

Filtre Kalman étendu (EKF)

Filtre Kalman Étendu (EKF)

- Le filtre Kalman exige des équations linéaires
- Très peu de problèmes réels le sont
- Que faire?

Linéariser!

- Attention! EKF sera donc une solution **approximative**

Linéarisation à 2 variables d'entrées

- Soit une fonction $f(l,m)$ à 2 variables d'entrée
- Linéarisation autour de (a,b) :

$$f(l,m) \approx f(a,b) + \left. \frac{\partial}{\partial l} f(l,m) \right|_{l=a,m=b} (l-a) + \left. \frac{\partial}{\partial m} f(l,m) \right|_{l=a,m=b} (m-b)$$

- Exemple $f(l,m) = \cos(l)\ln(m)$
 - on linéarise autour de $(a=0.4 \text{ rad}, b=30)$

$$f(l,m) \approx f_{lin}(l,m) = \cos(a)\ln(b) - \sin(a)\ln(b)(l-a) + \frac{\cos(a)}{b}(m-b)$$

$$f_{lin}(l,m) = 3.1327 - 1.3245(l-a) + 0.0307(m-b)$$

Jacobienne : linéarisation généralisée

- Soit une fonction F de n entrées, avec m sorties :



- La matrice jacobienne au point P est définie comme suit :

$$J_F(P) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

matrice de $m \times n$

C'est l'extension de la tangente pour fonctions multi-variables

Très utilisé en robotique!

Linéarisation avec Jacobienne

Linéarisation $f(\vec{X}) \Big|_{\vec{X}_A} \approx f(\vec{X}_A) + J(\vec{X}_A)(\vec{X} - \vec{X}_A)$

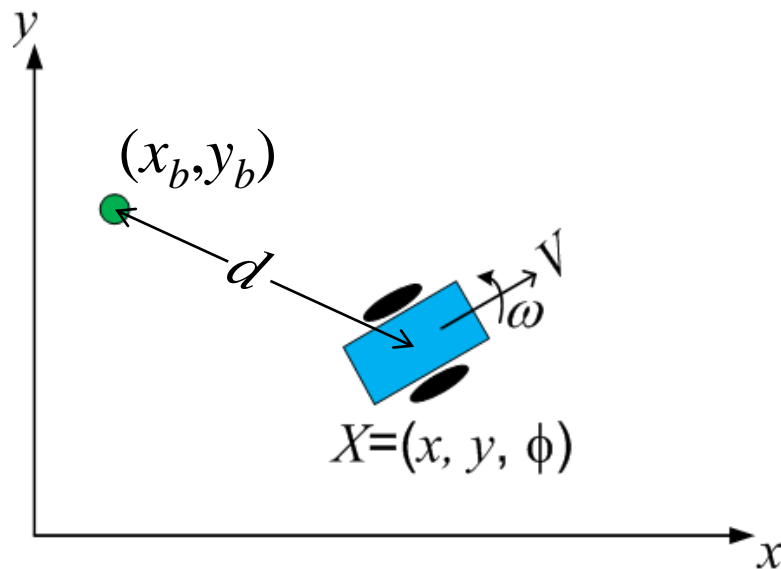
Exemple $\vec{X} = \begin{bmatrix} x \\ \theta \\ \phi \end{bmatrix}$ $f(\vec{X}) = \begin{bmatrix} f_1(\vec{X}) \\ f_2(\vec{X}) \end{bmatrix} = \begin{bmatrix} 3x \cos \theta \sin \phi \\ 7x + 14 \ln \phi \end{bmatrix}$

$$J(\vec{X}) = \begin{bmatrix} \frac{\partial}{\partial x} f_1(\vec{X}) & \frac{\partial}{\partial \theta} f_1(\vec{X}) & \frac{\partial}{\partial \phi} f_1(\vec{X}) \\ \frac{\partial}{\partial x} f_2(\vec{X}) & \frac{\partial}{\partial \theta} f_2(\vec{X}) & \frac{\partial}{\partial \phi} f_2(\vec{X}) \end{bmatrix} \begin{matrix} f_1 \\ f_2 \end{matrix}$$

$$J(\vec{X}) = \begin{bmatrix} 3 \cos \theta \sin \phi & -3x \sin \theta \sin \phi & 3x \cos \theta \cos \phi \\ 7 & 0 & \frac{14}{\phi} \end{bmatrix}$$

Filtre Kalman étendu (EKF)

- Pour problèmes non-linéaires. Exemple :
 - Véhicule avance V , tourne ω
 - Mesure distance d d'un point de repère fixe



Non linéaire parce que :

$$x_{t+1} = x_t + V\Delta t \cos \phi$$

$$y_{t+1} = y_t + V\Delta t \sin \phi$$

$$\phi_{t+1} = \phi_t + \omega\Delta t$$

$$d = \sqrt{(x_{t+1} - x_b)^2 + (y_{t+1} - y_b)^2}$$

- Comment utiliser d pour améliorer l'estimé \hat{x} ?

(d est une mesure incomplète)

Équations du filtre de Kalman étendu

changements vs. KF

$$(1) \hat{x}(k+1|k) = f_x(\hat{x}(k), u(k))$$

déplacement du robot

$$(2) \Phi = \left. \frac{df_x}{dx} \right|_{\hat{x}(k+1)} \quad \Gamma = \left. \frac{df_x}{du} \right|_{\hat{x}(k+1)}$$

calcul des Jacobiennes, pour propager erreur

$$(3) P(k+1|k) = \Phi P(k) \Phi^T + \Gamma C_v \Gamma^T$$

augmentation covariance P, après déplacement (le bruit ici est sur la commande)

$$(4) \hat{z}(k+1|k) = h_z(\hat{x}(k+1|k))$$

prédire ce que je devrait mesurer

$$(5) r(k+1) = z(k+1) - \hat{z}(k+1|k)$$

compare avec mesure

$$(6) \Lambda^T = \left. \frac{dh_z}{dx} \right|_{\hat{x}(k+1)}$$

calcul du Jacobien pour la mesure

$$(7) K(k+1) = P(k+1|k) \Lambda^T \{ \Lambda P(k+1|k) \Lambda^T + C_w(k+1) \}^{-1}$$

mélange optimal

$$(8) \hat{x}(k+1) = \hat{x}(k+1|k) + K(k+1)r(k+1)$$

combine estimé pose + mesure

$$(9) P(k+1) = (I - K(k+1)\Lambda)P(k+1|k)$$

ajuste covariance P, après gain info

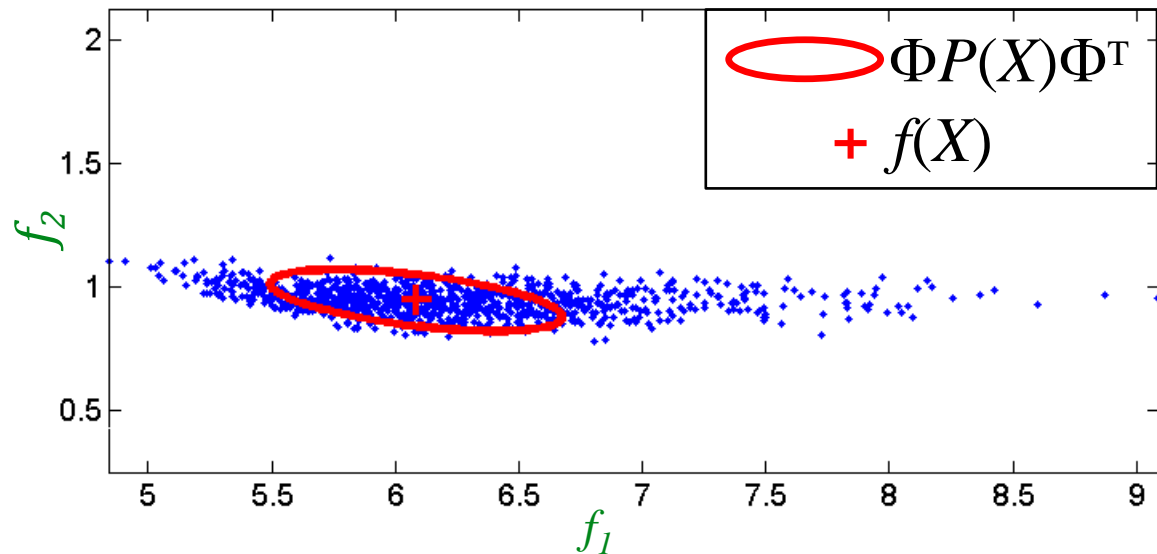
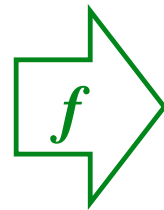
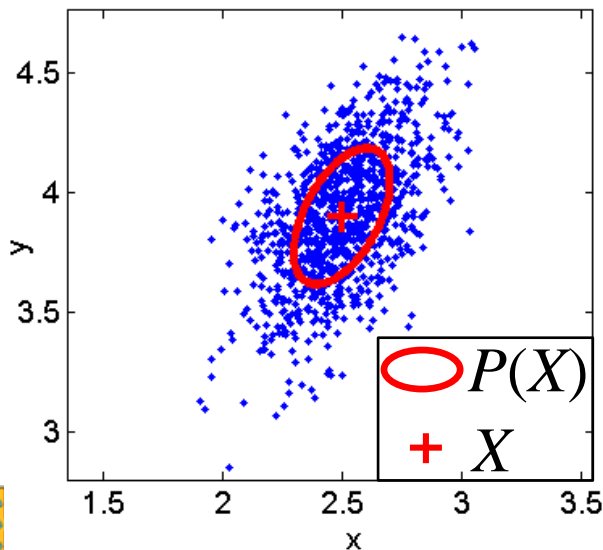
Jamais besoin d'inverser la fonction de capteur $h_z(\cdot)$!

Exemple propagation $P(X)$ (sans bruit)

$$X = \begin{bmatrix} x = 2.5 \\ y = 3.9 \end{bmatrix}, \quad \begin{aligned} f_1(X) &= 2 \cos(x) + 3 \sin(y) + xy \\ f_2(X) &= e^{-x} + \frac{y}{x} \end{aligned}, \quad P(X) = \begin{bmatrix} 0.04 & 0.03 \\ 0.03 & 0.08 \end{bmatrix}$$

$$\text{Jacobienne : } \Phi = \begin{bmatrix} y - 2 \sin(x) & 3 \cos(y) + x \\ -\frac{y}{x^2} - e^{-x} & \frac{1}{x} \end{bmatrix}$$

- Les points sont l'approximation par échantillonnage



Les approximations : linéarisations

- Le calcul de la propagation de la matrice P se fait avec la matrice jacobienne Φ .

$$P(k+1|k) = \Phi P(k) \Phi^T + \Gamma C_v \Gamma^T$$

- La correction à partir de la mesure z se fait avec la matrice jacobienne Λ .

$$K(k+1) = P(k+1|k) \Lambda^T \{ \Lambda P(k+1|k) \Lambda^T + C_w(k+1) \}^{-1}$$

$$P(k+1) = (I - K(k+1) \Lambda) P(k+1|k)$$

Systeme non-linéaire du robot 2D

- Propagation :

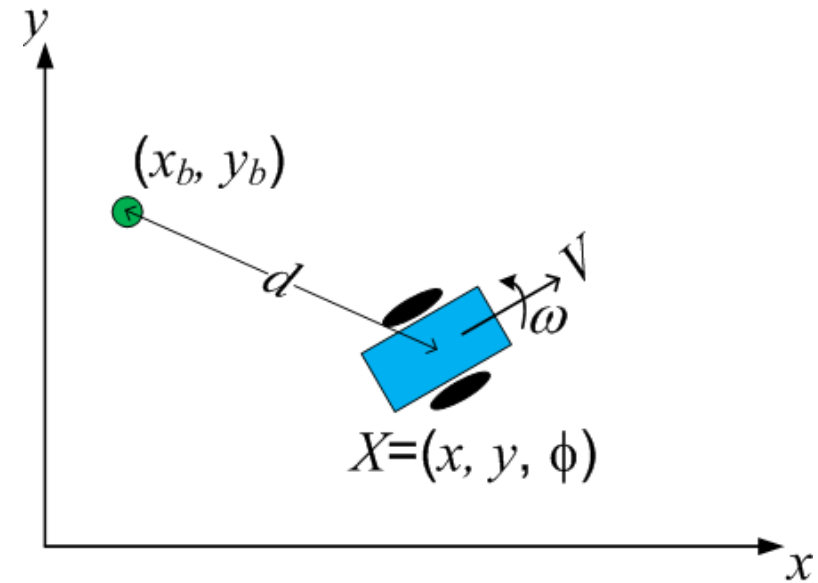
$$f_x = x + V \Delta t \cos \phi$$

$$f_y = y + V \Delta t \sin \phi$$

$$f_\phi = \phi + \Delta t \omega$$

- Mesure :

$$d = \sqrt{(x - x_b)^2 + (y - y_b)^2}$$



Kalman étendu : Jacobienne Φ

- Pour propager la covariance, on utilise Φ :

$$\mathbf{x} = \begin{bmatrix} x & y & \phi \end{bmatrix}^T$$

$$\begin{aligned} f_x &= x + V \Delta t \cos \phi \\ f_y &= y + V \Delta t \sin \phi \\ f_\phi &= \phi + \Delta t \omega \end{aligned} \quad \Phi = \begin{bmatrix} \frac{\partial f_x}{\partial x} & \frac{\partial f_x}{\partial y} & \frac{\partial f_x}{\partial \phi} \\ \frac{\partial f_y}{\partial x} & \frac{\partial f_y}{\partial y} & \frac{\partial f_y}{\partial \phi} \\ \frac{\partial f_\phi}{\partial x} & \frac{\partial f_\phi}{\partial y} & \frac{\partial f_\phi}{\partial \phi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -V \Delta t \sin \phi \\ 0 & 1 & V \Delta t \cos \phi \\ 0 & 0 & 1 \end{bmatrix}$$

Kalman étendu : Jacobienne Γ

- Pour propager le bruit des commandes V, ω vers $[x \ y \ \phi]$, on utilise Γ :

$$u = \begin{bmatrix} V & \omega \end{bmatrix}^T$$
$$x = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix}$$
$$\Gamma = \begin{bmatrix} \frac{\partial f_x}{\partial V} & \frac{\partial f_x}{\partial \omega} \\ \frac{\partial f_y}{\partial V} & \frac{\partial f_y}{\partial \omega} \\ \frac{\partial f_\phi}{\partial V} & \frac{\partial f_\phi}{\partial \omega} \end{bmatrix}$$
$$\Gamma = \begin{bmatrix} \Delta t \cos \phi & 0 \\ \Delta t \sin \phi & 0 \\ 0 & \Delta t \end{bmatrix}$$

$f_x = x + V \Delta t \cos \phi$
 $f_y = y + V \Delta t \sin \phi$
 $f_\phi = \phi + \Delta t \omega$

Kalman étendu : Propagation

- On propage l'estimé \hat{x} directement avec f :

$$\hat{x}(k+1|k) = \begin{bmatrix} f_x \\ f_y \\ f_\phi \end{bmatrix} \quad \begin{aligned} f_x &= x + V \Delta t \cos \phi \\ f_y &= y + V \Delta t \sin \phi \\ f_\phi &= \phi + \Delta t \omega \end{aligned}$$

- On propage la covariance P avec la jacobienne Φ :

$$P(k+1|k) = \Phi P(k) \Phi^T + \Gamma C_v \Gamma^T$$

Kalman étendu : Jacobien mesure Λ

- Pour la correction à partir de la mesure, on calcule la jacobienne Λ

fonction de capteur

$$h_z(X) = \sqrt{(x - x_b)^2 + (y - y_b)^2}$$

$$\mathbf{x} = \begin{bmatrix} x & y & \phi \end{bmatrix}^T$$

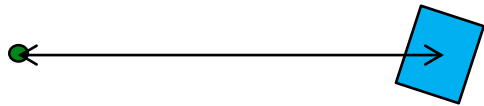
$$\Lambda = \begin{bmatrix} \frac{\partial h_z}{\partial x} & \frac{\partial h_z}{\partial y} & \frac{\partial h_z}{\partial \phi} \end{bmatrix} = \begin{bmatrix} \frac{x - x_b}{\sqrt{(x - x_b)^2 + (y - y_b)^2}} & \frac{y - y_b}{\sqrt{(x - x_b)^2 + (y - y_b)^2}} & 0 \end{bmatrix}$$

- Indique quelles variables d'état affectent directement la mesure z

Jacobienne mesure Λ

$$\Lambda = \begin{bmatrix} \frac{\partial h_z}{\partial x} & \frac{\partial h_z}{\partial y} & \frac{\partial h_z}{\partial \phi} \end{bmatrix} = \begin{bmatrix} \frac{x - x_b}{\sqrt{(x - x_b)^2 + (y - y_b)^2}} & \frac{y - y_b}{\sqrt{(x - x_b)^2 + (y - y_b)^2}} & 0 \end{bmatrix}$$

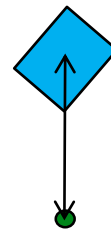
cas 1 : $y = y_b$



$$\Lambda = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

... toute la
mesure/correction se fait
selon la direction x ...

cas 2 : $x = x_b$



$$\Lambda = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

... toute la
mesure/correction se fait
selon la direction y ...

Kalman étendu : code matlab (boucle)

```
% simulation du système
xVrai(1) = xVrai(1) + (V+SV*randn)*cos(xVrai(3))*dT;
xVrai(2) = xVrai(2) + (V+SV*randn)*sin(xVrai(3))*dT;
xVrai(3) = xVrai(3) + (omega+Somega*randn)*dT;

% Je simule la reponse du capteur de distance
z = sqrt((xVrai(1)-Xb)^2 + (xVrai(2)-Yb)^2) + SBeacon*randn;
```

Kalman Étendu

```

% ===== Propagation =====
U = [V*dT omega*dT]';
X(1) = X(1) + V*cos(X(3))*dT;
X(2) = X(2) + V*sin(X(3))*dT;
X(3) = X(3) + omega*dT;

% J'estime la distance
zhat = sqrt((X(1)-Xb)^2 + (X(2)-Yb)^2);

% ===== Calcul des matrices Jacobiennes =====
F = [1 0 -V*sin(X(3))*dT; 0 1 V*cos(X(3))*dT; 0 0 1];
G = [dT*cos(X(3)) dT*sin(X(3)) 0; 0 0 dT]';
H = [(X(1)-Xb)/zhat (X(2)-Yb)/zhat 0];

P = F*P*F' + G*Cv*G'; % Propagation covariance

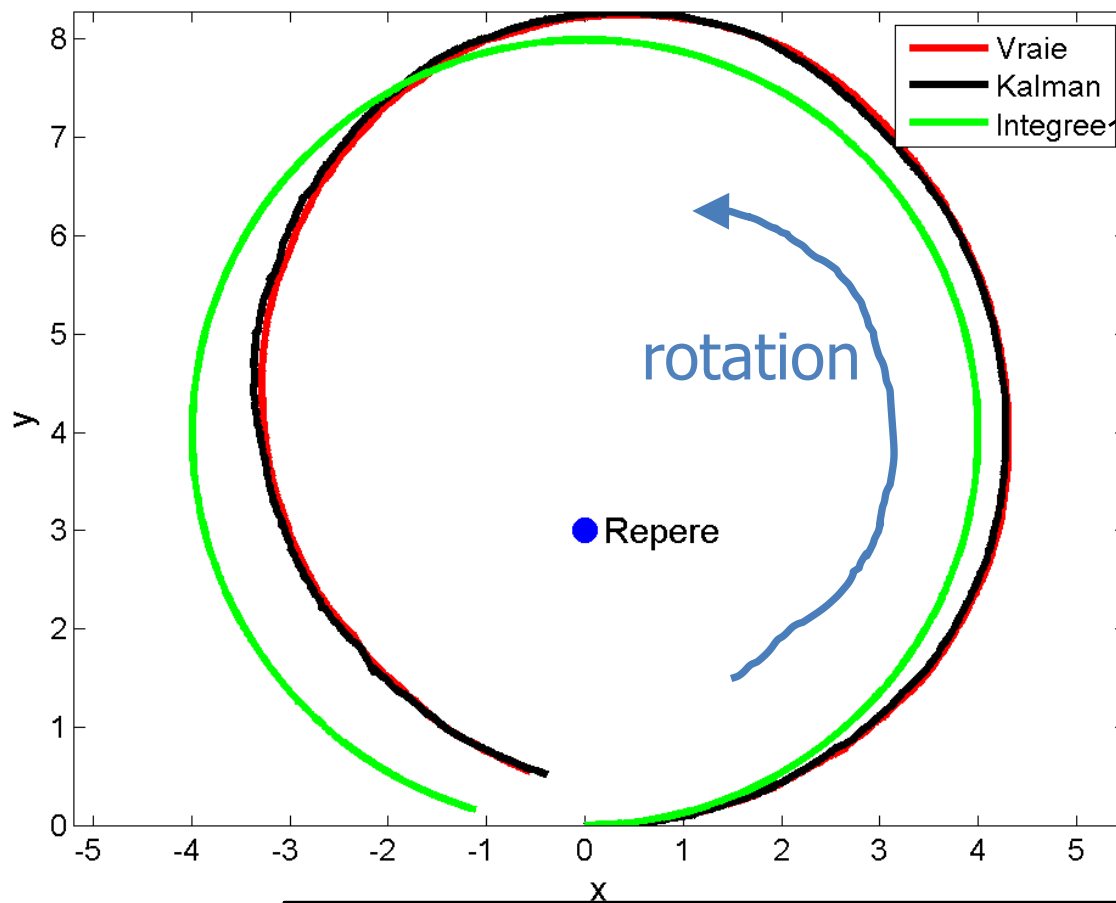
% ===== Mise-a-jour =====
K = P*H'/(H*P*H'+Cw); % matrice [3x1]
r = (z-zhat); % matrice [1]
X = X + K*r;
P = (eye(3)-K*H)*P;
```

Φ

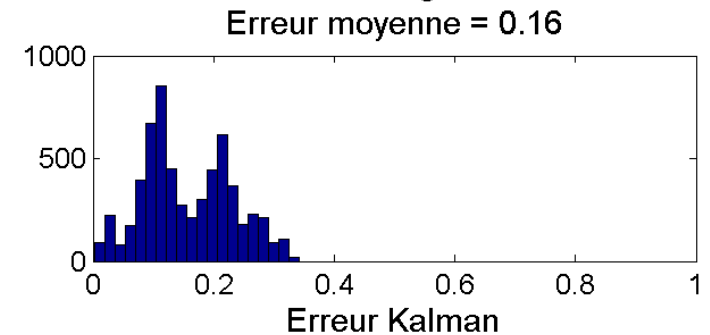
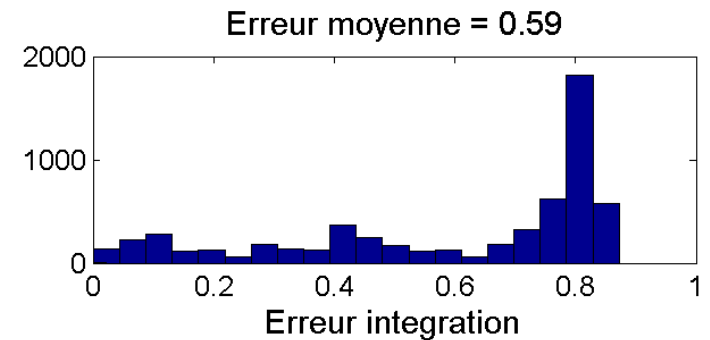
Λ

$$\text{eye}(3) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Kalman étendu : résultats



intégration des commandes $u(t)$



On arrive à corriger x, y, ϕ même si le capteur ne les mesure pas directement! (mesures incomplètes)

Exemple 2 : pseudo GPS

- État $[x \ y \ \theta]^T$
- Propagation non-linéaire
- Mesure x, y linéaire
- Bruits de déplacement: $\sigma_v = 0.2 \text{ m/s}$, $\sigma_\omega = 0.1 \text{ rad/s}$
- Commande fixe $u = [10 \text{ m/s} \ 0.1 \text{ rad/s}]^T$: tourne en rond!
- Bruits de capteurs : $\sigma_x = \sigma_y = 12 \text{ mètres}$
- Trois cas testés :
 - Bonne initialisation de l'état;
 - Erreur sur initialisation de l'angle θ ;
 - Biais de 0.01 rad/s sur la commande en rotation.

FiltreKalmanNonLineairePseudoGPS.m

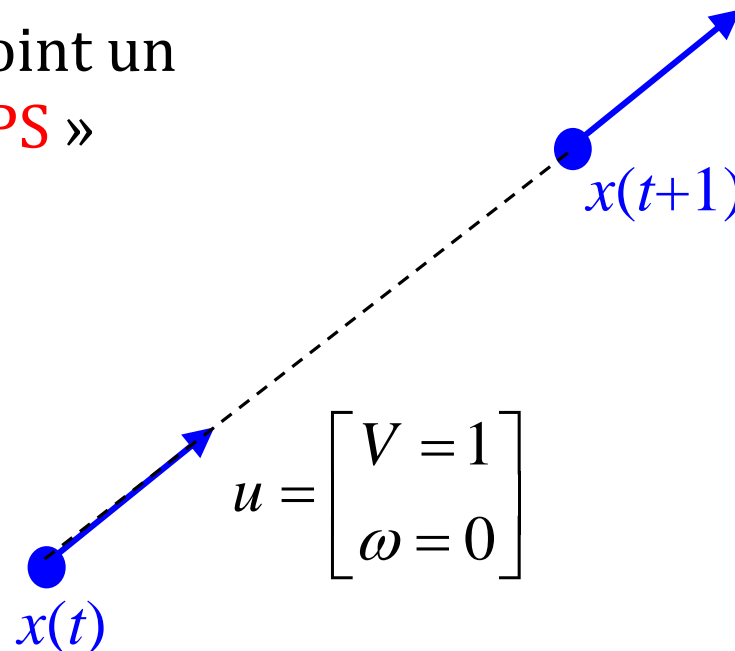
Bayes : réconcilier l'innovation

Qui est fautif? La position? L'angle?

Kalman : « Mmmm, mais aussi l'orientation est peut-être aussi à blâmer... je vais la corriger aussi... »

■ GPS

Kalman : « Je dois déplacer le point un peu vers le GPS »



Kalman : « Le tout en proportion des bruits/covariance »

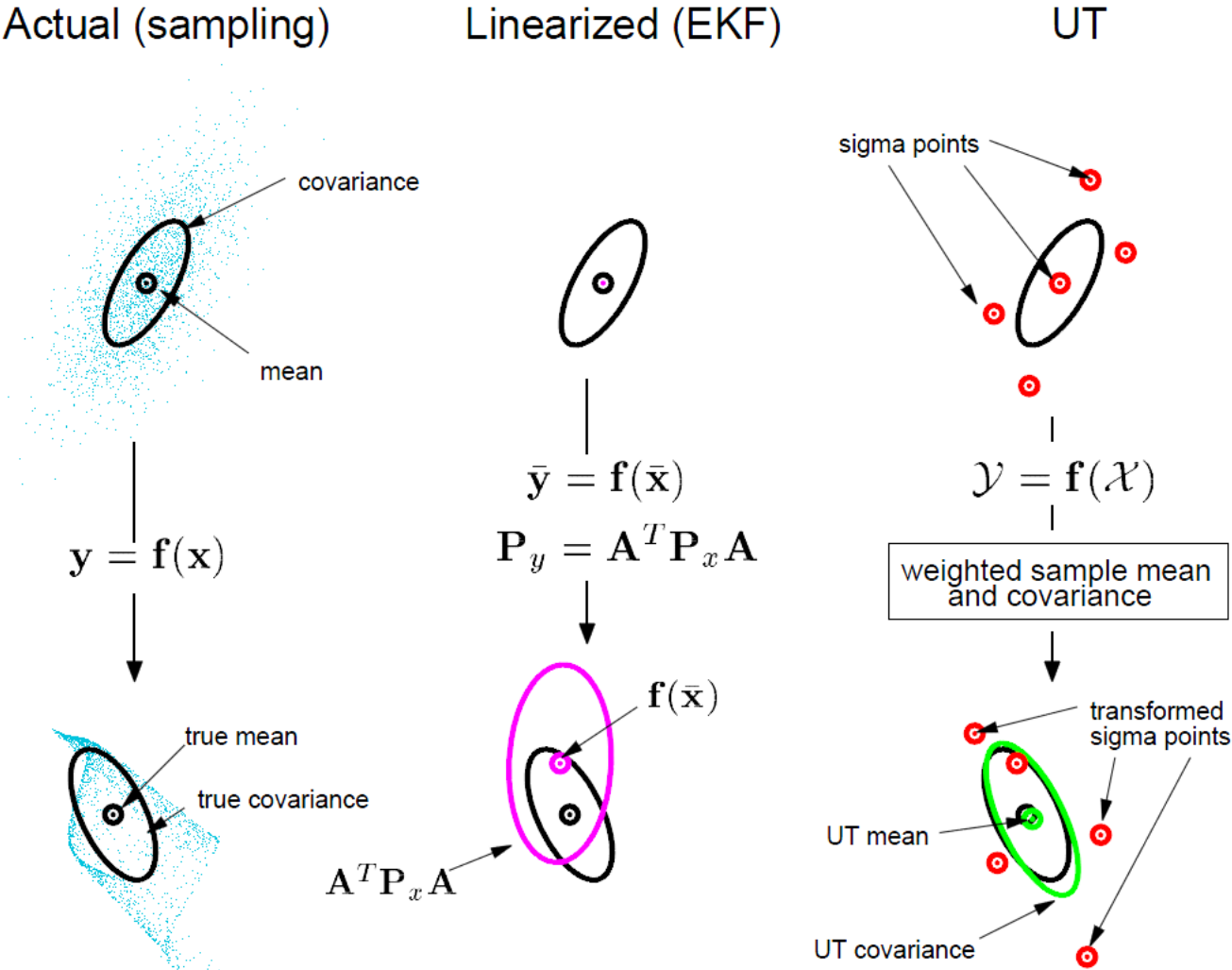
Filtre Kalman non-parfumé (unscented)

Fonctions fortement non-linéaires?

- Le filtre EKF se basait sur la linéarisation (série de Taylor) pour calculer la nouvelle matrice P , via la jacobienne Φ
- Si la fonction f est fortement non-linéaire, l'approximation ne sera pas bonne
- Autre approche : utiliser des échantillons d'état X choisis judicieusement (*sigma points*), que l'on passe directement dans la fonction f
- Extraire les statistiques par la suite

UT : Unscented Transform

- Tiré de « A tutorial on Dynamic Bayesian Networks », Kevin Murphy.



UT : Choix des *sigma points* $X^{[i]}$

- Pour état à n dimensions, on utilisera $2n+1$ points
- Premier sigma point : $X^{[0]} = \mu$
- Choisit un facteur de distance λ par rapport au centre, via α , β et κ : $\lambda = \alpha^2 (n + \kappa) - n$
 λ est généralement petit : $1e-3$
- Calcule la position des $2n$ sigma points restants :

$$X^{[i]} = \mu + \left(\sqrt{(n + \lambda)\Sigma} \right)_i, \text{ pour } i = 1, \dots, n$$

sqrtm (pointing to the square root)

i^{ème} colonne (pointing to the subscript i)

$$X^{[i]} = \mu - \left(\sqrt{(n + \lambda)\Sigma} \right)_i, \text{ pour } i = n + 1, \dots, 2n$$

UT: Poids w associés aux *sigma points*

- Poids pour calculer la **moyenne** et la **covariance**
- Pour 1^{er} *sigma point* :

$$w_m^{[0]} = \frac{\lambda}{n + \lambda} \quad w_C^{[0]} = \frac{\lambda}{n + \lambda} + (1 - \alpha^2 + \beta)$$

- Pour tous les autres *sigma points* :

$$w_m^{[i]} = w_C^{[i]} = \frac{1}{2(n + \lambda)}$$

- Statistiques en passant les points dans la fonction $f(\cdot)$:

$$\mu' = \sum_{i=0}^{2n} w_m^{[i]} f(X^{[i]})$$

$$\Sigma' = \sum_{i=0}^{2n} w_C^{[i]} \left(f(X^{[i]}) - \mu' \right) \left(f(X^{[i]}) - \mu' \right)^T$$

- Plus besoin de calculer les dérivés 😊

UT : exemple

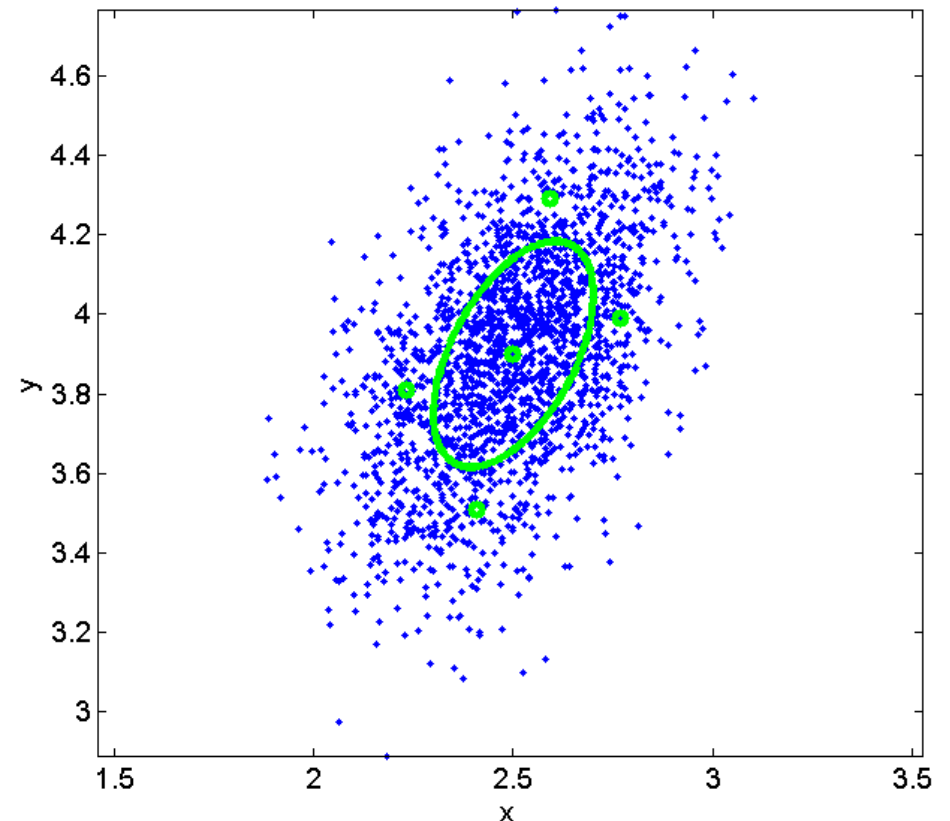
- Reprise de l'exemple de linéarisation précédent :

$$X = \begin{bmatrix} x = 2.5 \\ y = 3.9 \end{bmatrix}, \quad \begin{aligned} f_1(X) &= 2 \cos(x) + 3 \sin(y) + xy \\ f_2(X) &= e^{-x} + \frac{y}{x} \end{aligned}, \quad P(X) = \begin{bmatrix} 0.04 & 0.03 \\ 0.03 & 0.08 \end{bmatrix}$$

- $n = 2$ dimensions, donne $2n+1 = 5$ sigma points
- Paramètres UT utilisés :
 $\alpha = 0.001, \beta = 2, \kappa = 1$

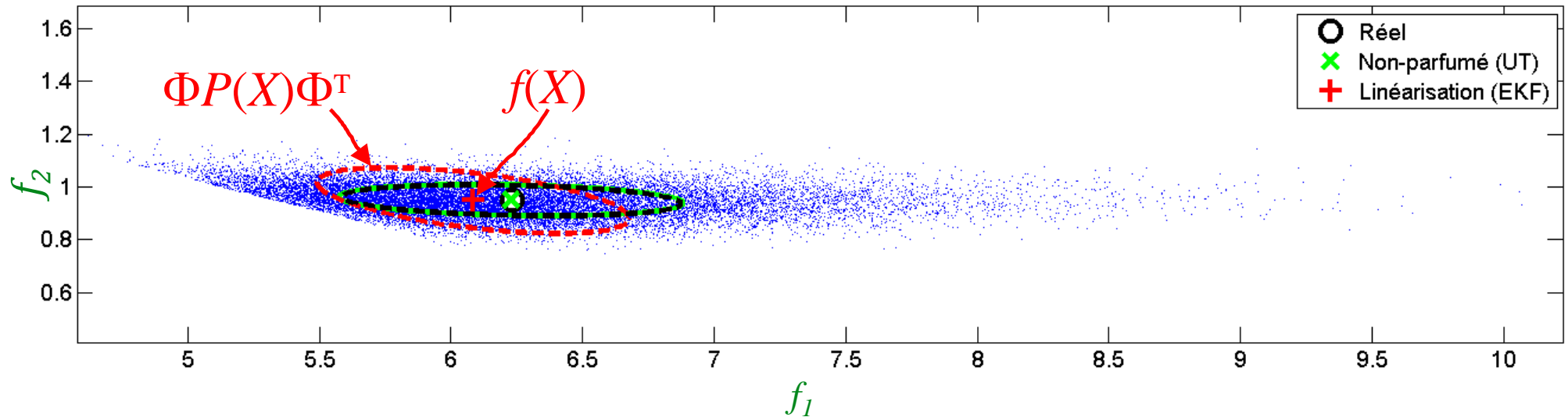
Points sigma dans matlab

2.5000	2.7677	2.5913	2.2323	2.4087
3.9000	3.9913	4.2894	3.8087	3.5106



UT : exemple

- Plus précis que la simple linéarisation dans EKF



- Il peut être démontré que UT est exact pour les deux premiers termes de Taylor

Équations du filtre UT (1)

- Étape 1 : calcul de la propagation non-linéaire

$$X_{t-1} = (\mu_{t-1} \quad \mu_{t-1} + \gamma \sqrt{\Sigma_{t-1}} \quad \mu_{t-1} - \gamma \sqrt{\Sigma_{t-1}})$$

génère les sigma points

$$\bar{X}_t^* = f(u_t, X_{t-1})$$

passé les sigma points dans la fonction non-linéaire

$$\bar{\mu}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{X}_t^{*[i]}$$

$$\bar{\Sigma}_t = \sum_{i=0}^{2n} w_C^{[i]} \left(\bar{X}_t^{*[i]} - \bar{\mu}_t \right) \left(\bar{X}_t^{*[i]} - \bar{\mu}_t \right)^T + R_t$$

calculé de la distribution après déplacement via $f(\cdot), u_t$

tient compte du bruit de déplacement R_t

$$\gamma = \sqrt{n + \lambda}$$

Équations du filtre UT (2)

- Étape 2 : prédiction de la mesure

$$\bar{X}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \gamma\sqrt{\bar{\Sigma}_t} \quad \bar{\mu}_t - \gamma\sqrt{\bar{\Sigma}_t})$$

génère des nouveaux sigma points

$$\bar{Z}_t = h(\bar{X}_t)$$

passe ces sigma points dans la fonction de mesure

$$\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{Z}_t^{[i]}$$

prédit la mesure moyenne

$$S_t = \sum_{i=0}^{2n} w_C^{[i]} \left(\bar{Z}_t^{[i]} - \hat{z}_t \right) \left(\bar{Z}_t^{[i]} - \hat{z}_t \right)^T + Q_t$$

covariance sur cette prédiction

$$K(k+1) = P(k+1|k)\Lambda^T \{ \Lambda P(k+1|k)\Lambda^T + C_w \}^{-1} \leftarrow \text{EKF}$$

$$\gamma = \sqrt{n + \lambda}$$

Équations du filtre UT (3)

- Étape 3 : mise-à-jour

$$\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_C^{[i]} \left(\bar{X}_t^{[i]} - \bar{\mu}_t \right) \left(\bar{Z}_t^{[i]} - \hat{z}_t \right)^T$$

covariance entre la mesure et l'estimation d'état

$$K(k+1) = P(k+1|k) \Lambda^T \{ \Lambda P(k+1|k) \Lambda^T + C_w \}^{-1} \leftarrow \text{EKF}$$

$$K_t = \bar{\Sigma}_t^{x,z} S_t^{-1} \quad \text{calcul du gain Kalman}$$

$$\mu_t = \bar{\mu}_t + K_t (z_t - \hat{z}_t)$$
$$\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$$

mise-à-jour de l'état et de la covariance

tiré de *Probabilistic Robotics*, p. 70

Résumé des filtres de Kalman

- Estimateur optimal (ou proche-optimal) d'états cachés (pose du robot)
- Fonctionne par récursions
- Trois variantes vues en classe :
 - Kalman (linéaire)
 - Kalman étendu EKF (non-linéaire)
 - Kalman non-parfumé (*unscented* : UKF)
- Principales difficultés d'implémentations :
 - mise en équation du système
 - bien évaluer les bruits (déplacements, commandes, capteurs)
 - risque d'erreurs dans les nombreux termes