Filtrage linéaire et non-linéaire - Projet - Sujet n°1

Thomas GOUMONT - Léo GRUCHOCIAK

2026-01-26

Suivi d'une cible par azimuts (Bearings-Only Tracking)

Les packages suivants sont requis pour pouvoir faire tourner le code.

```
library(MASS)
library(ggplot2)
library(dplyr)
```

```
##
## Attachement du package : 'dplyr'
```

```
## L'objet suivant est masqué depuis 'package:MASS':
##
##     select
```

```
## Les objets suivants sont masqués depuis 'package:stats':
##
##     filter, lag
```

```
## Les objets suivants sont masqués depuis 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(gridExtra)
```

```
##
## Attachement du package : 'gridExtra'
```

```
## L'objet suivant est masqué depuis 'package:dplyr':
##
##     combine
```

0. Introduction et description du problème

Dans ce projet, nous nous attaquons à un problème classique en analyse de cible en mouvement (Target Motion Analysis), à savoir celui du suivi d'une cible par azimuts ("bearings"). On suppose qu'il existe une cible unique suivant une certaine trajectoire, et que deux capteurs bruités nous permettent d'avoir des informations sur cette trajectoire. On peut penser à un mobile en mouvement (une voiture, un drone,...) dont on cherche à suivre le plus précisément possible le trajet, afin d'en analyser le comportement ou de répondre à des problématiques applicatives. Notre objectif s'inscrit dans le contexte de l'inférence dans les modèles à espace d'états: on cherche à reconstruire la trajectoire de notre cible en utilisant uniquement les observations (bruitées) obtenues grâce aux capteurs. Pour cela, nous allons recourir à des techniques de filtrage adaptées, que nous avons étudiées en cours.

Les capteurs à notre disposition sont des capteurs angulaires, i.e. qu'ils fournissent une mesure de la position de la cible à certains instants selon la forme d'un angle. On suppose que la dynamique de notre cible (i.e. la

façon dont la cible évolue dans le temps et dans l'espace suivant sa trajectoire) est observée sur la période temporelle $\{1, \ldots, 500\}$, et on dénote par k le pas de temps auquel la cible est observée. Nous disposons donc de 500 points d'observation (bruités) pour notre cible. L'état de notre cible à l'instant k est donné par le vecteur $\mathbf{x}$ de dimension 4, qui décrit la cinématique de notre cible, i.e. à la fois sa position (x_k, y_k) et sa vélocité $(\dot{x}_k, \dot{y}_k)$:

$$\mathbf{x}_k = (x_k, y_k, \dot{x}_k, \dot{y}_k)^T$$

Pour modéliser la dynamique de notre vecteur d'état, on suppose un modèle de vélocité presque constante (NCV), équivalent à un modèle de Wiener intégré discrétisé, donné par: $\forall k \in \{1, \ldots, 500\}$,

$$\begin{pmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{pmatrix} := \begin{pmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{pmatrix} + \mathbf{q}_k = A\mathbf{x}_{k-1} + \mathbf{q}_k = \begin{pmatrix} x_{k-1} + \Delta\dot{x}_{k-1} \\ y_{k-1} + \Delta\dot{y}_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{pmatrix} + \mathbf{q}_k,$$

avec $(\mathbf{q}_k)_{k=1}^{500}$ un processus Gaussien tel que $\mathbf{q}_k \sim \mathcal{N}(0, Q)$, et:

$$Q = \begin{pmatrix} q_1^c \frac{\Delta^3}{3} & 0 & q_1^c \frac{\Delta^2}{2} & 0 \\ 0 & q_2^c \frac{\Delta^3}{3} & 0 & q_2^c \frac{\Delta^2}{2} \\ q_1^c \frac{\Delta^2}{2} & 0 & q_1^c \Delta & 0 \\ 0 & q_2^c \frac{\Delta^2}{2} & 0 & q_2^c \Delta \end{pmatrix}$$

Par ailleurs, pour ce problème, les coefficients de diffusion ont pour valeur $q_1^c = q_2^c = 0.1$, et la période d'échantillonnage est fixée à $\Delta = 0.01$. Enfin, pour que nos deux capteurs puissent nous fournir de l'information sur la trajectoire de notre cible, nous devons définir la mesure d'angle qui leur permet d'observer cette trajectoire. Cette mesure d'azimut ("bearing measurement") est donnée pour tout capteur $i \in \{1, 2\}$ et par $\forall k \in \{1, \ldots, 500\}$:

$$\theta_k^i = \tan^{-1}(\frac{y_k - s_y^i}{x_k - s_x^i}) + \mathbf{r}_k = \arctan(\frac{y_k - s_y^i}{x_k - s_x^i}) + \mathbf{r}_k,$$

où (s_x^i, s_y^i) dénotent les coordonnées de la position de chaque capteur, et $\mathbf{r}_k \sim \mathcal{N}(0, \sigma^2)$, $\quad \sigma = 0.05$ est un bruit de mesure Gaussien. À partir de ces éléments, il nous est possible de reformuler le problème sous la forme classique d'un problème d'inférence dans un espace d'état discret. On obtient les équations d'état (1), et de mesure (2) suivantes : $\forall k \in \{1, \ldots, 500\}$, $\forall\ i \in \{1, 2\}$,

$$\mathbf{x}_k = f_\theta(u_k, \mathbf{x}_{k-1}, \eta_k) = A_\theta \mathbf{x}_{k-1} + G_\theta \eta_k = A\mathbf{x}_{k-1} + \mathbf{q}_k \qquad (1)\theta_k^i = h_\theta(d_k, \mathbf{x}_k, \epsilon_k) = h(\mathbf{x}_k) + \mathbf{r}_k = \arctan(\frac{y_k - s_y^i}{x_k - s_x^i}) + \mathbf{r}_k \qquad (2)$$

Notre objectif est donc d'estimer la trajectoire de notre cible, i.e. d'estimer la séquence $\mathbf{x}_{1:n} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, à partir des observations bruitées à notre disposition $\theta_{1:n}^i = (\theta_1^i, \ldots, \theta_n^i)$, pour les capteurs $i \in \{1, 2\}$. Pour ce faire, nous allons recourir aux méthodes de filtrage.

En se plaçant dans le contexte des modèles à espaces d'états, il est possible de vérifier si certaines des hypothèses de base sont vérifiées par la configuration de notre problème. Tout d'abord, on observe que la fonction f_θ de l'équation d'état, qui décrit la dynamique du système, est linéaire en ses arguments : l'évolution dynamique de la cible (vitesse et position) est linéaire et gaussienne, puisqu'il s'agit d'un modèle de vélocité presque constante. Cependant, l'équation de mesure, qui régit elle la dynamique des observations, n'est pas linéaire. En effet, les mesures de nos capteurs étant des angles, la relation entre l'état de coordonnées (x, y) de notre cible et la mesure d'angle est décrite par une fonction arctangente, qui est non-linéaire. Le bruit associé à l'erreur de mesure est cependant toujours supposé gaussien. Cette reformulation de notre problème dans le contexte des modèles à espaces d'états nous permet d'ores et déjà de voir que l'utilisation d'un filtre de Kalman (1967) classique pour estimer la trajectoire de notre cible est impossible, puisque l'hypothèse principale de linéarité de la fonction h_θ dans l'équation de mesure est invalide. Nous allons donc prendre en compte cette non-linéarité inhérente à notre système pour utiliser une technique de filtrage adaptée.

1. Estimation de la trajectoire de la cible : cas d'un seul capteur

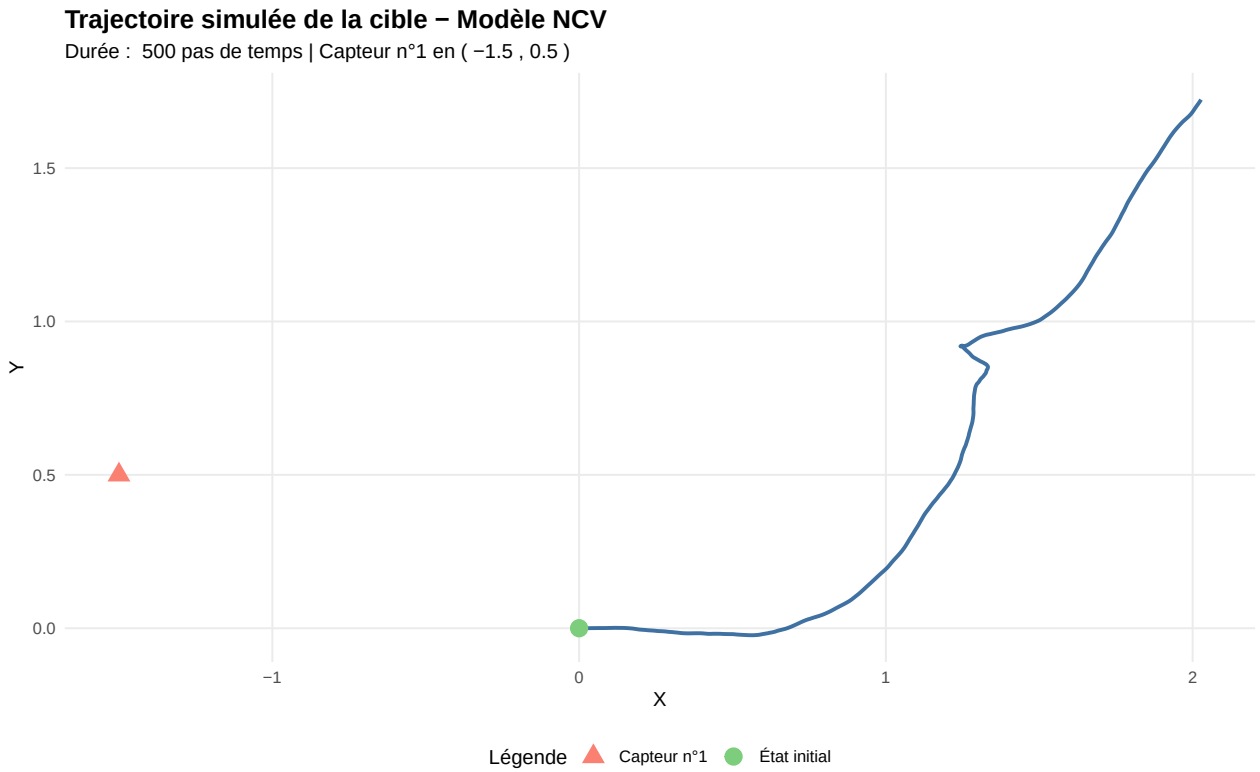
Supposons pour débuter que seul le capteur $i = 1$ est fonctionnel, et qu'il a pour position les coordonnées $(s_x^1, s_y^1) = (-1.5, 0.5)$. On commence par simuler les données $\theta_{1:n}^i$ avec les paramètres définis plus haut, et avec pour état initial $\mathbf{x}_1 = (0, 0, 1, 0)$

(a) Simulation des données

Conformément aux informations données dans l'énoncé, on génère $n = 500$ observations $(\theta_k^1)_{k=1}^{500}$. Une graine aléatoire est fixée pour permettre la reproductibilité de nos opérations.

```
# On fixe la graine aléatoire pour permettre la reproductibilité de nos expériences.
set.seed(2026)
# On reprend les données de l'énoncé :
n <- 500
delta <- 0.01
q1c <- 0.1
q2c <- 0.1
s_x <- -1.5
s_y <- 0.5
sigma <- 0.05

A <- matrix(c(1,0,delta,0,
              0,1,0,delta,
              0,0,1,0,
              0,0,0,1), 4, 4, byrow = TRUE) # On génère la matrice V qui décrit
# la dynamique de la cible, selon le modèle à vitesse presque constante

# On génère la matrice de covariance du bruit Gaussien de l'équation d'état :

Q<- matrix(c(q1c*delta^3/3, 0, q1c*delta^2/2, 0,
              0, q2c*delta^3/3, 0, q2c*delta^2/2,
              q1c*delta^2/2, 0, q1c*delta, 0,
              0, q2c*delta^2/2, 0, q2c*delta), 4, 4, byrow = TRUE)

# Puis, on génère la matrice de taille 4x500 qui va stocker les états de notre
# cible à chaque pas de temps, en incorporant l'état initial :

x_cible <- matrix(0, 4, n)
x_cible[,1] <- c(0, 0, 1, 0)

# Après l'état initial, on génère les 499 états de la cible conformément à l'équation d'état.
# à noter que cette trajectoire réelle est indisponible dans la réalité, et constitue l'objet de
# notre estimation.

for (k in 2:n) {
  x_cible[,k] <- A%*%x_cible[,k-1] + mvrnorm(1, rep(0,4), Q)
}

# Enfin, on peut désormais générer les observations (azimuts)
# selon la formule fournie par l'équation de mesure:

theta <- numeric(n)
for (k in 1:n) {
```

```r
  theta[k] <- atan2(x_cible[2,k] - s_y, x_cible[1,k] - s_x) + rnorm(1, 0, sigma)
}
# La fonction 'atan2' est utilisée ici comme variante de la fonction arctangente,
# afin de gérer les signes dans les 4 quadrants.
# On peut aussi visualiser la trajectoire simulée de notre cible,
# pour avoir une idée de sa régularité :

df_traj <- data.frame(
  Temps = 1:n,
  Pos_X = x_cible[1, ],
  Pos_Y = x_cible[2, ]
)

df_points <- data.frame(
  x = c(x_cible[1,1], s_x),
  y = c(x_cible[2,1], s_y),
  Type = c("État initial", "Capteur n°1")
)

ggplot() +
  geom_path(data = df_traj, aes(x = Pos_X, y = Pos_Y),
            color = "dodgerblue4", linewidth = 1, alpha = 0.8) +
    geom_point(data = df_points, aes(x = x, y = y, color = Type, shape = Type),
               size = 4) +
    scale_color_manual(values = c("Capteur n°1" = "salmon", "État initial" = "palegreen3")) +
  scale_shape_manual(values = c("Capteur n°1" = 17, "État initial" = 19)) +
  labs(title = "Trajectoire simulée de la cible - Modèle NCV",
       subtitle = paste("Durée : ", n, "pas de temps | Capteur n°1 en (", s_x, ",", s_y, ")"),
       x = "X",
       y = "Y",
       color = "Légende", shape = "Légende") +
    theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    legend.position = "bottom",
    panel.grid.minor = element_blank()
  ) +

  coord_fixed(ratio = 1)
```



La figure ci-dessus illustre une réalisation de la trajectoire simulée selon le modèle de vitesse quasi-constante (NCV). On visualise bien la position du capteur (triangle rouge) par rapport au départ de la cible. La cible s'éloigne du capteur au cours du temps. On peut observer une trajectoire globalement linéaire, traduisant la dynamique du modèle, mais perturbée toutefois par des irrégularités, notamment vers le milieu de la simulation. Ces déviations ne sont pas des manœuvres déterministes, mais sont induites par le bruit de processus injecté via la matrice de variance-covariance du bruit Q. Le bruit agit en fait comme une marche aléatoire sur la vitesse de la cible étudiée. Cette trajectoire irrégulière est nécessaire pour tester la robustesse des algorithmes de filtrage, afin d'évaluer leur capacité à reconstruire efficacement la trajectoire de la cible à partir des mesures bruitées observées. Le filtre utilisé devra être capable de distinguer ces vrais changements de trajectoire de notre cible (qui sont dus à Q) du simple bruit de mesure des capteurs.

(b) Choix du filtre

Comme expliqué en introduction, même si l'équation décrivant la dynamique de la cible est linéaire, l'utilisation directe du filtre de Kalman standard n'est pas appropriée pour estimer la trajectoire de notre cible à partir de nos observations angulaires du fait de la non-linéarité de l'équation de mesure. Puisque l'on ne souhaite pas recourir à un algorithme du type filtre particulaire pour résoudre ce problème (probablement en raison des coûts de calcul plus importants), un algorithme de filtrage pertinent à utiliser dans notre cas s'avère être le filtre de Kalman étendu (EKF), qui convient lorsque les hypothèses de bruits gaussiens sont valides, mais que l'équation de mesure est faiblement linéaire, ce qui est le cas ici. L'EKF consiste en une linéarisation (via une approximation de Taylor au premier ordre) du filtre Kalman classique autour de l'estimation prédite et du bruit $\mathbf{r}_k = 0$, ce qui permet de traiter le cas de mesure non-linéaire. Autrement dit, nous allons devoir linéariser la fonction $h(\mathbf{x}_k)$ en calculant la jacobienne $C_\theta = \frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}|_{\hat{\mathbf{x}}_k^-}$: $\forall k \in \{1, \dots, 500\}$,

$$\theta_k^1 = h(\mathbf{x}_k) + \frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}|_{\hat{\mathbf{x}}_k^-}(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) + \frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{r}_k}|_{\hat{\mathbf{x}}_k^-}\mathbf{r}_k = C_\theta \mathbf{x}_k + M_\theta \mathbf{r}_k$$

Comme dans le cours, cette linéarisation permet d'écrire une équation de mesure linéaire dépendant du point de linéarisation, sur laquelle les équations du filtre de Kalman peuvent être appliquées. L'EKF se révèle très

utile en pratique, en particulier grâce à son coût computationnel très faible. Ici, le recours à l'EKF apparaît pertinent, puisque la non-linéarité de la fonction de mesure est modérée et que le bruit de mesure est faible, ce qui limite l'erreur induite par la linéarisation.

(c) Implémentation de l'EKF

Nous allons maintenant implémenter le filtre de Kalman étendu. On commence par dériver l'équation d'observation par rapport aux composantes (x_k, y_k) à k fixé, ce qui revient à calculer la Jacobienne C_θ :

$$C_\theta = \frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}|_{\hat{\mathbf{x}}_k^-} = \begin{pmatrix} \frac{\partial \theta_k^1}{\partial x_k} & \frac{\partial \theta_k^1}{\partial y_k} & \frac{\partial \theta_k^1}{\partial \dot{x}_k} & \frac{\partial \theta_k^1}{\partial \dot{y}_k} \end{pmatrix}$$

En utilisant la dérivée de la fonction $u \mapsto \arctan(u)$, donnée par, pour toute fonction $u : \mathbb{R} \to \mathbb{R}$, $(\arctan(u))' = \frac{u'}{1+u^2}$, il vient que: $\forall k \in \{1, \dots, 500\}$,

$$\frac{\partial \theta_k^1}{\partial x_k} = \frac{\frac{-(y_k - s_y^1)}{(x_k - s_x^1)^2}}{1 + (\frac{y_k - s_y^1}{x_k - s_x^1})^2} = \frac{-(y_k - s_y^1)}{(x_k - s_x^1)^2 + (y_k - s_y^1)^2} \quad \frac{\partial \theta_k^1}{\partial y_k} = \frac{\frac{1}{x_k - s_x^1}}{1 + (\frac{y_k - s_y^1}{x_k - s_x^1})^2} = \frac{x_k - s_x^1}{(x_k - s_x^1)^2 + (y_k - s_y^1)^2}$$

Comme les mesures d'angle θ_k^1 ne dépendent pas directement des vitesses $\dot{x}_k$ et $\dot{y}_k$, les dérivées partielles par rapport à ces variables sont nulles, conduisant à la matrice Jacobienne suivante :

$$C_\theta = \begin{pmatrix} \frac{-(y_k - s_y^1)}{(x_k - s_x^1)^2 + (y_k - s_y^1)^2} & \frac{x_k - s_x^1}{(x_k - s_x^1)^2 + (y_k - s_y^1)^2} & 0 & 0 \end{pmatrix}$$

Grâce à cette matrice, il nous est désormais possible de linéariser l'équation de mesure, et donc d'implémenter intégralement l'EKF pour notre problème. Il est important de souligner que cette matrice Jacobienne dépend explicitement de l'état prédit $\hat{\mathbf{x}}_k^-$, et doit donc être recalculée à chaque itération du filtre.

1.c.1 Initialisation

Pour implémenter l'EKF, une première étape est le choix du point d'initialisation de l'algorithme. Il s'agit d'une étape cruciale pour permettre le bon fonctionnement de l'algorithme : en effet, une mauvaise initialisation pourrait faire diverger l'algorithme. Même si un point initial $\mathbf{x}_1$ est donné pour la simulation, ce point n'est pas disponible pour l'EKF. En effet, dans une situation opérationnelle réelle, l'état initial exact de la cible est inconnu des capteurs de mouvements. Aussi, initialiser l'algorithme avec la vraie valeur initiale de la cible serait artificiel, et ne permettrait pas d'évaluer pleinement la capacité de convergence de l'algorithme.

Puisque l'énoncé ne fournit aucune information précise sur le principe d'initialisation de l'algorithme choisi, nous décidons d'initialiser l'EKF avec une estimation bruitée de l'état initial $\mathbf{x}_1$. Autrement dit, le vecteur d'état initial du filtre $\hat{\mathbf{x}}_1$ est initialisé avec une perturbation aléatoire, avec un écart type égal à 1, autour de la vraie position $\mathbf{x}_1$. Cette configuration permet de simuler une connaissance a priori imparfaite de la zone de la cible, et va nous permettre de montrer que l'algorithme cherche vraiment à trouver la cible. Nous faisons toutefois attention à ne pas trop perturber notre point initial, pour que l'algorithme ne se mette à diverger.

Par ailleurs, le filtre nécessite également une matrice de covariance initiale, $\hat{P}_1$. Ici, nous choisissons $\hat{P}_1 = I_4$, avec I_4 la matrice identité de dimension 4, qui nous semble être une incertitude raisonnable pour débuter. Cette variance relativement large reflète notre incertitude initiale sur l'état. Elle est essentielle pour que le filtre ne soit pas trop confiant dès le début sur la position de la cible, ce qui lui permet d'accorder plus de poids aux premières mesures pour corriger sa trajectoire (i.e. que l'on s'attend à un gain de Kalman plus élevé au démarrage). Notons que la valeur de la matrice $\hat{P}_1$ ne rajoute pas de bruit au système, mais informe simplement le filtre qu'il ne doit pas faire confiance à l'état initial et doit corriger rapidement son estimation de la trajectoire grâce aux mesures.

L'étape d'initialisation de l'algorithme est donc:

```r
P <- diag(c(1, 1, 1, 1))
R <- matrix(sigma^2, 1, 1)  # R est la matrice de covariance du bruit de mesure.
# Elle est de taille 1x1, puisqu'un seul capteur est observé.
# D'après l'énoncé, on a sigma = 0.05.

x_hat <- matrix(0, 4, n) # Notre matrice des états estimés par le filtre

# On initialise l'algorithme avec un état initial perturbé: x_cible[,1] est
# la vraie valeur (0,0,1,0), et on lui ajoute un bruit aléatoire
# sur ses positions (x,y)
x_hat[,1] <- x_cible[,1] + c(rnorm(1, 0, 0.5), rnorm(1, 0, 0.5), 0, 0)
```

Remarque: Les vitesses auraient également pu être bruitées.

1.c.2 Étape de prédiction

Le processus de prédiction du prochain état $\hat{\mathbf{x}}_k^-$ est identique à celui effectué dans le filtre de Kalman standard:

- L'état prédit est: $\hat{\mathbf{x}}_k^- = A\hat{\mathbf{x}}_{k-1}$
- La covariance prédite est : $P_k^- = AP_{k-1}^- A^T + Q$

1.c.3 Étape de mise à jour

Pour la phase de mise à jour de l'algorithme EKF, il est nécessaire de calculer la matrice de gain ou matrice de Kalman K_k, qui a pour expression:

$$K_k = P_k^- C_\theta^T (C_\theta P_k^- C_\theta^T + R)^{-1}$$

On calcule ensuite l'innovation, i.e. le résidu de l'estimation :

$$\tilde{\theta}_k^1 = \theta_k^1 - h(\hat{\mathbf{x}}_k^-),$$

avec $h(\mathbf{x}_k)$ la mesure réelle de l'angle et $h(\hat{\mathbf{x}}_k^-)$ est l'angle calculé à partir de la prédiction. Pour ce terme de résidu, nous devons gérer la discontinuité de l'angle (modulo π ou 2π). En ce sens, l'innovation devra être normalisée entre $[-\pi, \pi]$. Sans cette étape de normalisation, le gain de Kalman réagirait fortement à cette erreur, et le filtre divergerait instantanément.

Enfin, une étape de correction de l'état prédit et de la covariance prédite est effectuée de la façon suivante:

- Correction de l'état prédit: $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + K_k\tilde{\theta}_k^1$
- Correction de la covariance prédite: $P_k = (Id - K_k C_\theta)P_k^-$

Finalement, l'algorithme EKF est implémenté via le code suivant.

```r
# Pour pouvoir afficher les erreurs faites par notre filtre (cf. infra), on
# initialise une matrice de stockage des variances ainsi qu'un vecteur de stockage des résidus

P_diag <- matrix(0, 4, n)
P_diag[,1] <- diag(P) # Stocker la variance initiale
innovations <- numeric(n) # Vecteur pour stocker les résidus
# On crée une boucle pour l'algorithme EKF, pour k allant de 2 à 500 :
for (k in 2:n) {
# (i) Étape de prédiction
x_pred <- A %*% x_hat[,k-1]
P_pred <- A %*% P %*% t(A) + Q
```

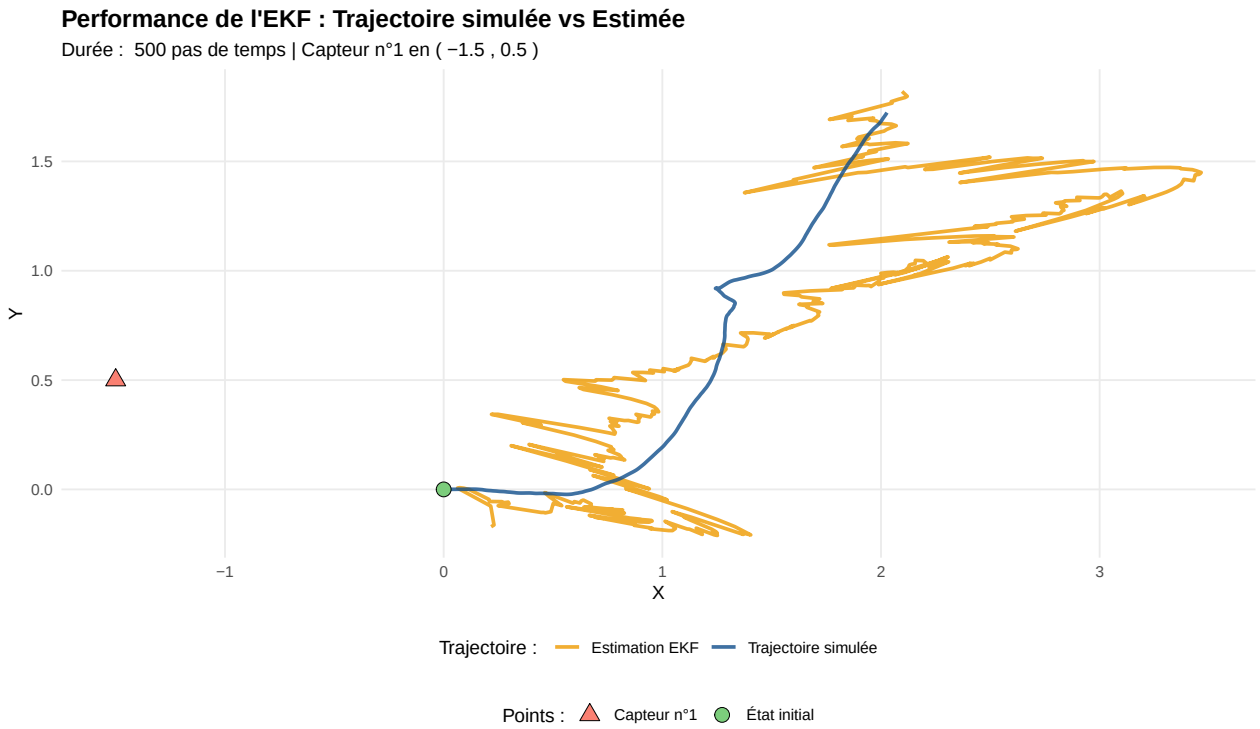
```
# (ii) Étape de mise à jour:
# On définit la fonction de mesure des angles

a <- x_pred[1] - s_x
b <- x_pred[2] - s_y
h <- atan2(b,a)

# On calcule la Jacobienne C linéarisée autour de la prédiction

denom <- a^2 + b^2
C <- matrix(c(-b/denom, a/denom, 0, 0), 1, 4)

# On calcule l'innovation. Pour gérer la discontinuité de l'angle,
# on utilise atan2(sin(diff), cos(diff)), qui gère automatiquement le modulo Pi

theta_tilde <- atan2(sin(theta[k] - h), cos(theta[k] - h))
innovations[k] <- theta_tilde
# On calcule le gain de Kalman

S <- C%*%P_pred%*% t(C) + R
K <- P_pred%*% t(C) %*%solve(S)

# On effectue la correction de l'état prédit et de la covariance
x_hat[,k] <- x_pred + K * theta_tilde
P <- (diag(4) - K%*% C)%*% P_pred
P_diag[,k] <- diag(P) # On sauvegarde les variances
}
```

(d) Interprétations des résultats et commentaires

On affiche maintenant les résultats de notre estimation de la trajectoire de la cible, obtenue avec le filtre EKF implémenté ci-dessus. Une interprétation des résultats est donnée ci-après.

```
set.seed(2026) # Graine aléatoire fixée pour la reproductibilité

df_vrai <- data.frame(
  Temps = 1:n,
  X = x_cible[1,],
  Y = x_cible[2,],
  Type = "Trajectoire simulée"
)

df_estime <- data.frame(
  Temps = 1:n,
  X = x_hat[1,],
  Y = x_hat[2,],
  Type = "Estimation EKF"
)

df_global <- rbind(df_vrai, df_estime) # Dataset avec la trajectoire simulée et celle estimée

df_points <- data.frame(
  x = c(x_cible[1,1], s_x),
```

```
  y = c(x_cible[2,1], s_y),
  Label = c("État initial", "Capteur n°1")
) # Dataset avec le point du capteur et celui de l'état initial

ggplot() +
  geom_path(data = df_global, aes(x = X, y = Y, color = Type),
            linewidth = 1, alpha = 0.8) +

  geom_point(data = df_points, aes(x = x, y = y, fill = Label, shape = Label),
             size = 4, stroke = 0) +

  scale_color_manual(values = c("Trajectoire simulée" = "dodgerblue4",
                                "Estimation EKF" = "orange2")) +

  scale_fill_manual(name = "Points :", values = c("Capteur n°1" = "salmon",
                                                  "État initial" = "palegreen3")) +
  scale_shape_manual(name = "Points :", values = c("Capteur n°1" = 24,
                                                   "État initial" = 21)) +

  labs(title = "Performance de l'EKF : Trajectoire simulée vs Estimée",
       subtitle = paste("Durée : ", n, "pas de temps | Capteur n°1 en (", s_x, ",", s_y, ")"),
       x = "X",
       y = "Y",
       color = "Trajectoire : ") +

  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    legend.position = "bottom",
    legend.box = "vertical",
    panel.grid.minor = element_blank()
  ) +

  coord_fixed(ratio = 1)
```

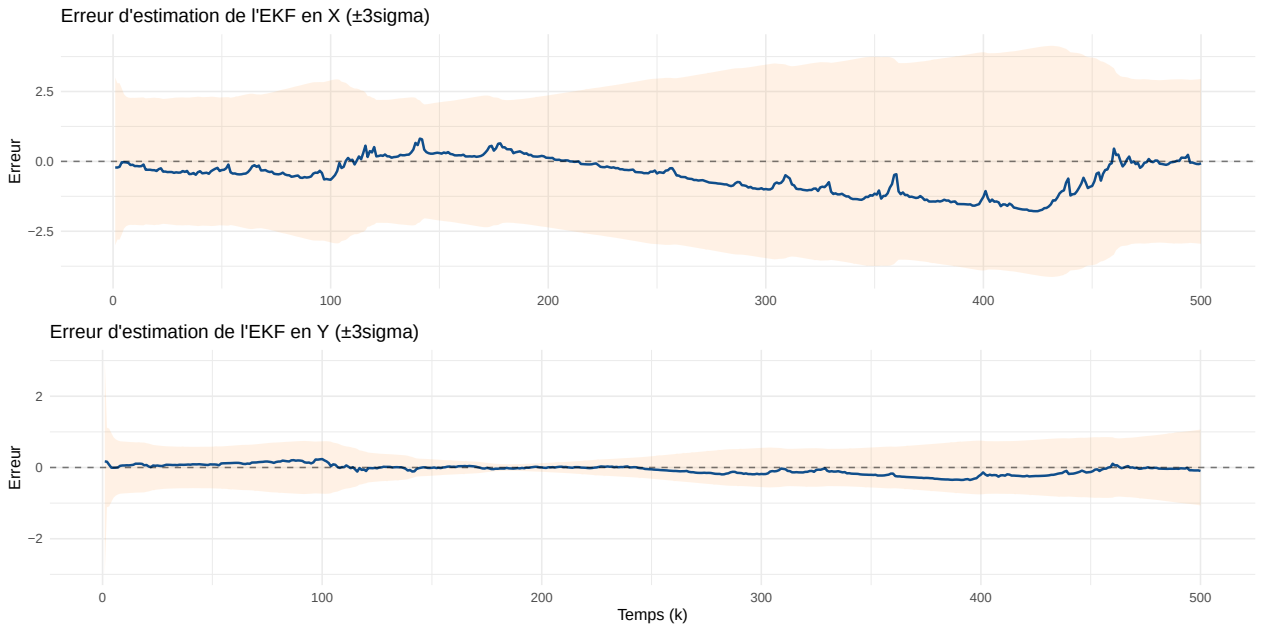


Sur le graphique ci-dessus, on peut voir que la trajectoire estimée à partir du capteur bruité (la courbe orange) est bien plus irrégulière que la trajectoire simulée (la courbe bleue) de la cible (notre vérité terrain). Plus spécifiquement, l'allure chaotique de l'estimation révèle que les oscillations ne sont pas aléatoires, mais elles s'étirent le long de l'axe reliant le capteur à la cible. Ce phénomène illustre l'inobservabilité inhérente au suivi par azimuts seuls avec un unique capteur fixe. Notre EKF corrige efficacement l'erreur angulaire (i.e. que la direction générale qui est estimée est correcte), mais il peine à estimer avec précision la trajectoire exacte, à cause de l'augmentation de la distance capteur-cible. Or, en l'absence de manœuvre du capteur, cette ambiguïté sur la distance ne peut être levée mathématiquement, ce qui empêche l'estimation de converger vers la trajectoire lisse réelle et explique également l'élargissement progressif de l'incertitude observé sur les graphiques d'erreur ci-dessous.

En effet, pour analyser la qualité de notre filtre EKF, il nous est possible d'afficher les erreurs qu'il a effectuées lors des étapes de prédictions. On se sert de la matrice P_diag, définie dans la boucle du filtre, qui stocke les variances au fur et à mesure des itérations du filtre EKF.

```
# On calcule les erreurs et les écarts-types sigma pour construire des bornes de
# confiance à ±3sigma
err_x <- x_cible[1, ] - x_hat[1, ]
err_y <- x_cible[2, ] - x_hat[2, ]
sigma_x <- sqrt(P_diag[1, ])
sigma_y <- sqrt(P_diag[2, ])

# On définit un dataframe avec une borne pour X et une borne pour Y
df_err <- data.frame(
  Temps = 1:n,
  Erreur_X = err_x,
  Erreur_Y = err_y,
  Borne_X = 3 * sigma_x,
  Borne_Y = 3 * sigma_y
)

# On plot le graphique de l'erreur d'estimation en X
```

```r
p1 <- ggplot(df_err, aes(x = Temps)) +
  geom_ribbon(aes(ymin = -Borne_X, ymax = Borne_X), fill = "tan1", alpha = 0.15) +
  geom_line(aes(y = Erreur_X), color = "dodgerblue4", linewidth = 0.8) +
  geom_hline(yintercept = 0, linetype = "dashed", alpha = 0.5) +
  labs(title = "Erreur d'estimation de l'EKF en X (±3sigma)", y = "Erreur", x = NULL) +
  theme_minimal()

# On plot le graphique de l'erreur d'estimation en Y
p2 <- ggplot(df_err, aes(x = Temps)) +
  geom_ribbon(aes(ymin = -Borne_Y, ymax = Borne_Y), fill = "tan1", alpha = 0.15) +
  geom_line(aes(y = Erreur_Y), color = "dodgerblue4", linewidth = 0.8) +
  geom_hline(yintercept = 0, linetype = "dashed", alpha = 0.5) +
  labs(title = "Erreur d'estimation de l'EKF en Y (±3sigma)", y = "Erreur", x = "Temps (k)") +
  theme_minimal()

grid.arrange(p1, p2, ncol = 1)
```



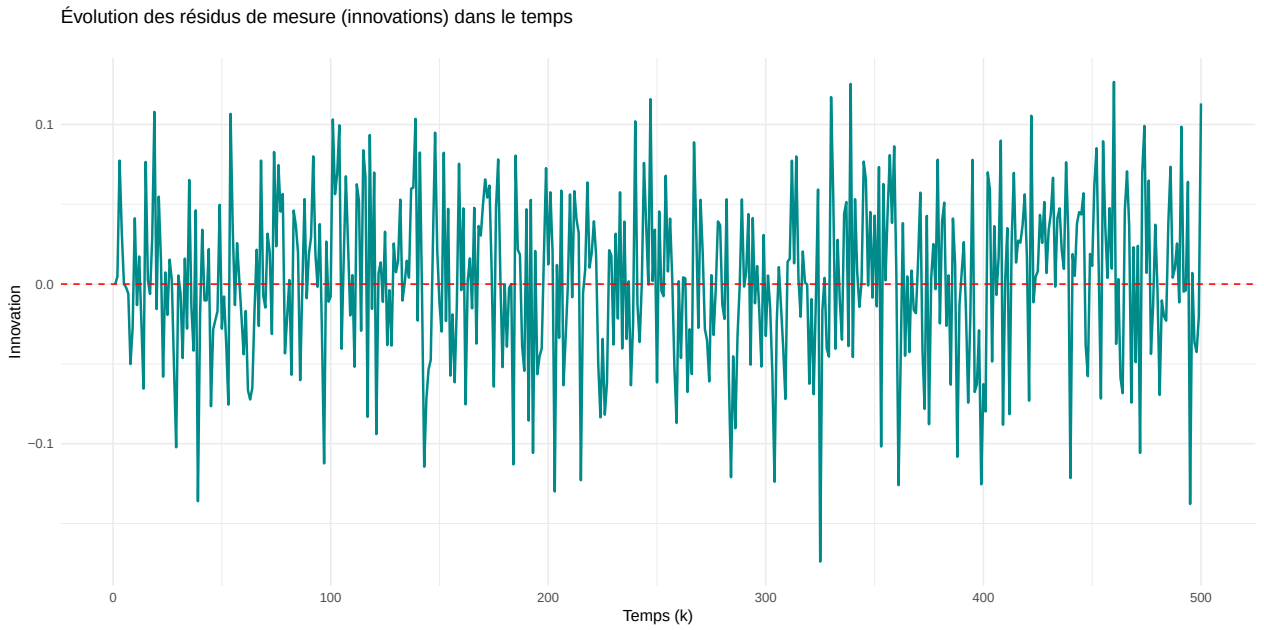
Ces graphiques racontent une histoire très précise sur les limites du filtrage mono-capteur. On observe que la cible ne suit pas une ligne droite parfaite. Ces irrégularités confirment que le bruit de processus (matrice Q) joue bien son rôle, et la cible subit des variations de vitesse aléatoires (selon le modèle NCV). D'abord, on voit que le filtre est fonctionnel et consistant, puisque l'erreur réelle reste confinée dans les intervalles de confiance à $\pm 3\sigma$, ce qui valide notre implémentation de l'EKF et le calcul de la Jacobienne.

Cependant, ces graphiques mettent en évidence la perte d'observabilité en distance. Nous observons en effet un élargissement significatif des bornes de confiance au cours du temps, surtout en X. Du point de vue physique, cela confirme qu'avec un seul capteur d'angle et une cible à vitesse quasi-constante, la distance est faiblement observable. À mesure que la cible s'éloigne, une petite erreur d'angle se traduit par une erreur de position de plus en plus grande. L'EKF perd en fait la notion de distance, d'où l'explosion de la variance P_k.

Pour réduire cette incertitude, il serait nécessaire d'ajouter un second capteur, ou que le capteur effectue une manœuvre pour observer la cible sous différents angles.

Aussi, pour effectuer une analyse détaillée des résultats de l'estimation de notre trajectoire, on peut s'intéresser aux résidus (ou innovations) de l'estimation EKF.

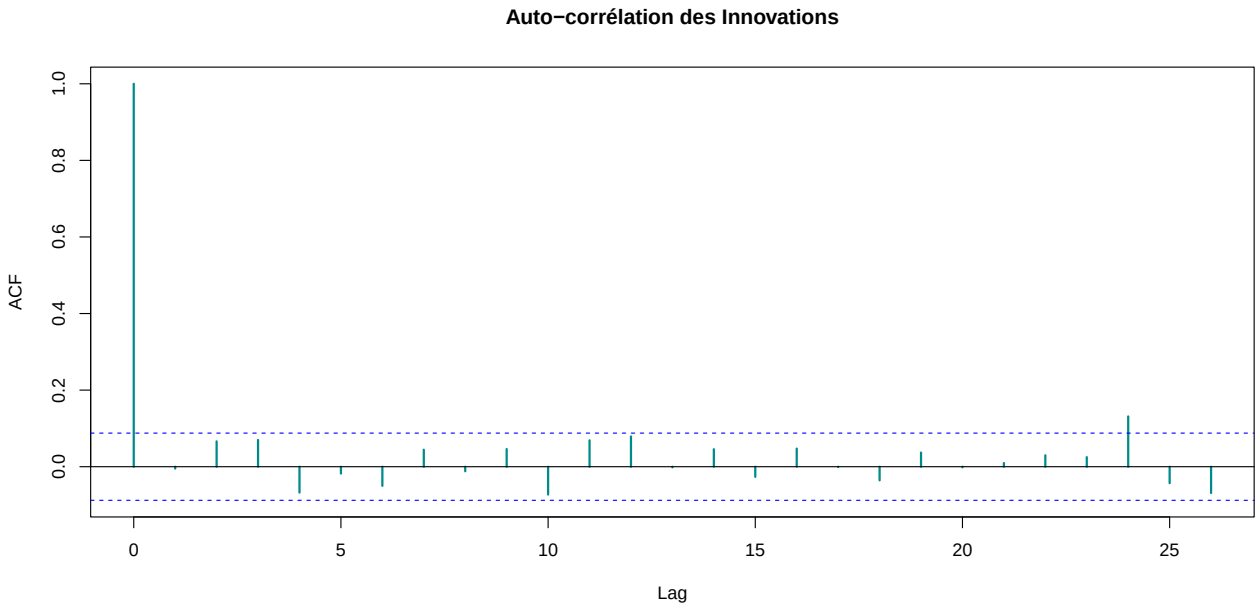
```r
# On crée le dataset avec les innovations stockées dans la boucle de l'EKF
df_innov <- data.frame(
  Temps = 1:n,
  Innovation = innovations
)

# On affiche le graphique des résidus de l'estimation
ggplot(df_innov, aes(x = Temps, y = Innovation)) +
  geom_line(color = "darkcyan", linewidth = 0.8) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +

  # Esthétique
  labs(title = "Évolution des résidus de mesure (innovations) dans le temps",
       subtitle = " ",
       y = "Innovation",
       x = "Temps (k)") +
  theme_minimal()
```



Le graphique ci-dessus confirme la blancheur des résidus : il montre un signal oscillant autour de 0, sans tendance ni motif périodique apparent. Cela indique l'absence de biais systématique dans le modèle de mesure.

On peut également regarder le diagramme d'autocorrélation (ACF) de nos résidus de l'estimation EKF.

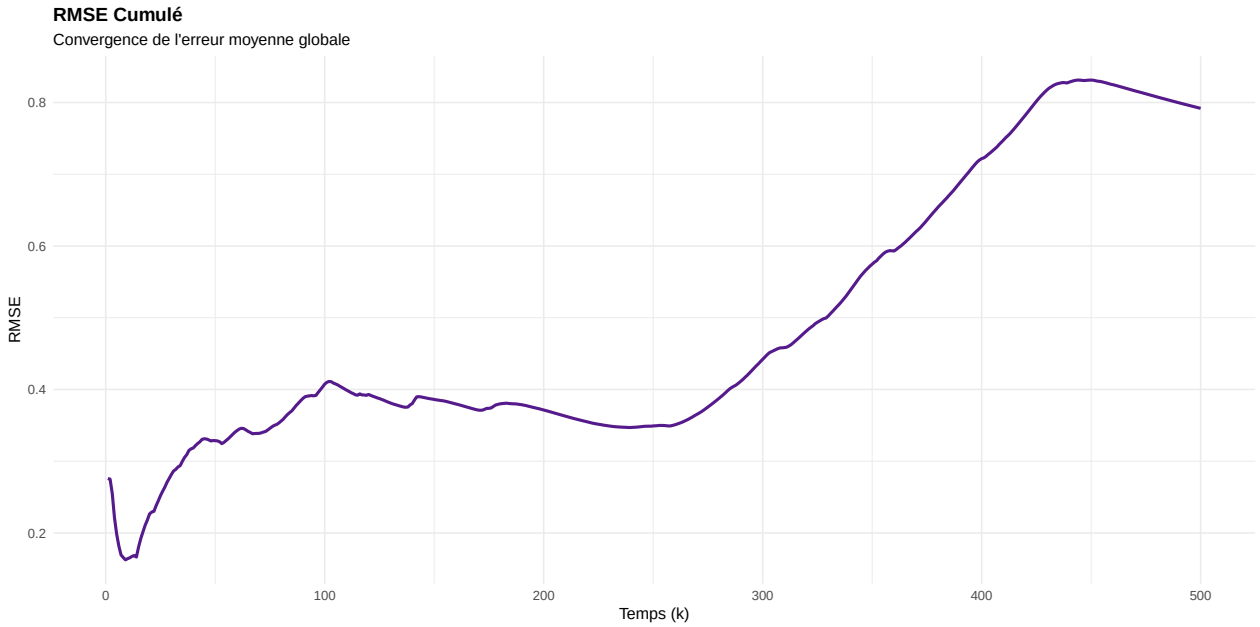
```r
# On exclut le premier élément souvent nul ou artificiel si k commence à 2
acf(innovations[2:n],
    main = "Auto-corrélation des Innovations",
    col = "darkcyan",
    lwd = 2)
```



Le diagramme d'auto-corrélation ci-dessus valide formellement l'observation précédente. La quasi-totalité des lags reste confinés dans l'intervalle de confiance à 95% (pointillés bleus). Cela prouve que le filtre a extrait toute l'information structurelle disponible. Cette analyse confirme la décorrélation temporelle des estimations. Ce comportement des résidus est conforme aux hypothèses du filtre de Kalman étendu, et indique que le modèle dynamique et le modèle de mesure sont globalement cohérents avec les données observées.

Enfin, on peut terminer par regarder la valeur du RMSE de notre estimation au cours du temps.

```
# On reprend err_x et err_y viennent du code plus haut
pos_err_sq <- err_x^2 + err_y^2

# On calcule le RMSE cumulé
rmse_cumule <- sqrt(cumsum(pos_err_sq) / seq_along(pos_err_sq))

df_rmse <- data.frame(
  Temps = 1:n,
  RMSE = rmse_cumule
)

ggplot(df_rmse, aes(x = Temps, y = RMSE)) +
  geom_line(color = "purple4", linewidth = 1) +

  labs(title = "RMSE Cumulé",
       subtitle = "Convergence de l'erreur moyenne globale",
       y = "RMSE",
       x = "Temps (k)") +

  theme_minimal() +

  theme(plot.title = element_text(face = "bold"))
```



Au niveau de la performance globale, le graphique du RMSE confirme l'analyse précédente. Contrairement à un problème classique où le RMSE tend vers 0, ici il a tendance à augmenter (dérive au-delà de 0.4). Cela ne vient pas d'une erreur de code, ou une d'une mauvaise implémentation mathématique du filtre, mais confirme que l'estimation se dégrade géométriquement avec l'éloignement de la cible. Cette dérive du RMSE résulte donc directement de la faible observabilité de la distance dans un problème de suivi par azimut mono-capteur.

Sans implémenter une autre méthode, on peut aborder d'autres approches de filtrage qui auraient pu se révéler plus adaptées à notre tâche. En particulier, les filtres particulaires sont généralement mieux adaptés aux fortes non-linéarités de l'équation de mesure (comme la fonction arctangente) car ils ne reposent pas sur une linéarisation locale comme l'EKF. Ces méthodes présentent cependant l'inconvénient d'être computationnellement plus chères que l'EKF.

De plus, la littérature récente sur les problème de suivi de cible par azimuts nous oriente vers des variantes plus robustes, comme par exemple le filtre particulaire régularisé par boîtes (BRPF), proposé par Merlinge et al. (2016) [2]. Cette approche « ensembliste » permet de mieux gérer les incertitudes initiales élevées, typiques du suivi par azimuts. L'étude compare explicitement cette méthode à l'EKF et démontre que l'EKF peut souffrir de divergences soudaines en cas de faible observabilité, là où le BRPF reste stable. Enfin, pour le cas mono-capteur (correspondant à notre situation), les auteurs expliquent aussi que l'amélioration des résultats passe par la stratégie de guidage du capteur : pour rendre la distance observable avec un seul capteur, celui-ci doit effectuer des manœuvres spécifiques pour maximiser l'information, plutôt que de suivre une trajectoire rectiligne.

Par ailleurs, on a déjà mentionné plus haut que l'observabilité est faible avec un seul capteur, et que ces résultats motivent l'utilisation d'un second capteur pour améliorer la précision de l'estimation. En effet, le problème d'estimation semble mal conditionné, puisqu'un seul angle ne fournit pas la distance précise à la cible.

En conclusion de cette première partie, nous pouvons affirmer que l'EKF implémenté fonctionne, i.e. qu'il est cohérent et statistiquement valide, mais qu'il atteint les limites théoriques du suivi par azimuts mono-capteur. Cette analyse motive l'utilisation d'un second capteur, en complément du premier, afin de stabiliser l'erreur et d'assurer une meilleure estimation de la trajectoire.

2. Partie supplémentaire : le cas de deux capteurs

Cette partie prolonge naturellement l'étude du cas mono-capteur en s'intéressant à l'estimation de la trajectoire de la cible à l'aide de deux capteurs angulaires. L'objectif est d'évaluer dans quelle mesure l'ajout d'un second capteur permet d'améliorer l'observabilité du système et la qualité de l'estimation, tout en conservant le même cadre méthodologique, à savoir l'utilisation d'un filtre de Kalman étendu (EKF).

Contrairement au cas précédent, le modèle d'observation devient vectoriel : à chaque instant, deux mesures d'azimut sont disponibles, issues de deux positions distinctes. Cette redondance d'information doit théoriquement permettre de lever une partie de l'ambiguïté géométrique présente dans le suivi mono-capteur, en particulier sur l'estimation de la distance à la cible. Le vecteur d'observation s'écrit alors :

$$\boldsymbol{\theta}_k = \begin{pmatrix} \theta_k^1 \\ \theta_k^2 \end{pmatrix} = \begin{pmatrix} \arctan\left(\dfrac{y_k - s_y^1}{x_k - s_x^1}\right) \\ \arctan\left(\dfrac{y_k - s_y^2}{x_k - s_x^2}\right) \end{pmatrix} + \mathbf{r}_k,$$

où (s_x^i, s_y^i) désignent les coordonnées du capteur $i \in \{1, 2\}$, et où le bruit de mesure $\mathbf{r}_k$ est toujours supposé gaussien, centré, et de covariance R :

$$\mathbf{r}_k \sim \mathcal{N}(0, R), \qquad R = \sigma^2 I_2.$$

Comme dans le cas mono-capteur, la fonction d'observation est non linéaire en l'état $\mathbf{x}_k$, en raison de la présence de la fonction arctangente. Le recours au filtre de Kalman étendu reste donc nécessaire afin de traiter cette non-linéarité.

La linéarisation autour de l'état prédit $\hat{\mathbf{x}}_k^-$ se fait de la même manière que précédemment, à la différence près que la jacobienne est obtenue en empilant les jacobiennes associées à chacun des deux capteurs.

Elle s'écrit alors :

$$H_k = \begin{pmatrix} \dfrac{-(y_k - s_y^1)}{(x_k - s_x^1)^2 + (y_k - s_y^1)^2} & \dfrac{x_k - s_x^1}{(x_k - s_x^1)^2 + (y_k - s_y^1)^2} & 0 & 0 \\ \dfrac{-(y_k - s_y^2)}{(x_k - s_x^2)^2 + (y_k - s_y^2)^2} & \dfrac{x_k - s_x^2}{(x_k - s_x^2)^2 + (y_k - s_y^2)^2} & 0 & 0 \end{pmatrix}.$$

Cette matrice $H_k \in \mathbb{R}^{2\times 4}$ correspond à la linéarisation du modèle de mesure vectoriel et permet alors l'application directe des équations standards du filtre de Kalman étendu. Dans la suite, la matrice H_k est évaluée en l'état prédit $\hat{\mathbf{x}}_k^-$ à chaque itération du filtre.

```r
n <- 500
delta <- 0.01
q1c <- 0.1
q2c <- 0.1
sigma <- 0.05

A <- matrix(c(1, 0, delta, 0,
              0, 1, 0, delta,
              0, 0, 1, 0,
              0, 0, 0, 1), 4, 4, byrow = TRUE)

Q <- matrix(c(q1c*delta^3/3, 0, q1c*delta^2/2, 0,
              0, q2c*delta^3/3, 0, q2c*delta^2/2,
```

```
              q1c*delta^2/2, 0, q1c*delta, 0,
              0, q2c*delta^2/2, 0, q2c*delta), 4, 4, byrow = TRUE)

# On simule la trajectoire de la cible
set.seed(2026)
x_true <- matrix(0, 4, n)
x_true[,1] <- c(0, 0, 1, 0)

for (k in 2:n) {
  x_true[,k] <- A %*% x_true[,k-1] + mvrnorm(1, rep(0,4), Q)
}

# On implémente cette fois deux capteurs
s1 <- c(-1.5, 0.5) # Capteur 1
s2 <- c(1, 1)      # Capteur 2

# On crée la matrice pour stocker les mesures des capteurs
theta <- matrix(0, 2, n)
for (k in 1:n) {
  # Mesure du Capteur 1
  theta[1,k] <- atan2(x_true[2,k] - s1[2], x_true[1,k] - s1[1]) + rnorm(1, 0, sigma)
  # Mesure du Capteur 2
  theta[2,k] <- atan2(x_true[2,k] - s2[2], x_true[1,k] - s2[1]) + rnorm(1, 0, sigma)
}
```

Puis, on initialise notre filtre de la même façon que précédemment, et on implémente une nouvelle fois le filtre de Kalman :

```
# On reprend les mêmes initialisations que pour la question 1.
x_hat <- matrix(0, 4, n)
x_hat[,1] <- x_true[,1] + c(rnorm(1,0,0.5), rnorm(1,0,0.5), 0, 0)
P <- diag(c(1,1,1,1))
R <- diag(sigma^2, 2)

# On prépare le stockage des variances
P_diag <- matrix(0, 4, n)
P_diag[,1] <- diag(P)

for (k in 2:n) {

  x_pred <- A %*% x_hat[,k-1]
  P_pred <- A %*% P %*% t(A) + Q

  dx1 <- x_pred[1] - s1[1]; dy1 <- x_pred[2] - s1[2]
  dx2 <- x_pred[1] - s2[1]; dy2 <- x_pred[2] - s2[2]
  d1_sq <- dx1^2 + dy1^2; d2_sq <- dx2^2 + dy2^2

  H <- matrix(c(-dy1/d1_sq, dx1/d1_sq, 0, 0,
                -dy2/d2_sq, dx2/d2_sq, 0, 0), 2, 4, byrow=TRUE)

  h_val <- c(atan2(dy1, dx1), atan2(dy2, dx2))
  raw_res <- theta[,k] - h_val
  y_innov <- atan2(sin(raw_res), cos(raw_res))
```

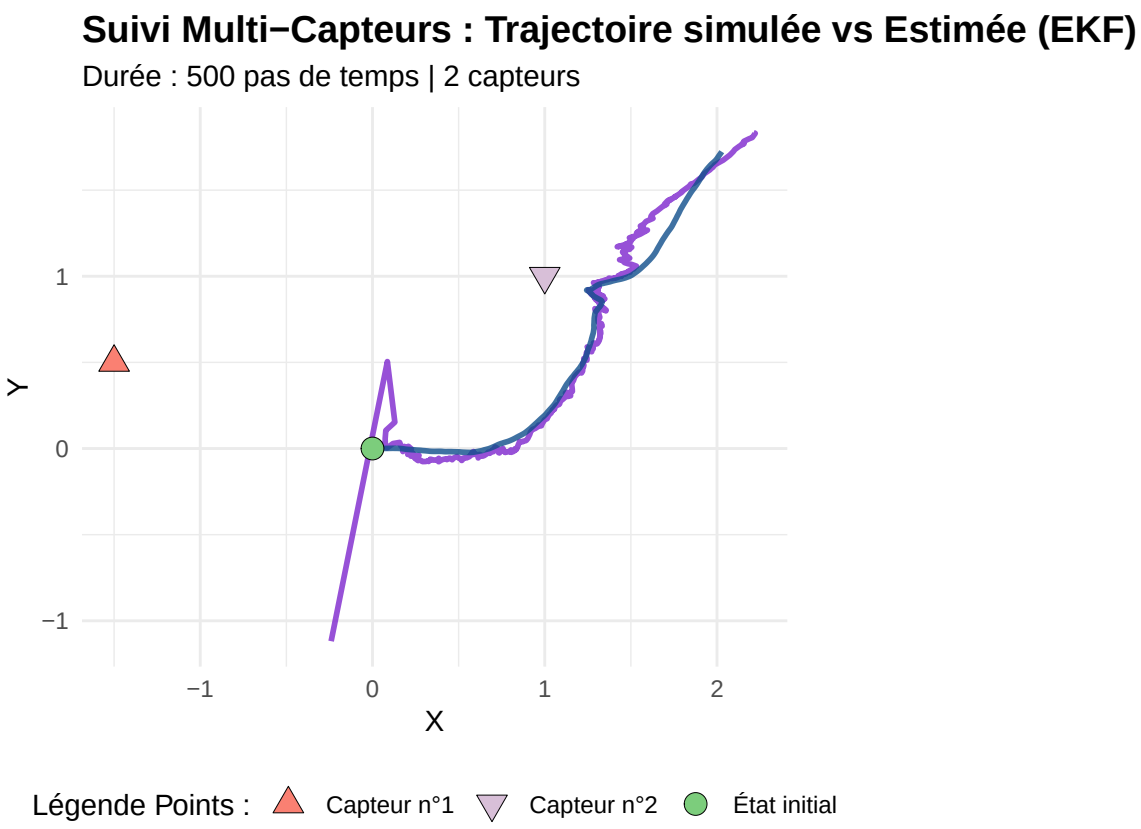
```
  S <- H %*% P_pred %*% t(H) + R
  K <- P_pred %*% t(H) %*% solve(S)

  x_hat[,k] <- x_pred + K %*% y_innov
  P <- (diag(4) - K %*% H) %*% P_pred

  P_diag[,k] <- diag(P)
}
```

Comme dans le cas mono-capteur, la normalisation angulaire via la fonction $\text{atan2}(\sin(\cdot), \cos(\cdot))$ permet d'éviter les discontinuités liées à la périodicité des angles.

On peut désormais afficher les résultats de l'estimation sur un graphique, comme précédemment.

```
# On prépare les DataFrames
df_vrai <- data.frame(Temps = 1:n, X = x_true[1,], Y = x_true[2,], Type = "Trajectoire simulée")
df_estime <- data.frame(Temps = 1:n, X = x_hat[1,], Y = x_hat[2,], Type = "Estimation EKF (2 capteurs)")
df_global <- rbind(df_vrai, df_estime)

df_points <- data.frame(
  x = c(x_true[1,1], s1[1], s2[1]),
  y = c(x_true[2,1], s1[2], s2[2]),
  Label = c("État initial", "Capteur n°1", "Capteur n°2")
)

ggplot() +
  geom_path(data = df_global, aes(x = X, y = Y, color = Type, linetype = Type),
            linewidth = 1, alpha = 0.8) +

  geom_point(data = df_points, aes(x = x, y = y, fill = Label, shape = Label),
             size = 4, stroke = 0) +

  scale_color_manual(values = c("Trajectoire simulée" = "dodgerblue4",
                                "Estimation EKF (2 capteurs)" = "purple3")) +
  scale_linetype_manual(values = c("Trajectoire simulée" = "solid",
                                   "Estimation EKF (2 capteurs)" = "solid")) +

  scale_fill_manual(name = "Légende Points :",
                    values = c("Capteur n°1" = "salmon",
                               "Capteur n°2" = "thistle",
                               "État initial" = "palegreen3")) +

  scale_shape_manual(name = "Légende Points :",
                     values = c("Capteur n°1" = 24,
                                "Capteur n°2" = 25,
                                "État initial" = 21)) +

  labs(
  title = "Suivi Multi-Capteurs : Trajectoire simulée vs Estimée (EKF)",
  subtitle = paste("Durée :", n, "pas de temps | 2 capteurs"),
  x = "X",
  y = "Y",
  color = "Trajectoire :",
  linetype = "Trajectoire :"
) +
```

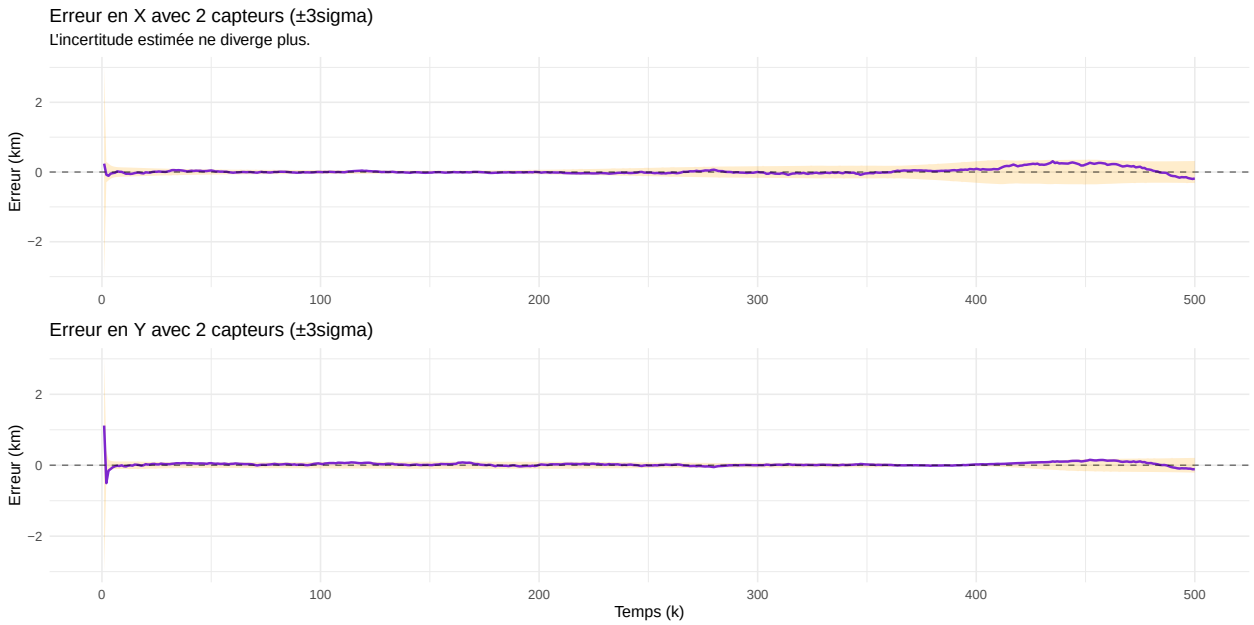
```
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    legend.position = "bottom",
    legend.box = "vertical"
  ) +

  coord_fixed(ratio = 1)
```



Sur le graphique représentant la trajectoire réelle et la trajectoire estimée, on observe que l'estimation fournie par l'EKF converge rapidement vers la trajectoire simulée, et reste ensuite très proche de celle-ci sur l'ensemble de l'horizon temporel. Contrairement au cas mono-capteur, aucune dérive significative n'apparaît sur l'horizon temporel considéré.

```
# On calcule les erreurs et les sigmas
err_x <- x_true[1, ] - x_hat[1, ]
err_y <- x_true[2, ] - x_hat[2, ]
sigma_x <- sqrt(P_diag[1, ])
sigma_y <- sqrt(P_diag[2, ])

df_err_2capteurs <- data.frame(
  Temps = 1:n,
  Erreur_X = err_x,
  Erreur_Y = err_y,
```

```
  Borne_X = 3 * sigma_x,
  Borne_Y = 3 * sigma_y
)

p1 <- ggplot(df_err_2capteurs, aes(x = Temps)) +
  geom_ribbon(aes(ymin = -Borne_X, ymax = Borne_X), fill = "orange", alpha = 0.2) +
  geom_line(aes(y = Erreur_X), color = "purple3", linewidth = 0.8) +
  geom_hline(yintercept = 0, linetype = "dashed", alpha = 0.5) +
  labs(title = "Erreur en X avec 2 capteurs (±3sigma)",
       subtitle = "L'incertitude estimée ne diverge plus.",
       y = "Erreur (km)", x = NULL) +
  theme_minimal()

p2 <- ggplot(df_err_2capteurs, aes(x = Temps)) +
  geom_ribbon(aes(ymin = -Borne_Y, ymax = Borne_Y), fill = "orange", alpha = 0.2) +
  geom_line(aes(y = Erreur_Y), color = "purple3", linewidth = 0.8) +
  geom_hline(yintercept = 0, linetype = "dashed", alpha = 0.5) +
  labs(title = "Erreur en Y avec 2 capteurs (±3sigma)", y = "Erreur (km)", x = "Temps (k)") +
  theme_minimal()

grid.arrange(p1, p2, ncol = 1)
```

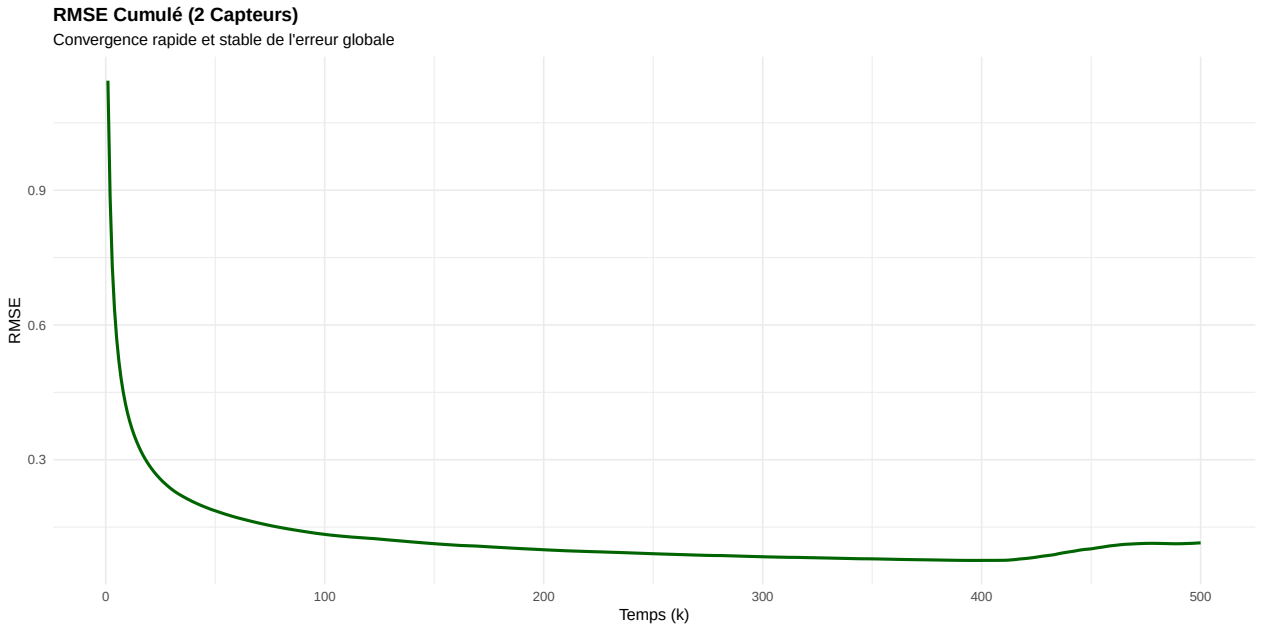


On observe que les erreurs restent majoritairement contenues à l'intérieur des bornes de confiance, ce qui indique que l'incertitude estimée par le filtre est compatible avec l'erreur réelle. Contrairement au cas mono-capteur, les bandes d'incertitude ne divergent plus au cours du temps, traduisant une estimation stable et bien conditionnée. La combinaison de deux mesures angulaires issues de positions distinctes permet ainsi de reconstruire efficacement la géométrie de la scène, rendant observable la distance à la cible et supprimant l'ambiguïté radiale présente dans le cas mono-capteur.

La présence de deux capteurs amène donc directement une réduction et une stabilisation des variances estimées. Les fluctuations résiduelles des erreurs sont cohérentes avec le bruit de processus du modèle NCV et montrent que le filtre parvient à distinguer les variations réelles de trajectoire du simple bruit de mesure.

```
# On calcule une nouvelle fois le RMSE Cumulé
pos_err_sq <- err_x^2 + err_y^2
```

```r
rmse_cumule <- sqrt(cumsum(pos_err_sq) / seq_along(pos_err_sq))

df_rmse_2capteurs <- data.frame(
  Temps = 1:n,
  RMSE = rmse_cumule
)

ggplot(df_rmse_2capteurs, aes(x = Temps, y = RMSE)) +
  geom_line(color = "darkgreen", linewidth = 1) +

  labs(title = "RMSE Cumulé (2 Capteurs)",
       subtitle = "Convergence rapide et stable de l'erreur globale",
       y = "RMSE",
       x = "Temps (k)") +

  theme_minimal() +
  theme(plot.title = element_text(face = "bold"))
```



L'évolution de la RMSE cumulée confirme l'amélioration globale des performances d'estimation dans le cas bi-capteurs. Après une phase transitoire initiale, liée à l'incertitude sur l'état initial, la RMSE converge rapidement vers une valeur faible et se stabilise sans présenter de dérive à long terme. Ce comportement contraste fortement avec le cas mono-capteur, pour lequel la RMSE augmentait progressivement au cours du temps.

La stabilisation de la RMSE traduit un problème d'estimation désormais bien conditionné, rendu possible par la complémentarité des mesures angulaires fournies par les deux capteurs. L'information géométrique supplémentaire permet de contraindre efficacement la position de la cible, notamment la distance, et d'assurer une estimation robuste sur l'ensemble de l'horizon temporel. Ces résultats confirment que l'ajout d'un second capteur constitue une solution simple et efficace pour améliorer significativement la précision et la stabilité du suivi.

Conclusion

Durant ce projet, le filtre de Kalman étendu est apparu comme une solution pertinente pour traiter la non-linéarité de l'équation de mesure associée aux observations angulaires, rendant inapplicable le filtre de Kalman classique. Il a permis de mettre en évidence le rôle central de l'observabilité dans les problèmes de suivi de cibles. Le cas mono-capteur illustre en effet les limites intrinsèques d'une mesure d'azimut unique, notamment l'ambiguïté sur la distance à la cible, qui se traduit par une croissance progressive de l'incertitude et une estimation instable à long terme.

Ces résultats soulignent également les limites méthodologiques du filtre de Kalman étendu dans ce contexte. En effet, l'EKF repose sur une linéarisation locale de l'équation de mesure, qui peut devenir peu précise lorsque la non-linéarité est forte ou lorsque l'incertitude initiale sur l'état est élevée, ce qui peut conduire à des phénomènes de divergence. Les performances observées dans ce projet s'expliquent notamment par un régime favorable, caractérisé par une dynamique simple de type vitesse quasi constante et par des niveaux de bruit modérés, pour lesquels l'approximation linéaire reste localement valide.

L'introduction d'un second capteur angulaire permet en revanche de lever l'ambiguïté géométrique propre au cas mono-capteur, de stabiliser les variances estimées et d'améliorer significativement la précision globale du suivi, sans modifier la structure du filtre utilisé. Ce résultat met donc en évidence l'importance de la géométrie de l'observation dans les problèmes de filtrage non linéaire.

Enfin, ces travaux ouvrent naturellement la voie à des approches alternatives ou complémentaires, telles que l'utilisation de filtres particulaires ou de filtres de Kalman non linéaires de type UKF, mieux adaptés aux fortes non-linéarités, ainsi qu'à des stratégies de manœuvre des capteurs visant à améliorer l'observabilité du système. Ces pistes constituent des prolongements naturels à ce travail.

References

- [1] S. El Kolei, Filtering methods course, Third year, ENSAI, 2025.
- [2] N. Merlinge, J. Marzat, L. Reboul, Optimal guidance and observer design for target tracking using bearing-only measurements, ONERA, 2016.
- [3] T. Kailath, A.H. Sayed, B. Hassibi, Linear estimation, Chapitre 9, section 9.7.1, p.338-342, 2000.
- [4] Y. Bar-Shalom, X. Li, T. Kirubarajan, Estimation with Applications to Tracking and Navigation : Theory Algorithm and Software, John Wiley & Sons, 2004.