

TP2: Filtrage de Kalman appliqué à la navigation et à l'estimation d'état

RO12 - Navigation pour les systèmes autonomes

Gallego, Natalia
AST
ENSTA Paris
natalia.gallego@ensta-paris.fr

I. INTRODUCTION

Dans la navigation des véhicules autonomes, les systèmes d'estimation d'état jouent un rôle crucial pour l'intégration et l'analyse des informations provenant de divers capteurs, afin de déterminer la position et l'orientation du véhicule. Le filtre de Kalman étendu (EKF) est une technique couramment utilisée pour estimer les états non linéaires, combinant les prédictions d'un modèle dynamique avec des observations de capteurs bruyants.

Ce travail pratique vise à mettre en œuvre et à régler un EKF pour estimer la position d'un véhicule, en utilisant les informations des capteurs d'odométrie et des mesures simulées supplémentaires, telles que les plages de référence et les angles par rapport à certains points connus de l'environnement.

Le projet est structuré en plusieurs étapes, où les effets de différentes configurations EKF sont explorés, tels que la variation de la fréquence de mesure, la dynamique et le bruit du capteur, et la disponibilité de certaines mesures. De plus, le comportement du filtre est évalué par rapport aux interruptions des mesures et aux modifications du nombre de points de référence disponibles. Le code de simulation comprend un modèle de mouvement du véhicule, la génération de mesures et le filtre EKF, dont les équations clés doivent être complétées pour un fonctionnement correct.

Tout au long du TP, les résultats obtenus en ajustant les matrices de covariance du filtre seront analysés et commentés, simulant diverses conditions de mesure pour évaluer ses performances dans différents scénarios.

II. QUESTION 1 : EXPLICATION INITIALE DU CODE (SIMULATION DE VÉHICULE, CAPTEURS, ODOMÉTRIE, FILTRE DE KALMAN).

get_odometry() : l'odométrie fait référence à la façon dont le robot estime sa position au fil du temps en fonction du mouvement précédent.

simulation_world() : C'est l'évolution de la position réelle du robot : L'état réel du robot (x_{True}) est mis à jour à chaque pas à l'aide d'un contrôle basé sur une trajectoire sinusoïdale.

get_observation() : modélise les capteurs, qui simulent les observations de caractéristiques (points de repère) dans l'environnement du robot.

motion_model(x , u , dt_pred) : Cette fonction est utilisée pour prédire le nouvel état du robot, étant donné l'état actuel (x), le contrôle appliqué (u) et l'intervalle de temps de prédiction (dt_pred). L'état du robot (x) inclut sa position et son orientation, tandis que le contrôle (u) décrit les vitesses linéaires et angulaires.

observation_model($xVeh$, $iFeature$, Map) : génère une prédiction des mesures observées en fonction de l'état du véhicule ($xVeh$) et de la position des entités sur la carte (Map). Le résultat de cette fonction est une observation simulée qui inclut la distance et l'angle de ce point de référence par rapport au robot.

get_obs_jac($xPred$, $iFeature$, Map) : calcule la matrice jacobienne de la fonction d'observation par rapport à l'état prédit du robot ($xPred$). Il est utilisé pour linéariser le modèle d'observation.

F(x , u , dt_pred) : Calcule la matrice jacobienne du modèle de mouvement par rapport à l'état du robot.

G(x , u , dt_pred) : Cette fonction calcule la matrice jacobienne du modèle de mouvement par rapport au bruit dans les commandes du robot. Le bruit dans les commandes représente les incertitudes ou les erreurs associées aux actions du robot.

tcomp(tab , tbc , dt) : Cette fonction effectue une composition de transformations entre deux états. Prend l'état actuel du robot (tab) et lui applique une transformation de mouvement (tbc), en utilisant le pas de temps dt . Cela permet de mettre à jour la position et l'orientation du robot en fonction des commandes appliquées et du temps écoulé.

plot_covariance_ellipse : trace une ellipse qui représente l'incertitude dans l'estimation de la position du robot. A partir de la matrice de covariance estimée ($PEst$), l'ellipse décrit la plage probable d'erreurs dans l'estimation de la position du robot dans l'espace.

Filtre de Kalman : Le filtre de Kalman est une méthode d'estimation qui combine les prédictions du modèle de mouvement du robot avec les observations.

III. QUESTION 2 : CODE COMPLET

Les codes manquants ont été complétés afin d'implémenter le filtre de Kalman sur la trajectoire du robot. Vous trouverez ci-dessous les fonctions développées et une brève explication de celles-ci.

```
1 def motion_model(x, u, dt_pred):
2     xPred = np.zeros((3, 1))
3     xPred[0, 0] = x[0, 0] + u[0, 0] * dt_pred * cos(
4         x[2, 0])
5     xPred[1, 0] = x[1, 0] + u[0, 0] * dt_pred * sin(
6         x[2, 0])
7     xPred[2, 0] = x[2, 0] + u[2, 0] * dt_pred
8     xPred[2, 0] = angle_wrap(xPred[2, 0])
9     return xPred
```

motion_model : met à jour la position en x en utilisant la vitesse en x et l'orientation actuelle. Met à jour la position y en utilisant la vitesse x et l'orientation actuelle. Met à jour l'orientation en utilisant la vitesse angulaire. Enfin, ajustez l'orientation pour qu'elle soit comprise entre $-\pi$ et π à l'aide de la fonction `angle_wrap` et renvoyez l'état prédit `xPred`.

```
1 def observation_model(xVeh, iFeature, Map):
2     dx = Map[0, iFeature] - xVeh[0, 0]
3     dy = Map[1, iFeature] - xVeh[1, 0]
4     z = np.zeros((2, 1))
5     z[0, 0] = sqrt(dx**2 + dy**2)
6     z[1, 0] = atan2(dy, dx) - xVeh[2, 0]
7     z[1, 0] = angle_wrap(z[1, 0])
8     return z
```

observation_model : La fonction prend trois paramètres :

- **xVeh** : l'état du véhicule (x, y, θ).
- **iFonction** : index de l'entité observée sur la carte.
- **Carte** : La carte contenant les positions de toutes les entités.

Avec cela la fonction :

- 1) Calcule la différence des coordonnées x (dx) et y (dy) entre la position du robot et la position de l'entité observée.
- 2) Calcule la distance euclidienne (`z[0, 0]`) entre le robot et l'entité.
- 3) Calcule l'angle (`z[1, 0]`) entre la ligne reliant le robot et la caractéristique et l'orientation du robot.

Pour définir ces équations, on a utilisé les informations vues au cours, où la matrice x_k est donnée par l'expression:

$$x_k = \begin{pmatrix} x_{k-1} + (\tilde{v}_k^x \cos(\theta_{k-1}) - \tilde{v}_k^y \sin(\theta_{k-1}))\Delta t \\ y_{k-1} + (\tilde{v}_k^x \sin(\theta_{k-1}) + \tilde{v}_k^y \cos(\theta_{k-1}))\Delta t \\ \theta_{k-1} + \tilde{\omega}_k \Delta t \end{pmatrix} \quad (1)$$

```
1 def get_obs_jac(xPred, iFeature, Map):
2     dx = Map[0, iFeature] - xPred[0, 0]
3     dy = Map[1, iFeature] - xPred[1, 0]
4     q = dx**2 + dy**2
5     jH = np.zeros((2, 3))
6     jH[0, 0] = -dx / sqrt(q)
7     jH[0, 1] = -dy / sqrt(q)
8     jH[1, 0] = dy / q
9     jH[1, 1] = -dx / q
10    jH[1, 2] = -1
```

```
11    return jH
12
13 def F(x, u, dt_pred):
14     theta = x[2, 0]
15     Jac = np.eye(3)
16     Jac[0, 2] = -u[0, 0] * dt_pred * sin(theta)
17     Jac[1, 2] = u[0, 0] * dt_pred * cos(theta)
18     return Jac
19
20 def G(x, u, dt_pred):
21     theta = x[2, 0]
22     Jac = np.zeros((3, 3))
23     Jac[0, 0] = dt_pred * cos(theta)
24     Jac[0, 1] = -dt_pred * sin(theta)
25     Jac[1, 0] = dt_pred * sin(theta)
26     Jac[1, 1] = dt_pred * cos(theta)
27     Jac[2, 2] = dt_pred
28     return Jac
```

get_obs_jac :

- 1) Calcule la différence de coordonnées x (dx) et y (dy) entre la position du robot et la position de l'entité observée.
- 2) Calculez q, qui est la somme des carrés de dx et dy.
- 3) Calcul des dérivées partielles de la distance et de l'angle par rapport aux coordonnées du robot.
- 4) Calcule la dérivée partielle de l'angle par rapport à l'orientation du robot.

F : Calcule les dérivées partielles de la fonction de transition par rapport à l'orientation du robot (θ).

G : Calcule les dérivées partielles de la fonction de transition par rapport au bruit aux entrées de commande (u).

Pour cela, les définitions des matrices `jH`, `F` et `G` du filtre de Kalman étendu ont été utilisées.

$$F_k = \frac{\partial f}{\partial x} = \begin{pmatrix} 1 & 0 & -V_x \cdot \Delta t \cdot \sin(\theta) \\ 0 & 1 & V_x \cdot \Delta t \cdot \cos(\theta) \\ 0 & 0 & 1 \end{pmatrix} \quad (2)$$

$$G_k = \frac{\partial f}{\partial w} = \frac{\partial f}{\partial u} = \begin{pmatrix} \Delta t \cdot \cos(\theta) & -\Delta t \cdot \sin(\theta) & 0 \\ \Delta t \cdot \sin(\theta) & \Delta t \cdot \cos(\theta) & 0 \\ 0 & 0 & \Delta t \end{pmatrix} \quad (3)$$

```
1 # Kalman prediction
2 xPred = motion_model(xEst, u_tilde, simulation.
3     dt_pred)
4 F_x = F(xEst, u_tilde, simulation.dt_pred)
5 G_u = G(xEst, u_tilde, simulation.dt_pred)
6 PPred = F_x @ PEst @ F_x.T + G_u @ QEst @ G_u.T
7
8 # compute Kalman gain - with dir and distance
9 Innov = z - zPred # observation error (
10    innovation)
11 Innov[1, 0] = angle_wrap(Innov[1, 0])
12 S = REst + H @ PPred @ H.T
13 K = PPred @ H.T @ np.linalg.inv(S)
14
15 # Compute Kalman gain to use only distance
16 Innov = z[0:1, :] - zPred[0:1, :] # observation
17    error (innovation)
18 H = H[0:1, :]
19 S = H @ PPred @ H.T + REst[0:1, 0:1]
20 K = PPred @ H.T @ np.linalg.inv(S)
21
22 # Compute Kalman gain to use only direction
```

```

21 Innov = z[1:2, :] - zPred[1:2, :] # observation
    error (innovation)
22 Innov[0, 0] = angle_wrap(Innov[0, 0])
23 H = H[1:2, :]
24 S = H @ PPred @ H.T + REst[1:2, 1:2]
25 K = PPred @ H.T @ np.linalg.inv(S)

```

Ce code fait partie de la prédiction de l'état et de la covariance de l'état à l'aide du modèle de mouvement du robot et des entrées de contrôle. Le modèle de mouvement du robot est utilisé pour prédire l'état futur (xPred) en fonction de l'état actuel (xEst) et des entrées de contrôle (u_tilde).

Pour cela, les équations du filtre de Kalman étendu pour les systèmes nonlinéaires ont été utilisées :

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1}, u_k) \quad (4)$$

$$\hat{P}_{k|k-1} = F_k \hat{P}_{k-1} F_k^T + G_k Q_k G_k^T \quad (5)$$

$$H_k = \left. \frac{\partial h}{\partial x} \right|_{x=\hat{x}_k} \quad (6)$$

$$S_k = R_k + H_k \hat{P}_{k|k-1} H_k^T \quad (7)$$

$$K_k = \hat{P}_{k|k-1} H_k^T S_k^{-1} \quad (8)$$

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k (y_k - h(\hat{x}_{k|k-1})) \quad (9)$$

$$\hat{P}_k = (I_d - K_k H_k) \hat{P}_{k|k-1} \quad (10)$$

Calculer le gain de Kalman - Avec direction et distance : calcule le gain de Kalman en utilisant à la fois la distance et la direction de l'observation. La différence entre l'observation réelle (z) et l'observation prédite (zPred) est calculée. La matrice d'innovation (S) est calculée avec la covariance de l'observation et le produit des matrices jacobiennes et la covariance prédite. Calcul du gain de Kalman (K).

Calculer le gain de Kalman pour utiliser uniquement la distance : Cette section calcule le gain de Kalman en utilisant uniquement la distance d'observation. Calcul de l'innovation en utilisant uniquement la distance (Innov) : La différence entre la distance observée et la distance prédite est calculée. De plus, les matrices Innov et S ainsi que le gain K sont calculés. La matrice jacobienne (H) est réduite pour ne considérer que la distance.

Calculer le gain de Kalman pour utiliser uniquement la direction : Calculez le gain de Kalman en utilisant uniquement la direction de l'observation. La différence entre la distance observée et la distance prévue est calculée. De plus, les matrices Innov et S ainsi que le gain K sont calculés. La matrice H est également réduite.

A. Résultats calculant Kalman avec distance et direction

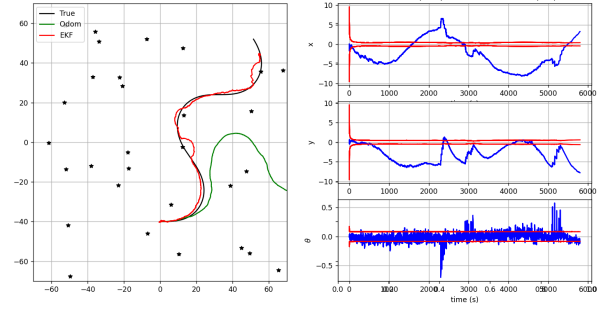


Fig. 1. Résultats calculant Kalman avec distance et direction

On constate que par rapport à la trajectoire calculée uniquement par odométrie, le filtre de Kalman améliore grandement la précision de la prédiction de trajectoire du robot. Mieux suivre la trajectoire. Concernant les erreurs des prédictions, nous pouvons voir que x et y sont tous deux en dehors des limites définies avec l'écart type. D'un autre côté, la prédiction de l'angle θ est assez précise et se comporte dans les limites de l'écart type.

IV. QUESTION 3 : MODIFICATION DE LA FRÉQUENCE DES MESURES

La valeur de dt_meas est l'intervalle de temps entre deux mises à jour de mesure, c'est-à-dire la fréquence à laquelle les observations de l'environnement sont obtenues. Par conséquent, changer cette valeur modifiera la fréquence du système et aura des implications importantes sur les performances et la précision du filtre.

Les résultats sont visibles dans les figures 2 et 3.

A. dt_meas est très petit (0.1 s) :

Les mises à jour des mesures nécessitent un calcul (calculer la matrice de gain de Kalman et mettre à jour les covariances). Avec un très petit dt_meas , le système peut être surchargé de calculs de mise à jour, affectant l'efficacité en temps réel. De plus, si les capteurs sont bruyants ou imprécis, des mesures fréquentes introduiront continuellement du bruit dans le système, ce qui pourrait avoir un impact négatif sur la précision des estimations si le filtre n'est pas bien réglé pour gérer le bruit. Dans ce cas, nous pouvons observer que le robot n'a pas bougé de manière significative entre les mises à jour, ce qui signifie que le filtre de Kalman peut continuer à s'ajuster pour maintenir la position estimée très proche du point initial.

B. dt_meas est très grand (5 s et 10 s) :

Les mesures seront obtenues moins fréquemment, ce qui obligera le filtre de Kalman à s'appuyer principalement sur des prédictions de mouvement basées sur un modèle entre les mesures. Cela amène le système à accumuler des erreurs en raison de l'incertitude de la prédiction. De plus, si le robot subit des changements brusques ou un glissement, ces

changements ne seront pas détectés tant qu'une mesure n'est pas effectuée, ce qui peut entraîner un écart important entre la position estimée et la position réelle.

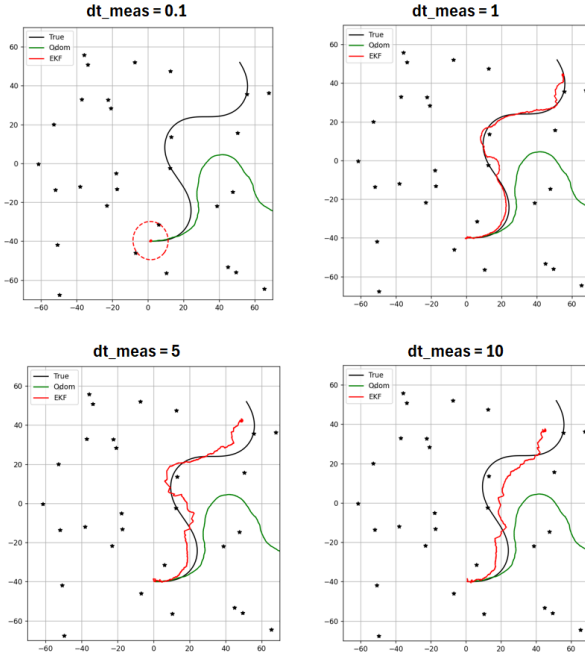


Fig. 2. Résultats de trajectoire pour différentes fréquences

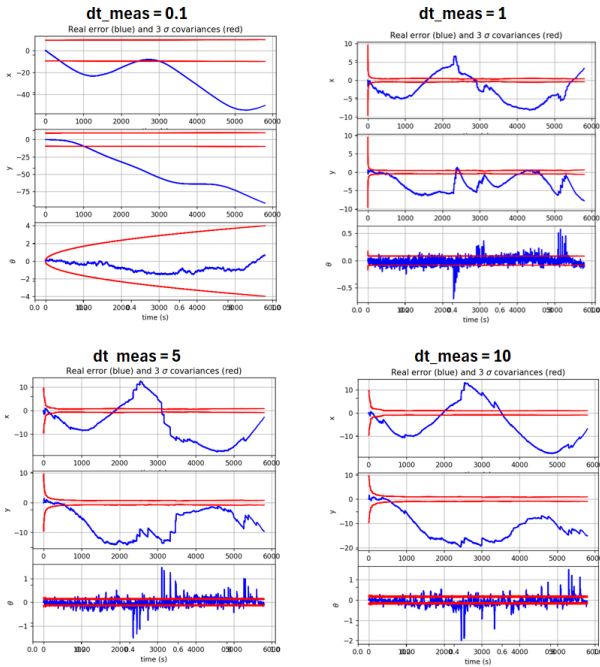


Fig. 3. Prédiction de trajectoire pour différentes fréquences

V. QUESTION 4: VARIATION DU BRUIT DE DYNAMIQUE DU FILTRE (MATRICE Q_{Est})

La valeur de Q_{Est} représente l'estimation de l'incertitude dans le modèle de mouvement du robot (bruit de processus).

Sa valeur peut affecter de manière significative la fusion des estimations odométriques et des mesures des capteurs :

A. Q_{Est} est très petit :

Le filtre de Kalman s'appuiera trop sur le modèle de mouvement du robot et n'accordera pas suffisamment de poids aux mesures du capteur, car le filtre estime que le modèle de mouvement est très précis et que la prédiction basée sur l'odométrie est fiable. Cela peut signifier que si le modèle de mouvement du robot n'est pas précis ou si l'odométrie comporte des erreurs, l'estimation du filtre peut s'éloigner de la position réelle du robot. C'est ce dernier que l'on voit dans les résultats, où la trajectoire finale (ligne rouge) est largement éloignée de la trajectoire réelle du robot.

B. Q_{Est} est très grand :

Le filtre de Kalman s'appuiera moins sur le modèle de mouvement du robot (odométrie) et davantage sur les mesures des capteurs, car le filtre croira que le modèle de mouvement comporte beaucoup d'incertitude et que ses prédictions ne sont donc pas fiables. Le problème est que si les mesures du capteur sont bruyantes ou inexactes, le filtre de Kalman pourrait trop réagir à ces mesures, provoquant une instabilité ou des réactions excessives dans l'estimation de la position. Cependant, nous pouvons voir que dans ce cas, en augmentant Q_{Est} , l'estimation de la trajectoire s'améliore grandement.

Ce qui précède nous permet de conclure que le modèle de mouvement (odométrie) est assez imprécis et comporte une erreur très élevée, il vaut donc mieux dans ce cas s'appuyer davantage sur les capteurs, qui semblent avoir des mesures suffisamment correctes.

Cela est logique puisque sur les figures on voit que l'odométrie (ligne verte) est très éloignée de la trajectoire réelle.

De plus, comme nous pouvons le voir sur les graphiques, les prédictions s'améliorent à mesure que Q_{Est} grandit, entrant entre les limites imposées par la covariance.

Les résultats sont visibles dans les figures 4 et 5.

VI. QUESTION 5: VARIATION DU BRUIT DE MESURE DU FILTRE (MATRICE R_{Est})

Cette question sera réalisée en utilisant un Q_{Est} de 1, afin de voir clairement comment la valeur de R_{Est} affecte les résultats.

La valeur de R_{Est} représente la matrice de covariance du bruit de mesure. Ce paramètre reflète le degré de confiance que vous avez dans les mesures du capteur.

A. R_{Est} est très petit :

Cela indique que les mesures des capteurs sont supposées être très précises, de sorte que le filtre de Kalman accordera plus de poids aux mesures des capteurs et moins de poids à la prédiction du modèle de mouvement. Cela présente le risque que si les capteurs présentent réellement du bruit ou des erreurs, le filtre de Kalman pourrait sur-ajuster ces mesures, conduisant à des estimations erratiques.

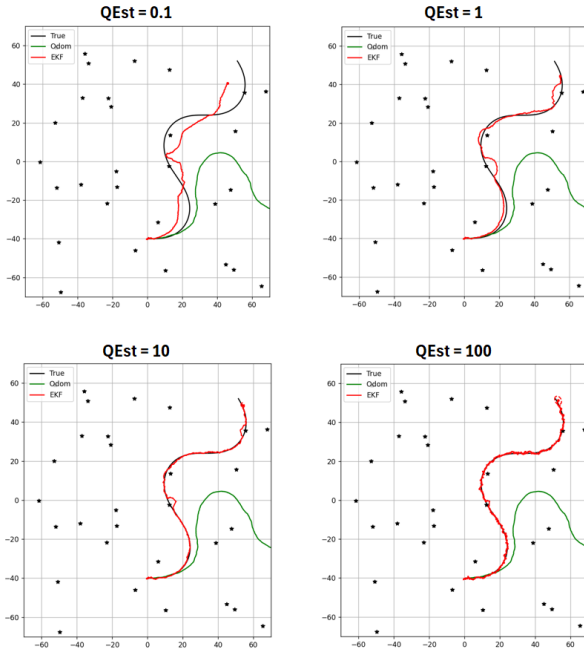


Fig. 4. Résultats de trajectoire pour différentes valeurs de QEst

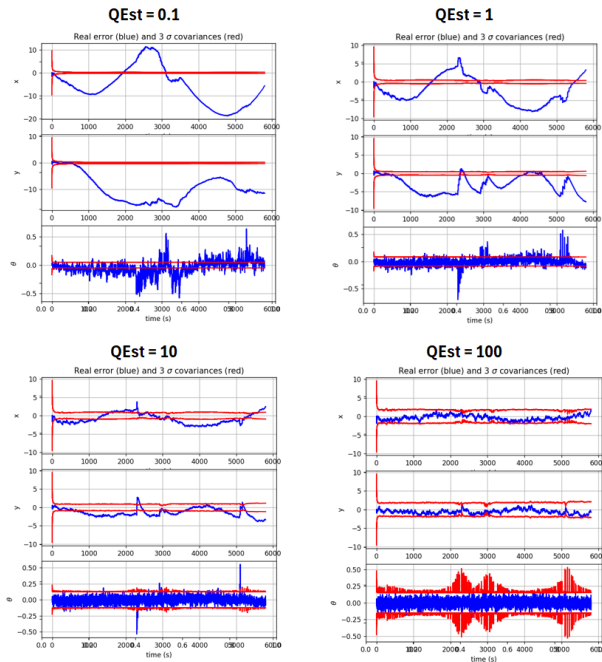


Fig. 5. Prédiction pour différentes valeurs de QEst

B. REst est très grand :

Cela indique que les mesures du capteur sont considérées comme très bruyantes ou imprécises. De ce fait, le filtre de Kalman accordera moins de poids aux mesures des capteurs et s'appuiera davantage sur la prédiction d'état (odométrie). Cela peut avoir pour conséquence que si le modèle de mouvement n'est pas précis ou a accumulé des erreurs, le filtre peut ne pas bien corriger la trajectoire ou l'état du système.

Nous pouvons alors conclure la même chose que dans la question précédente, où nous avons visualisé que les résultats se sont améliorés car nous nous sommes davantage appuyés sur les capteurs que sur l'odométrie du système. Cela peut être dû au fait que l'odométrie comporte des inexactitudes et ne constitue pas une bonne source d'informations pour estimer l'état du système.

De plus, comme nous pouvons le voir sur les graphiques, les prédictions s'améliorent à mesure que REst diminue, entrant entre les limites imposées par la covariance.

Les résultats sont visibles dans les figures 6 et 8.

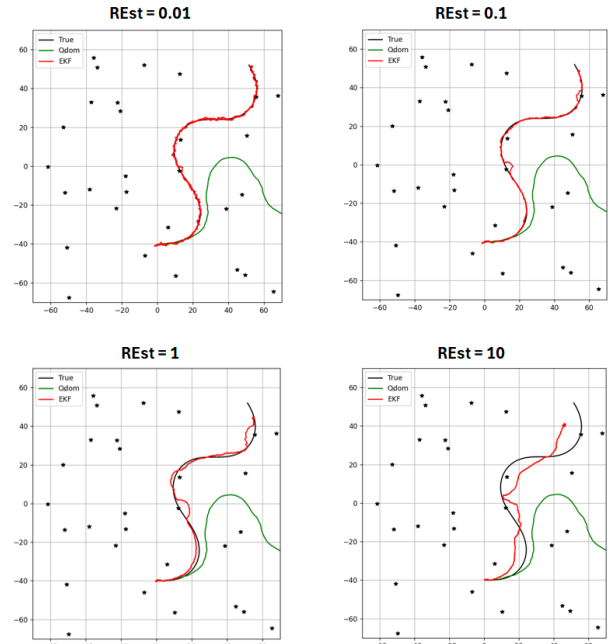


Fig. 6. Résultats de trajectoire pour différentes valeurs de REst

VII. QUESTION 6 : ÉCART DE MESURE ENTRE $T = 2500$ S ET $T = 3500$ S AVEC `notValidCondition`

```
1 if k*self.dt_pred % self.dt_meas == 0:
2     notValidCondition = (2500 <= k*self.dt_pred <=
3         3500)
4     if notValidCondition:
5         z = None
6         iFeature = None
7     else:
8         iFeature = np.random.randint(0, self.Map.
9             shape[1] - 1)
10        zNoise = np.sqrt(self.RTrue) @ np.random.
11            randn(2)
12        zNoise = np.array([zNoise]).T
13        z = observation_model(self.xTrue, iFeature,
14            self.Map) + zNoise
15        z[1, 0] = angle_wrap(z[1, 0])
```

Pour les questions suivantes, QEst et REst ont été définis comme 1, afin de maintenir l'uniformité de l'étude.

En écrivant `notValidCondition = (2500 <= k*self.dt_pred <= 3500)`, nous définissons une plage de temps spécifique pour marquer une condition comme

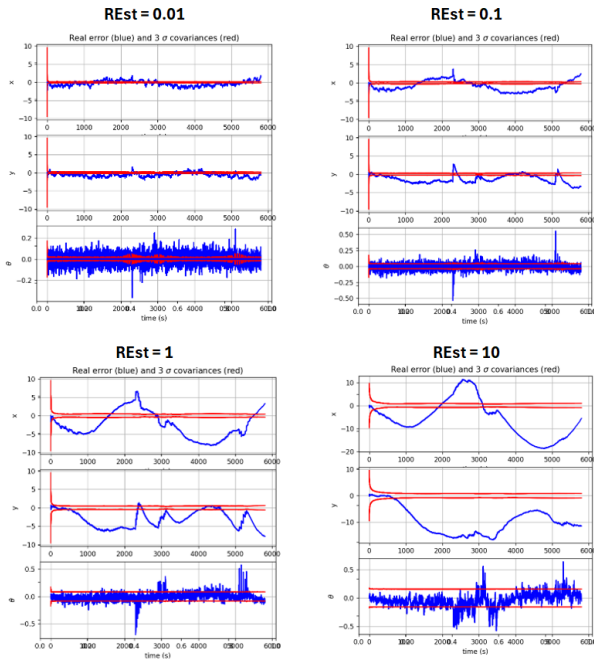


Fig. 7. Prédiction pour différentes valeurs de REst

invalide. Cela permet de simuler l'absence de mesures de capteur dans cet intervalle de temps.

Cela divise nos graphiques en trois moments différents, qui seront expliqués.

A. Avant l'intervalle de temps ($t < 2500$ s) :

Les graphiques d'erreur montrent les corrections du filtre de Kalman basées sur les observations des capteurs et de l'odométrie, comme vu dans les questions précédentes.

B. Pendant l'intervalle de temps ($2500 \text{ s} \leq t \leq 3500 \text{ s}$) :

Aucune observation ne sera disponible, le filtre de Kalman fera donc uniquement des prédictions basées sur le modèle de mouvement, dont nous savons d'après les résultats précédents qu'il ne donne pas de résultats très corrects. Nous pouvons voir que l'incertitude dans les estimations augmente en raison du manque de corrections basées sur les observations et nous observons également que les graphiques d'erreur montrent une augmentation de l'écart de l'erreur réelle par rapport à l'estimation. Par conséquent, dans les délais impartis (lignes verticales vertes), les courbes de prédiction commencent à s'éloigner rapidement de la valeur attendue.

C. Après l'intervalle de temps ($t > 3500$ s) :

Les observations des capteurs reprennent, permettant au filtre de Kalman d'effectuer à nouveau des corrections. Pour cette raison, et connaissant l'importance des mesures des capteurs dans ce système, nous voyons comment l'incertitude des estimations diminue à mesure que davantage de corrections sont apportées sur la base des observations.

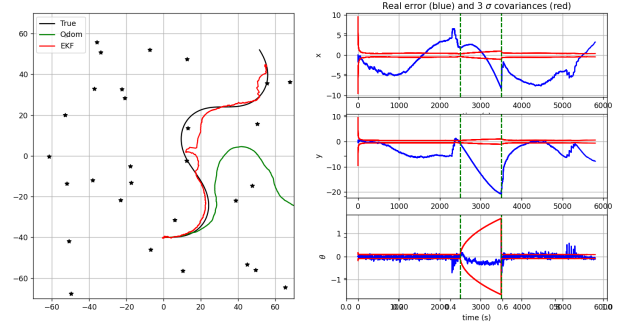


Fig. 8. Erreurs de prédiction pour différentes valeurs de REst

VIII. QUESTION 7 : VARIEZ LE NOMBRE DE POINTS DE RÉFÉRENCE.

Pour les questions suivantes, la modification apportée à la question précédente a été éliminée, de sorte que le système obtient à nouveau des mesures pendant l'ensemble des temps.

Les $n_{Landmarks}$ représentent le nombre de points de repère ou de caractéristiques de l'environnement que le robot peut observer. Le filtre de Kalman s'appuie sur ces éléments pour réduire l'incertitude de l'état du robot en fusionnant les données des capteurs avec la prédiction du modèle de mouvement du robot.

Par conséquent, le nombre de points de référence affecte le compromis du filtre de calme, comme le montrent les figures 9 et 10. Les résultats sont expliqués ci-dessous.

A. Réduction des $n_{Landmarks}$:

Avec moins de points de référence, le robot est moins capable d'observer une caractéristique et de corriger sa position. Cela peut conduire à de plus grandes incertitudes dans l'estimation de l'état du robot, surtout si le robot se déplace pendant de longues périodes sans observer de point de référence.

Dans les résultats, nous pouvons voir qu'avec 5 points de référence, on obtient la courbe la plus lisse, qui suit correctement la trajectoire réelle. Mais en le comparant avec les autres cas on voit que dans ce premier exemple, il ne corrige pas sa position pour se rapprocher de la courbe réelle de manière plus optimale. On sacrifie donc un peu de précision, mais on obtient un mouvement du robot plus fluide.

B. Augmentation des $n_{Landmarks}$:

Lorsqu'il y a plus de points de repère dans l'environnement, le robot a plus de possibilités d'observer une caractéristique, ce qui entraîne des corrections plus fréquentes de sa position estimée, ce qui permet d'espérer une précision de localisation améliorée.

Au fur et à mesure que le nombre de points augmente, la courbe de trajectoire calculée se rapproche de plus en plus de la courbe réelle. Cependant, cela amène le robot à se déplacer de manière plus erratique alors qu'il tente de corriger sa trajectoire de plus en plus fréquemment.

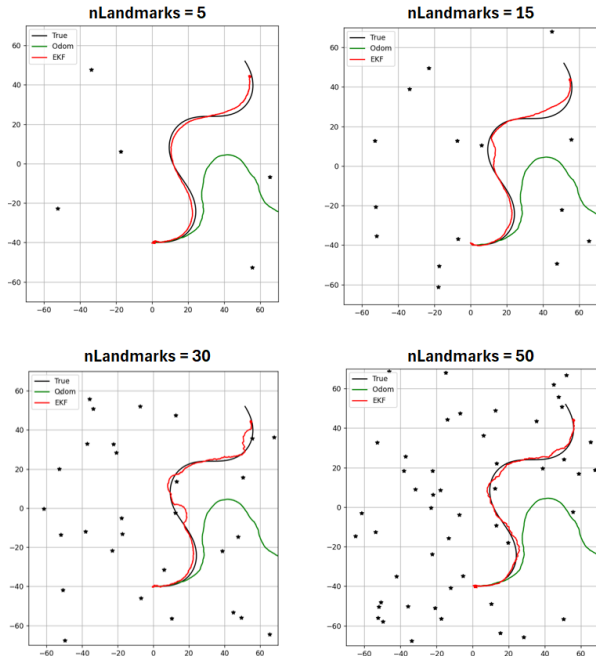


Fig. 9. Résultats de trajectoire pour différentes valeurs de nLandmarks

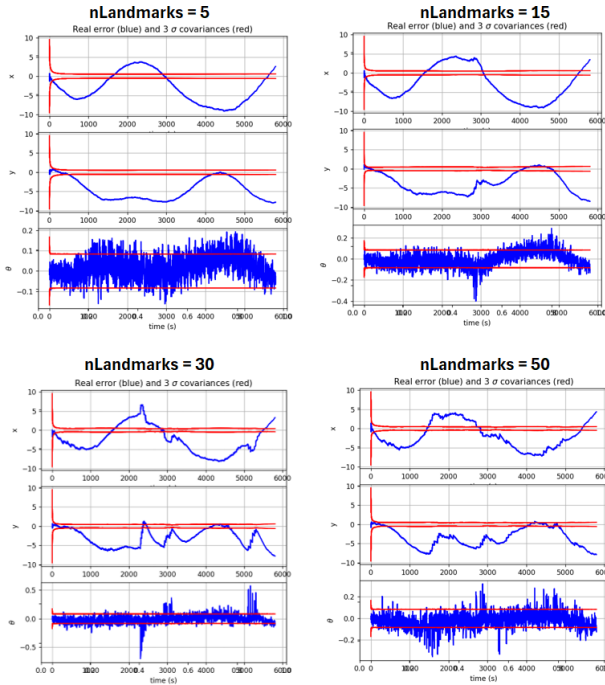


Fig. 10. Prédictions pour différentes valeurs de nLandmarks

IX. QUESTION 8 : RÉSULTATS CALCULANT KALMAN UNIQUEMENT AVEC LA DISTANCE

Dans cette méthode on se concentre uniquement sur la distance pour calculer le gain de Kalman, on utilise donc uniquement la première partie de l'observation ($z[0:1, :]$), qui correspond à la distance. En conséquence, nous pouvons deviner qu'ignorer la direction peut conduire à des

erreurs d'estimation, surtout si le robot se trouve dans un environnement où la direction est cruciale pour l'interprétation correcte des données. Cependant, comme on peut le voir dans les résultats de la figure 11, en les comparant avec les résultats de la figure 1, la courbe se comporte de manière plus fluide dans le temps. Et bien qu'il soit évident que la courbe est un peu plus éloignée de la trajectoire réelle, par rapport au cas initial, elle présente également un comportement suffisamment correct. Cela nous permet de conclure que la recherche du robot pour suivre l'angle défini est ce qui le rend moins fluide.

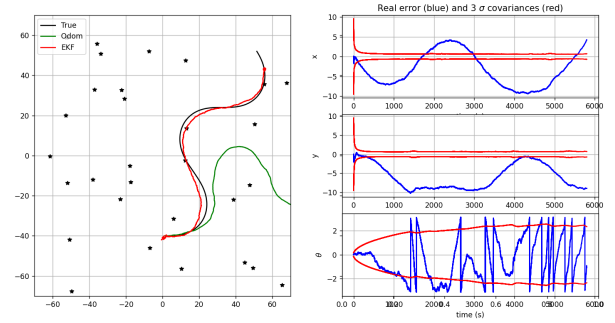


Fig. 11. Résultats calculant Kalman uniquement avec la distance

A. Variez le nombre de points de référence

De la même manière que précédemment, nous avons voulu voir comment le nombre de points de référence affectait le résultat du filtre de Kalman.

On voit que les résultats sont très similaires à ceux obtenus avec le filtre de Kalman en utilisant à la fois la distance et la direction. Dans ce cas, le nombre de points améliore le suivi de la trajectoire originale. Cependant, cette amélioration est assez faible à mesure que les points augmentent, donc le temps de calcul que peut fournir l'ajout de points supplémentaires au système doit également être pris en compte.

Pour voir les résultats rendez-vous sur les figures 12 et 13.

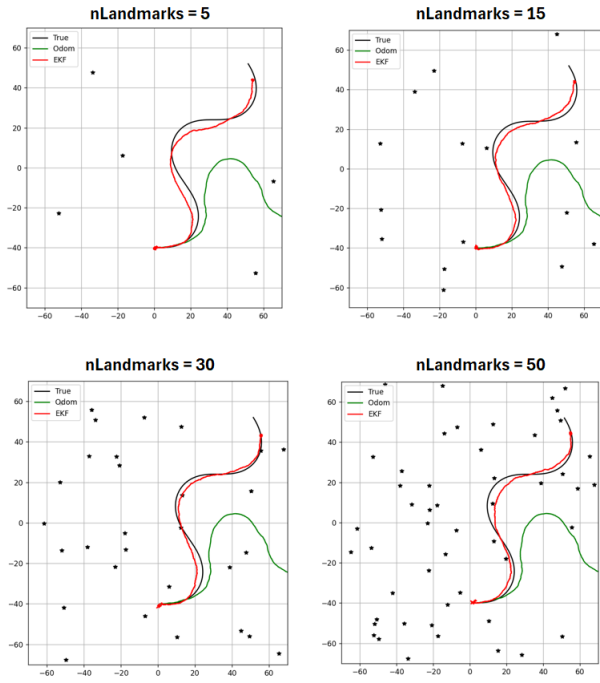


Fig. 12. Résultats utilisant uniquement des points de repère à distance variable

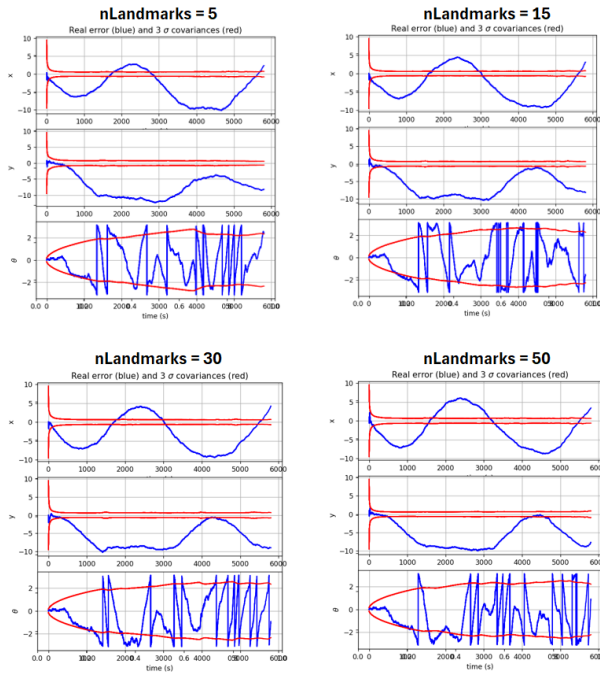


Fig. 13. Prédiction utilisant uniquement des points de repère à distance variable

B. Variez les valeurs de Q_{Est} et R_{Est}

De la même manière que précédemment, nous avons voulu voir comment le changement des valeurs de confiance Q_{Est} et R_{Est} affectait le comportement du filtre de Kalman.

Dans ce cas particulier, nous pouvons voir que varier les valeurs de R_{Est} ne change pas beaucoup les résultats obtenus,

contrairement à ce que nous avons vu avec la configuration précédente, où cette valeur affectait les résultats obtenus. Nous pouvons donc voir que la suppression de la direction fait que le niveau de confiance dans les capteurs n'affecte pas directement la trajectoire calculée avec Kalman.

En revanche, nous avons le cas inverse avec les résultats en faisant varier Q_{Est} , car cela génère de grands changements dans les résultats obtenus. Par conséquent, la valeur de la confiance que nous accordons aux valeurs du modèle de système est devenue importante.

Ces résultats ont du sens car toutes les observations sensorielles ne sont pas utilisées (seule la distance est utilisée), la valeur de R_{Est} , qui est liée aux capteurs, affecte moins le résultat final.

En revanche, la matrice de covariance du bruit du modèle de processus (Q_{Est}) a un impact beaucoup plus important. En effet, en s'appuyant uniquement sur la distance pour les corrections, la confiance dans le modèle interne (qui inclut l'odométrie et d'autres aspects du système) devient beaucoup plus cruciale. Puisque vous n'avez plus la possibilité d'effectuer des corrections supplémentaires, toute incertitude dans le modèle affecte directement la trajectoire calculée.

Pour voir les résultats, rendez-vous sur les figures 14 et 15.

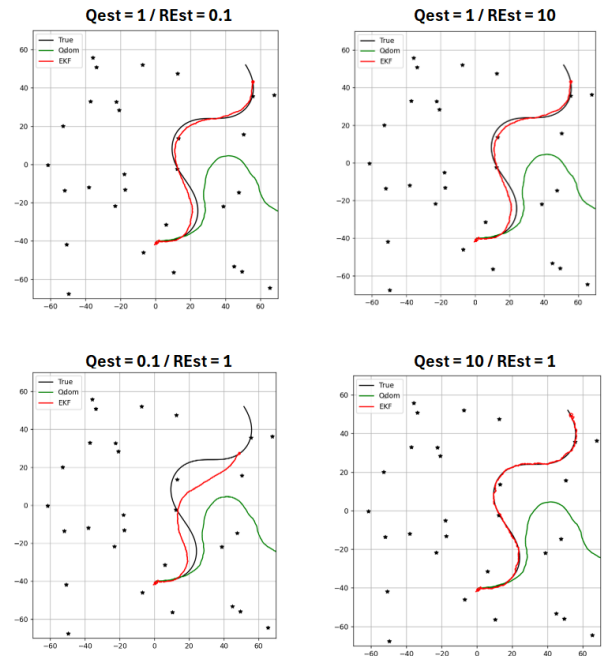


Fig. 14. Résultats utilisant uniquement la distance variant Q_{Est} et R_{Est}

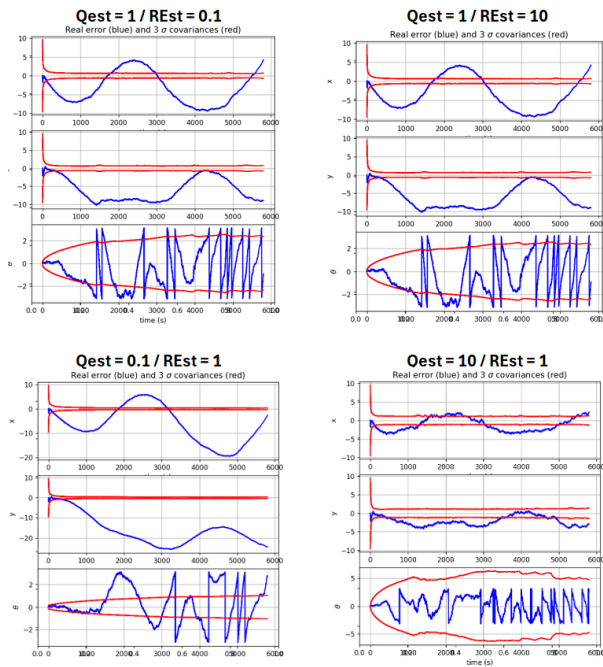


Fig. 15. Prédiction utilisant uniquement des distances variables QEst et REst

X. QUESTION 9 : RÉSULTATS CALCULANT KALMAN UNIQUEMENT AVEC LA DIRECTION

Cette méthode se concentre uniquement sur la direction, en utilisant la deuxième partie de l'observation ($z[1:2, :]$), qui correspond à l'angle. Comme dans le cas précédent, l'omission de la distance peut rendre l'estimation inexacte, en particulier dans les scénarios où la distance est importante pour la localisation.

Contrairement au cas précédent, comme on peut le voir dans les résultats de la figure 16, en les comparant avec les résultats de la figure 1, la courbe se comporte de manière similaire. Dans la figure actuelle, nous pouvons voir que la courbe présente les mêmes mouvements brusques du robot, qui sont également augmentés par rapport au cas de la question 1. Pour cette raison, à certains endroits du parcours, la courbe s'éloigne un peu de la réalité trajectoire.

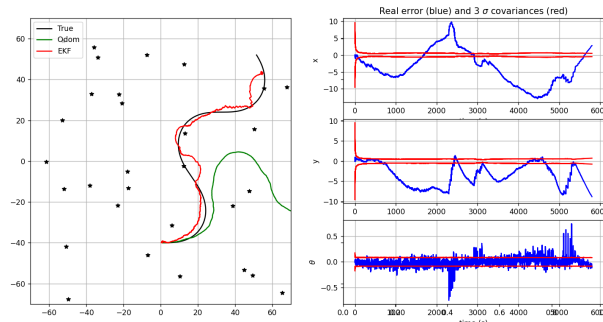


Fig. 16. Résultats calculant Kalman uniquement avec la direction

A. Variez le nombre de points de référence

De la même manière que précédemment, nous avons voulu voir comment le nombre de points de référence affectait le résultat du filtre de Kalman.

Contrairement aux deux cas précédents, dans cette situation on voit que le nombre de points de référence affecte grandement le résultat final de la trajectoire. Dans les cas où les points sont réduits (5 et 15 points), on constate que le système est incapable de calculer correctement la trajectoire du robot, ce qui entraîne sa perte totale. Lorsque le nombre de points augmente (30 ou 50), le système peut mettre à jour sa trajectoire et obtient un résultat similaire à la trajectoire réelle, qui s'améliore à mesure que le montant augmente.

Pour voir les résultats rendez-vous sur les figures 17 et 18

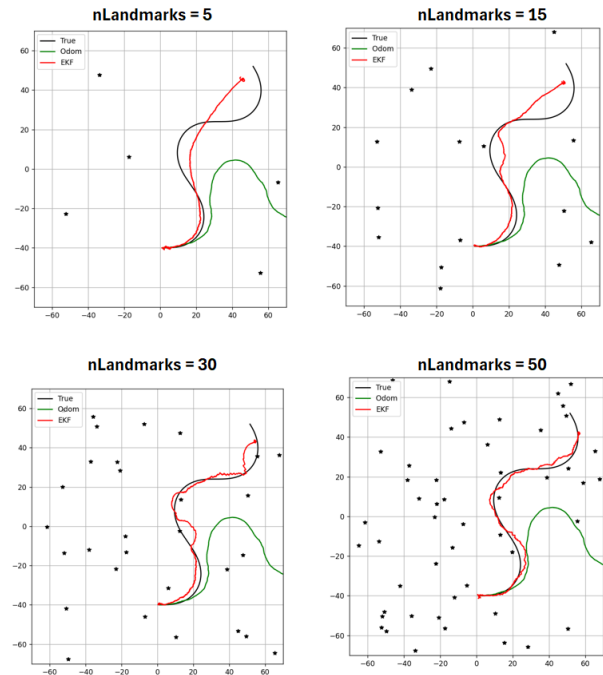


Fig. 17. Résultats utilisant uniquement l'adresse des différents points de repère

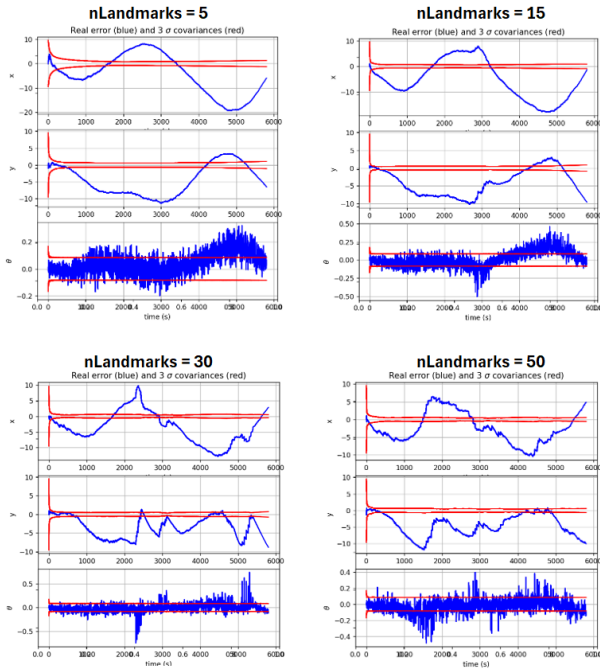


Fig. 18. Prédiction utilisant uniquement la direction variable des points de repère

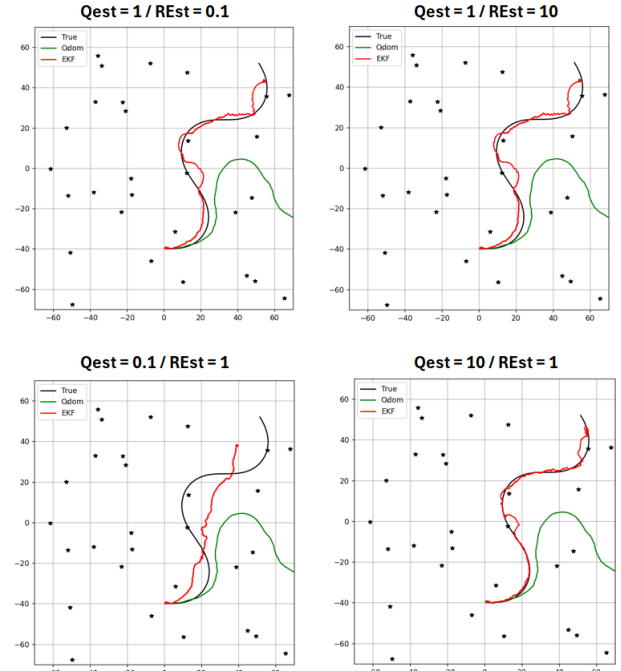


Fig. 19. Résultats utilisant uniquement la direction variant QEst et REst

B. Variez les valeurs de QEst et REst

De la même manière que précédemment, nous avons voulu voir comment le changement des valeurs de confiance QEst et REst affectait le comportement du filtre de Kalman.

De la même manière que précédemment, on constate que varier les valeurs de REst a un impact minime sur les résultats obtenus.

De plus, nous observons le même comportement précédent lorsque QEst varie, car cela entraîne des changements importants dans les résultats obtenus. Cela met en évidence l'importance de la confiance accordée aux valeurs du modèle de système.

Comme auparavant, en n'utilisant pas toutes les observations sensorielles, la valeur de REst, qui est liée à la confiance dans les capteurs, affecte moins le résultat final. En revanche, le QEst a un impact plus important car le système dépend exclusivement de la gestion correctionnelle.

Pour voir les résultats, rendez-vous sur les figures 19 et 20.

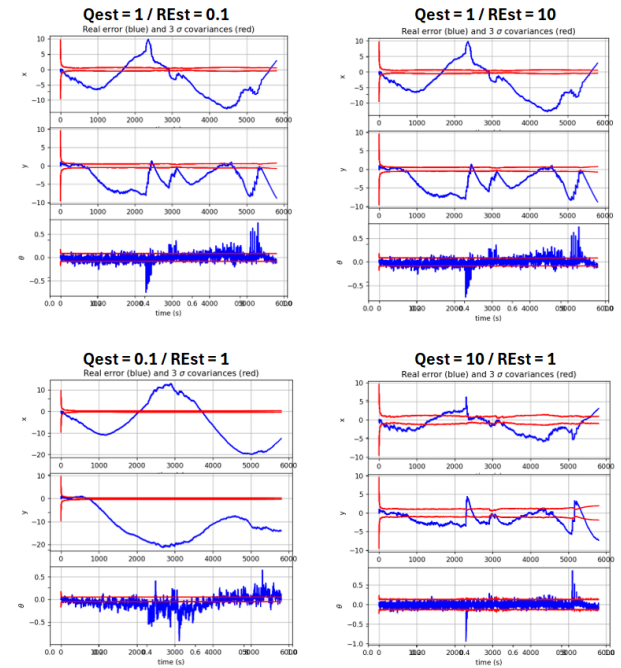


Fig. 20. Prédiction utilisant uniquement des direction variables QEst et REst

XI. GITHUB

Vous pouvez vous référer à la référence suivante pour voir les codes de nœud du projet: [GitHub/RO12/TP2](https://github.com/RO12/TP2)