# Filtre de Kalman et utilisation

## Part 1 : Filtre de Kalman et régulation du robot JOG

Le robot JOG était un robot de type char utilisé en UV 2.7 SPID comme plateforme d'application à des cours de découverte de la robotique et de ce qu'il y a autour. Après environ 5 ans de service, il a été remplacé en 2016 par le DART.

Ce robot est fait pour l'intérieur et n'a donc pas de GPS. Sa boussole sera considérée comme suffisamment précise donc theta pourra être considéré comme connu, au même titre que les entrées u1 et u2. De plus, il possède des odomètres qui mesurent la vitesse de rotation de chaque roue, mais ce n'est pas utilisé dans le simulateur fourni (http://www.ensta-bretagne.fr/lebars/Share/jog_simu.zip).

1. Réécrire les équations du JOG pour qu'elles deviennent compatibles avec les hypothèses du filtre de Kalman.

2. La position initiale sera considérée comme assez bien connue car en général, l'opérateur sait assez précisément où il a démarré son robot. Rajouter le code nécessaire pour indiquer que l'on connait la position initiale du robot avec une variance de 0.1 (environ 0.5 m d'erreur avec une probabilité de 99%).

3. Dans le scénario actuel, nous n'avons pas de mesures directes de x,y, donc nous n'avons pas de bruit de mesure non plus. Cependant, notre modèle étant forcément imprécis, nous avons du bruit d'état : on peut considérer par exemple qu'il est de variance 0.001 sur x et y (à ajuster en fonction des tailles des ellipses qui seront dessinées par la suite). En fonction de toutes ces informations, utiliser la fonction kalman pour estimer à chaque instant la position du robot avec sa matrice de covariance associée.

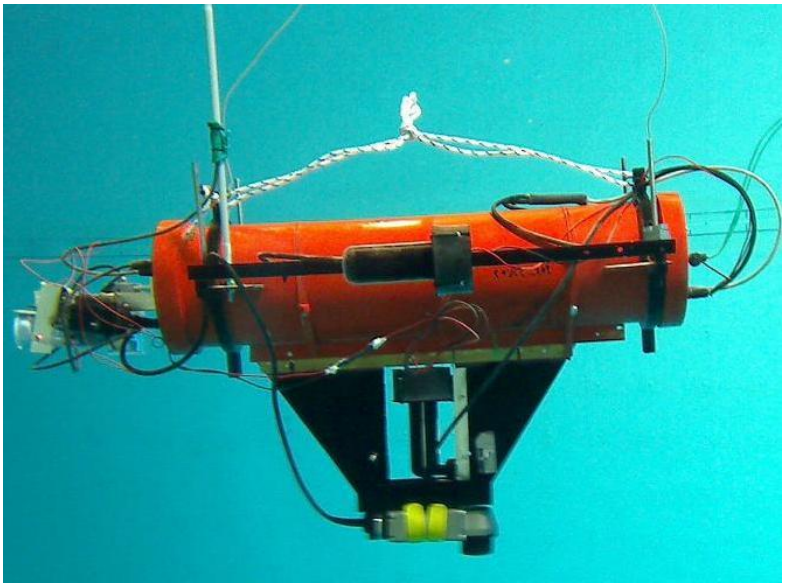
4. Dessiner les ellipsoïdes de confiance contenant la position du robot avec une probabilité de 90%.

5. En pratique, la boussole ne marche pas toujours très bien en intérieur. Rajouter un filtre de Kalman supplémentaire pour filtrer ses mesures grâce aux infos des entrées.

6. Rajouter le suivi autonome de waypoints.

## Part 2 : Filtre de Kalman et régulation d'un robot sous-marin

SAUC'ISSE est un robot sous-marin autonome fabriqué à l'ENSTA Bretagne par des étudiants, doctorants et personnels de l'école, pour participer à des compétitions de robotique sous-marine comme les concours SAUC-E et euRathlon.



Ce robot a un GPS qui lui permet de mesurer sa position lorsqu'il est en surface (typiquement la position initiale et la position finale, parfois aussi lorsqu'on souhaite qu'il fasse surface pour indiquer qu'il a atteint un waypoint, dès qu'il est sous l'eau le GPS ne fonctionne plus), une boussole pour mesurer son cap theta et un capteur de pression pour mesurer sa profondeur z. Ses actionneurs sont 2 propulseurs horizontaux pour se déplacer dans le plan et un propulseur vertical pour monter ou descendre, le robot étant dimensionné pour être de flottabilité presque neutre.

Dans les simulations, les équations d'état du robot sous-marin ont été décrites de cette façon :

$$\begin{cases} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= u_2 - u_1 \\ \dot{v} &= u_1 + u_2 - v \end{cases} \tag{2.5}$$

où $(x, y)$ sont les coordonnées du robot, $\theta$, son orientation et $v$ sa vitesse. Les entrées $u_1$ et $u_2$ sont les forces de propulsion fournies par les propulseurs droite et gauche. Ce modèle normalisé correspond à un robot sous-marin à une profondeur constante (la régulation en profondeur du robot est indépendante et est résolue par une commande dite bang-bang : si le sous-marin est trop profond, on le laisse remonter naturellement ou on actionne son propulseur vertical pour qu'il remonte s'il est loin de la profondeur voulue, autrement on actionne son propulseur vertical pour le faire descendre) sans roulis ni tangage et contrôlé directement via la force de poussée de ses 2 propulseurs horizontaux, avec prise en compte des forces de frottement fluides dues à l'eau (proportionnelles à la vitesse du sous-marin et le ralentissant). De cette manière, notre sous-marin peut être considéré comme un robot évoluant en 2 dimensions.

1. S'inspirer de http://www.ensta-bretagne.fr/lebars/Share/Ecrazor_game.zip pour simuler le sous-marin en 2D. Des coefficients seront peut-être à rajouter pour être plus réaliste.

2. Adapter les équations du robot pour qu'elles deviennent compatibles avec les hypothèses du filtre de Kalman.

3. La position initiale sera considérée comme assez bien connue car le robot a démarré sa mission en surface et de plus, l'opérateur a pris le temps de le placer à un point de départ connu. Rajouter le code nécessaire pour indiquer que l'on connait l'état initial

du robot avec une variance de 0.1 (environ 0.5 m d'erreur avec une probabilité de 99%).

4. Dans le scénario actuel, nous n'avons pas de mesures directes de x,y quand on est sous l'eau, donc nous n'avons pas de bruit de mesure non plus. Cependant, notre modèle étant forcément imprécis, nous avons du bruit d'état : on peut considérer par exemple qu'il est de variance 0.001 sur x, y, v (à ajuster en fonction des tailles des ellipses qui seront dessinées par la suite). En fonction de toutes ces informations, utiliser la fonction kalman pour estimer à chaque instant la position du robot avec sa matrice de covariance associée.

5. Dessiner les ellipsoïdes de confiance contenant la position du robot avec une probabilité de 90%.

6. On rajoute un DVL (Doppler Velocity Log) au robot. Ce capteur permet de mesurer sa vitesse. Rajouter la prise en compte de cette info dans le filtre de Kalman.

7. Rajouter le suivi autonome de waypoints.