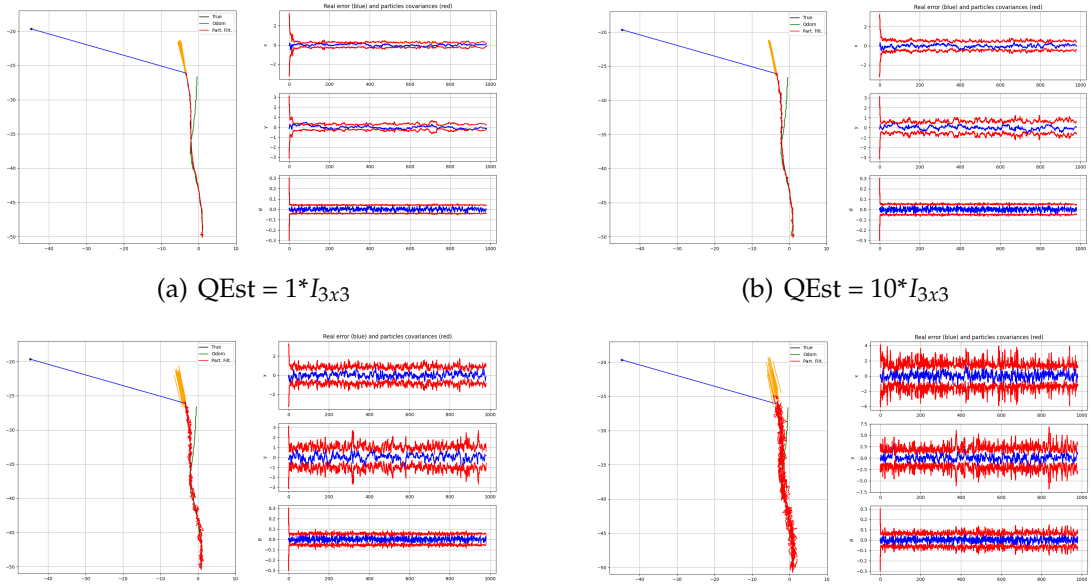
FIGURE 5 – itérations initiales du filtre particulaire

3.1 Bruit de dynamique du filtre

On va faire varier le bruit de dynamique du filtre (matrice QEst)

En augmentant le bruit de la dynamique du filtre et en laissant comme identité la matrice R, on peut observer que la trajectoire contient plus de bruit mais donne la priorité aux mesures des capteurs amers et la covariance augmente parce que le bruit dynamique du filtre continue à s'accumuler.

Dans ce cas, on ajoute du bruit au signal de commande dans le modèle dynamique de l'équation 1 car ce bruit de processus est une distribution gaussienne centrée sur 0 avec une covariance Q, ce qui fait que le modèle continue à donner la priorité aux mesures des capteurs par amers.



(a) QEst = 1*I_{3x3} (b) QEst = 10*I_{3x3}

(c) QEst = 100*I_{3x3} (d) QEst = 1000*I_{3x3}

FIGURE 6 – Modification de bruit de dynamique du filtre

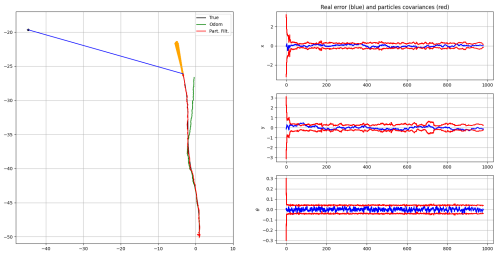
3.2 Bruit de mesure du filtre

On va faire varier le bruit de mesure du filtre (matrice REst)

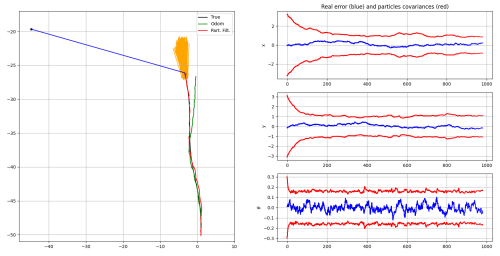
Il est visible dans la figure 7 avec ces modifications que le modèle tend à se détacher de la trajectoire des mesures des capteurs amers et commence à suivre en priorité l'odométrie, car si le R contient des valeurs élevées, étant multiplié inversement par l'erreur des prédictions, ce facteur sera diminué et sachant qu'il se situe à l'intérieur d'une fonction exponentielle négative, pour une petite valeur, on obtiendra une grande valeur. Ce terme est trouvé en multipliant les poids calculés précédemment, ce qui fait que les poids sont répétés sans aucune mise à jour.

Au contraire, si la matrice R est plus petite, on obtiendra une maximisation des erreurs de prédiction et, étant donné une grande valeur dans la fonction exponentielle négative, le résultat sera une petite valeur. Le poids précédent sera multiplié par cette petite valeur, ce qui provoquera une diminution du poids réel des particules et, par conséquent, la disparition des particules présentant une erreur significative.

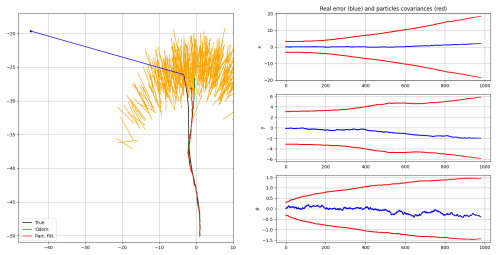
Ainsi, en réalisant le processus de rétention des particules lourdes et d'élimination des particules présentant des erreurs élevées par rapport aux mesures du capteur d'amers.



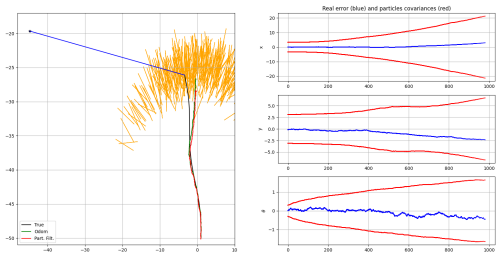
(a) REst = 1*I_{3x3}



(b) REst = 100*I_{3x3}



(c) REst = 1000000*I_{3x3}



(d) REst = 10000000*I_{3x3}

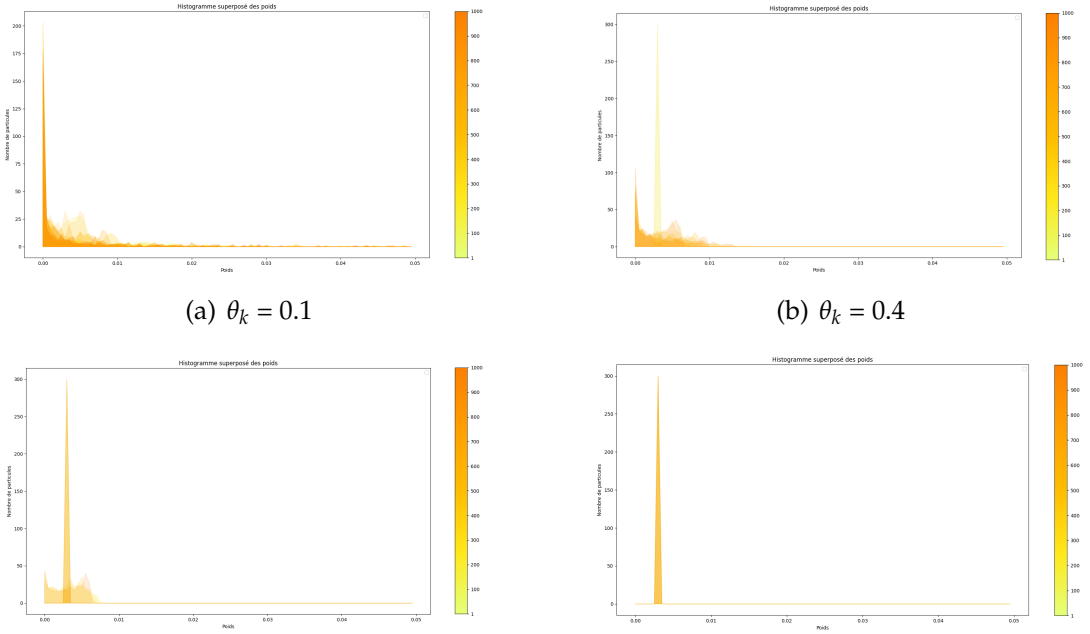
FIGURE 7 – Modification de bruit de dynamique du filtre

3.3 Seuil de ré-échantillonnage

On va faire varier le seuil de ré-échantillonnage θ_{eff} entre 0 et 1

Pour cette variation, des valeurs comprises entre 0 et 1 ont été utilisées pour étudier le comportement des poids des particules tout au long de l'algorithme.

Pour construire ces histogrammes, l'histogramme des poids a été stocké toutes les 100 itérations, puis superposé à des fins de comparaison, les plus clairs (jaunes) représentant les premières itérations et les plus foncés (orange), les dernières.



(a) $\theta_k = 0.1$ (b) $\theta_k = 0.4$

(c) $\theta_k = 0.7$ (d) $\theta_k = 0.9$

FIGURE 8 – Modification de seuil de ré-échantillonnage

On constate que dans les carrés supérieurs de la figure 8, où le thêta effectif ne dépasse pas 0,5, il est possible d'établir les poids dans toutes les itérations, tandis que dans les carrés inférieurs, les dernières itérations (orange) ne sont pas évidentes ou ne parviennent même pas à générer les poids pour toutes les itérations.

Ceci est dû à la condition $N_{eff} < (\theta_{eff} \cdot \mathrm{N})$ qui permet le rééchantillonnage. N_{eff} calcule une estimation du nombre effectif de particules, c'est-à-dire ce critère représente la variance des poids, tandis que θ_{eff} multiplié par le nombre de particules limite cette variance.

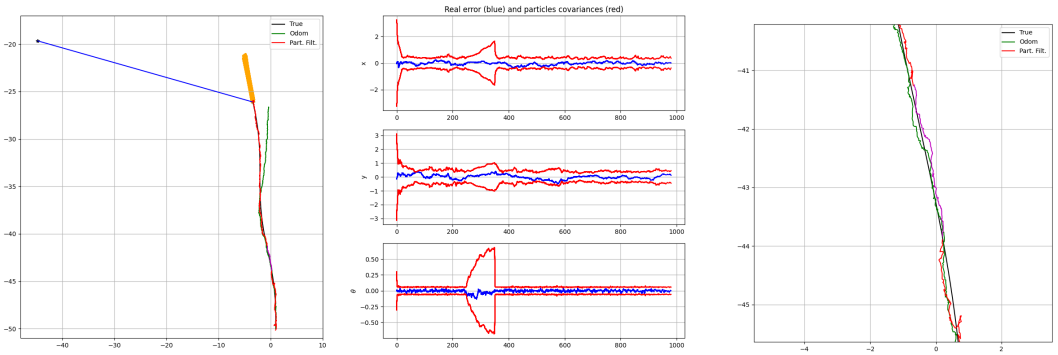
Ainsi, en fixant un nombre élevé (proche de 1) dans θ_{eff}, il est possible de rééchantillonner une grande variance de poids, en supprimant le "filtrage" qu'il effectue pour ignorer ou privilégier certaines particules.

3.4 Trou de mesures

On va simuler un trou de mesures entre t = 250 s et t = 350 s en utilisant la ligne de code suivante :

```
notValidCondition = False if k < 250 or k > 350 else True
```

En faisant un trou entre t=250s et t=350s des mesures, aucun changement dans l'estimation de la trajectoire n'est pas évident, car il s'agit d'un moment où l'odométrie et la trajectoire réelle sont très proches, donc aucune différence importante n'est remarquée.



(a) Résultat avec trou de mesures entre 250s et 350s (b) Zoom sur le trou

FIGURE 9 – Trou de mesures entre t = 250 s et t = 350 s

Cependant, si le trou est plus grand, on constate que le modèle suit l'odométrie parce qu'il n'effectue que la partie prédiction et n'apporte aucune correction au système, donc les poids dépendent purement du modèle d'observation fait pour chaque particule. Tout cela se réduit au fait que l'estimation finale de la trajectoire dépendra de la prédiction générée par la commande d'odométrie et les particules de filtrage.

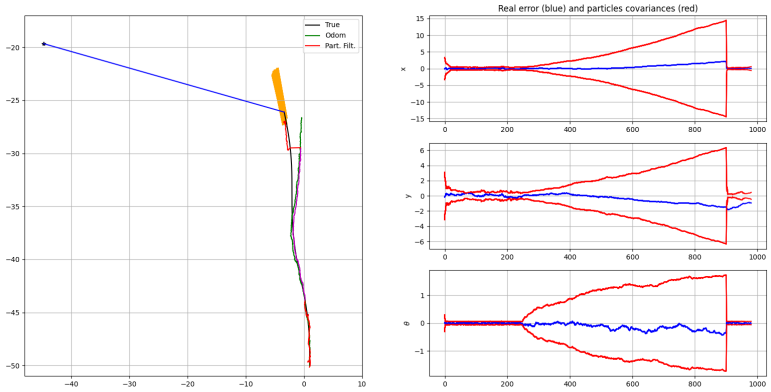
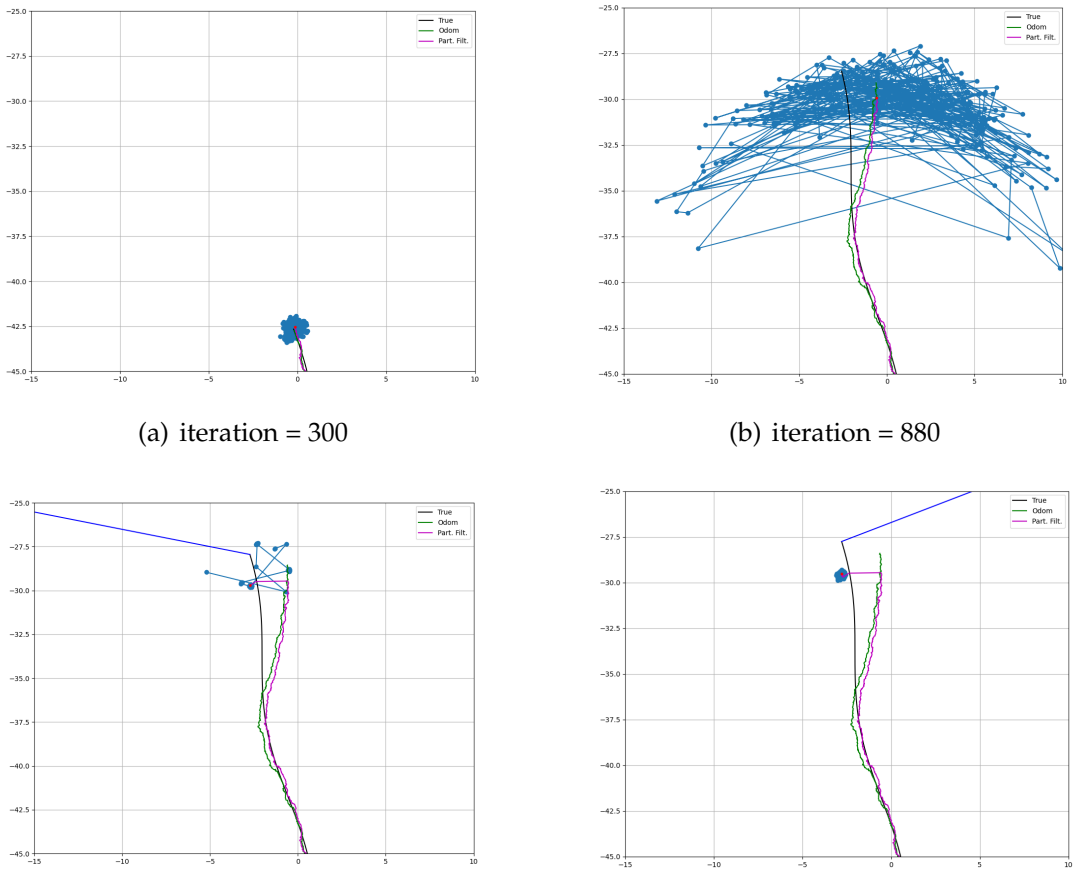


FIGURE 10 – Trou de mesures entre t = 250 s et t = 900 s

Enfin, cela produit au fil du temps une distribution presque aléatoire des particules puisqu'il donne un poids arbitraire à chacune d'entre elles comme on peut voir sur l'évolution des itérations dans la partie supérieure de la figure 11. Lorsque les valeurs mesurées par le capteur amers sont disponibles après l'itération 900, on observe une relocalisation rapide des particules autour de la bonne trajectoire.



(a) iteration = 300

(b) iteration = 880

(c) iteration = 902

(d) iteration = 910

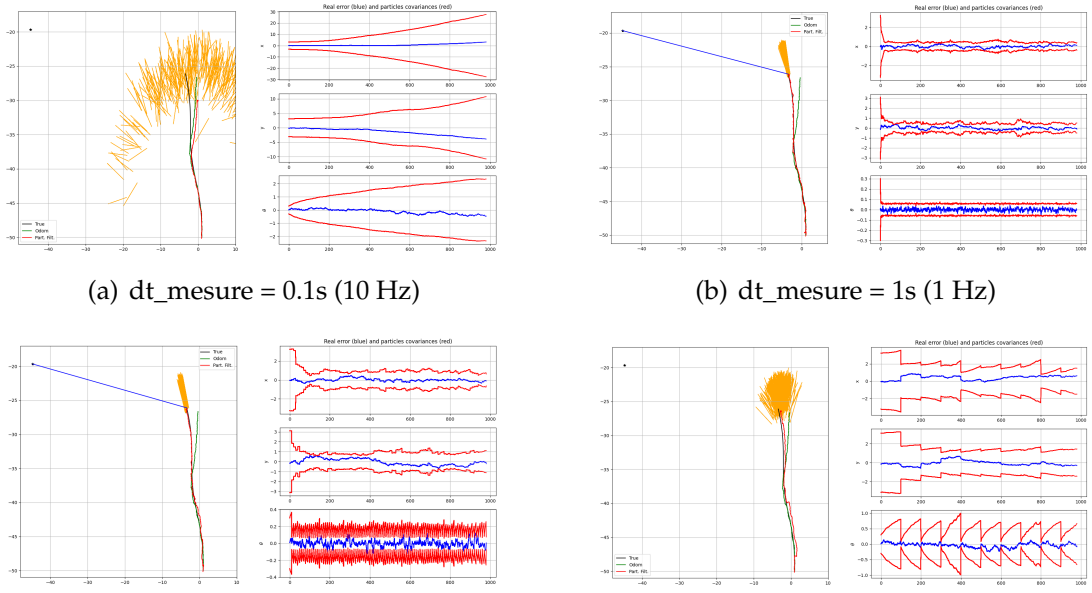
FIGURE 11 – Trou de mesures entre t = 250 s et t = 900 s

3.5 Fréquence des mesures

On va modifier la fréquence des mesures en utilisant la variable `dt_mesure`

La variable `dt_mesure` est utilisée pour obtenir le module par rapport aux itérations du programme (1000) dans la méthode `get_observation` pour obtenir le modèle d'observation avec un certain bruit. Rappelons que ce modèle utilise l'état (ou l'état prédit au temps suivant) pour estimer (ou prédire) les mesures des capteurs. Par conséquent, si on augmente le temps pendant lequel ces mesures sont calculées et prédites, la trajectoire sera moins précise.

Les figures ci-dessous montrent qu'à mesure que la fréquence d'échantillonnage diminue, la covariance du modèle augmente de manière significative, car les conditions préalables du modèle sont étroitement espacées et tendent à produire des estimations médiocres. La figure 12 (d) présente des pics de variance importants, car l'incertitude est grande pendant les périodes sans échantillonnage, mais lorsque l'échantillonnage est effectué, l'incertitude diminue considérablement.



(a) dt_mesure = 0.1s (10 Hz)

(b) dt_mesure = 1s (1 Hz)

(c) dt_mesure = 10s (0.1 Hz)

(d) dt_mesure = 100s (0.01 Hz)

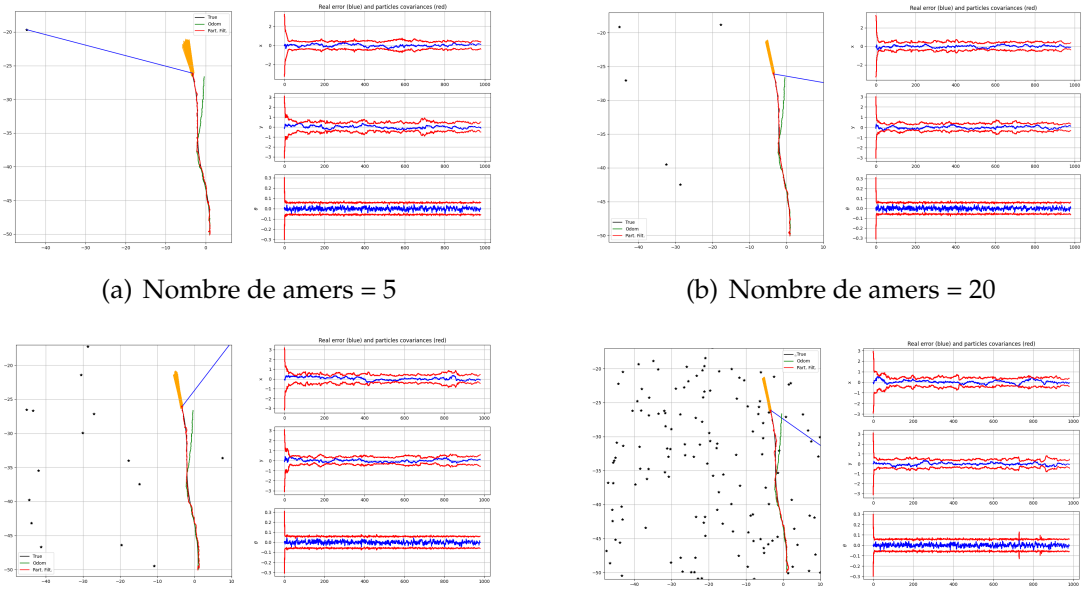
FIGURE 12 – Modification de fréquence des mesures

L'augmentation de la fréquence des mesures ne serait pas très optimale comme on peut voir sur la figure 12 (a), car on prendrait plus d'échantillons que le nombre de fois où une prédiction est faite (dt_pred=1) et pour certaines valeurs, le temps de prédiction ne coïncide pas avec le temps d'échantillonnage, donc le module ne sera jamais égal à 0 et ne générerait pas le modèle d'observation.

3.6 Nombre d'amers

On va faire varier le nombre d'amers et étudier les performances du filtre en fonction du nombre d'amers. Ensuite, on réglerait le filtre pour obtenir les meilleures performances possibles.

Une légère diminution de la variance de y peut être observée lorsque l'on augmente le nombre d'amers de 20. Dans ce modèle, on peut remarquer que la variance n'atteint pas 1, contrairement à ce qui se passe dans le modèle avec 5 amers.



(a) Nombre de amers = 5

(b) Nombre de amers = 20

(c) Nombre de amers = 100

(d) Nombre de amers = 1000

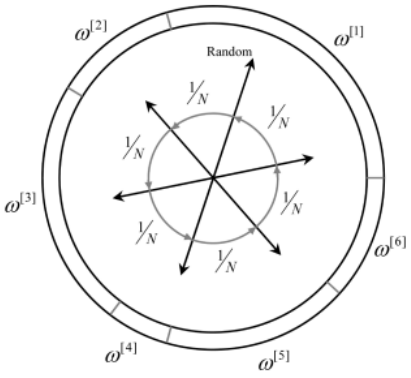
FIGURE 13 – Modification de nombre d'amers

On peut observer qu'il n'y a pas d'amélioration significative du filtre particulaire avec l'augmentation du nombre d'amers. Il n'y a donc pas de quantité optimale de points amers, mais ce qui nous intéresse le plus est leur distribution, puisque les prédictions seront beaucoup plus précises si certains points sont proches de la trajectoire effectuée par le robot, par opposition à des mesures de capteurs éloignées de cette trajectoire.

3.7 Autre façon de ré-échantillonner les poids

On va proposer une autre façon de ré-échantillonner les poids, en remplacement de la fonction `re_sampling`

Systematic resampling : Cette méthode ne tire qu'un seul nombre aléatoire, c'est-à-dire une direction dans la "roue", les N -1 autres directions étant fixées à des incréments de 1 N par rapport à cette direction choisie au hasard comme il est montré la figure 14 (a).



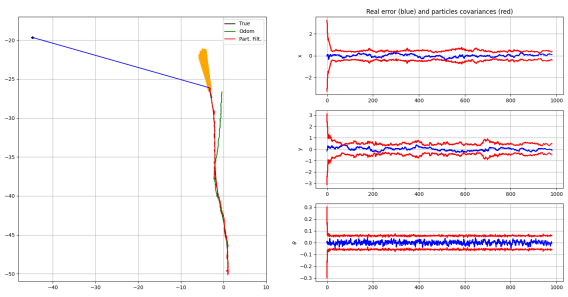
(a) Illustration, Source [3]

$xx[] = SR(x, w, N)$
$j = 0,\ accumW = w[j]$
$u = rand()/N$
for $i = 0..(N-1)$
  while $accumW < u$
    $j = j + 1$
    $accumW = accumW + w[j]$
  end
  $xx[i] = x[j]$
  $u = u + 1/N$
end

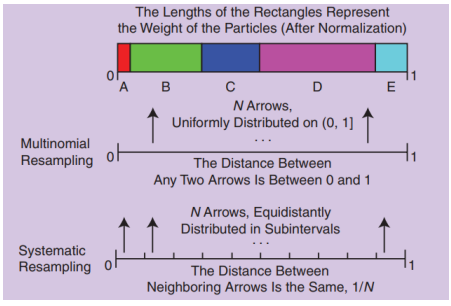
(b) Pseudo-code, Source [2]

FIGURE 14 – Systematic resampling

A l'aide du pseudocode de la figure 14 (b),on a obtenu les résultats de la figure ci-dessous (a), qui montrent que que les deux méthodes de rééchantillonnage donnent de bons résultats, bien que la manière de sélectionner les données change, puisque dans le rééchantillonnage multinomial, la distance entre deux flèches observées dans la figure 15 (b) est comprise entre 0 et 1 et que dans le rééchantillonnage systématique, la distance entre les flèches voisines est la même.



(a) Résultat avec Systematic resampling



(b) Méthodes, Source [1]

FIGURE 15 – Résultats et comparaison avec Systematic resampling

Références

[1] Tianqing LI, Miodrag BOLIC et Petar M DJURIC : Resampling methods for particle filtering : Classification, implementation, and strategies. *IEEE Signal Processing Magazine*, 32(3):70–86, 2015.

[2] T.Y. LIM, Yeong FAI, E. SU, Shaekh SHITHIL, S.F. CHIK, F. DUAN et P.J.H. CHIN : Enhanced localization with adaptive normal distribution transform monte carlo localization for map based navigation robot. *ELEKTRIKA- Journal of Electrical Engineering*, 18:17–24, 12 2019.

[3] Suresh Kumar VEERAMALLA et Venkata Krishna Hari Rama TALARI : Multiple dipole source localization of eeg measurements using particle filter with partial stratified resampling. *Biomedical Engineering Letters*, 10(2):205–215, 2020.