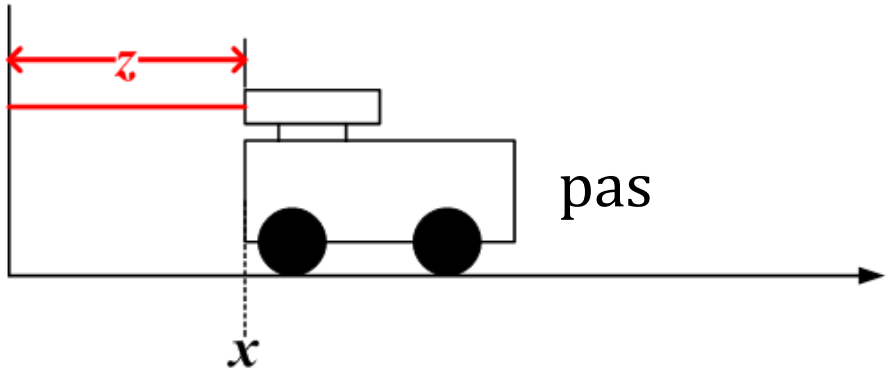
Exemples de filtre de Kalman linéaires

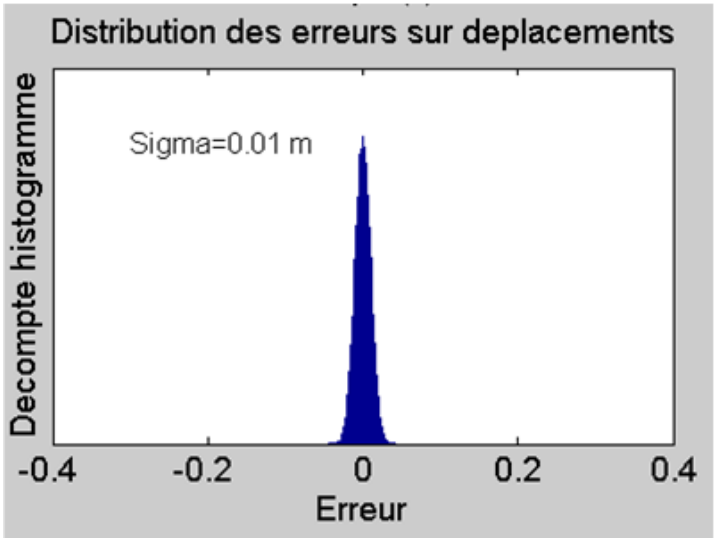
2 Exemples de filtre Kalman

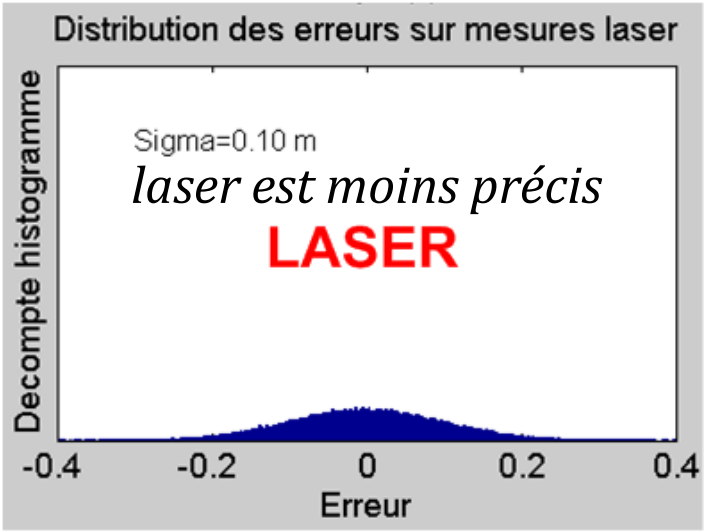
- 1er exemple : 1 variable d'état
- 2ème exemple : 2 variables d'état

1er exemple : état 1 variable

- Chariot sur roues avance pas-à-pas, laser mesure en mètre







Note : l'odométrie est souvent une mesure localement très précise, mais dérive.

1er exemple : équations

- Propagation

$$\hat{x}(k+1|k) = \hat{x}(k) + pas$$

$$P(k+1|k) = P(k) + \sigma_{pas}^{2}$$

$$\Phi = [1] \quad \textit{prop. d'état}$$

$$\Gamma = [1] \quad \textit{commande}$$

$$\Lambda = [1] \quad \textit{mesure}$$

1er exemple : équations

- Propagation

$$\hat{x}(k+1|k) = \hat{x}(k) + pas$$

$$P(k+1|k) = P(k) + \sigma_{pas}^{2}$$

$$\Phi = [1] \quad \textit{prop. d'état}$$

$$\Gamma = [1] \quad \textit{commande}$$

$$\Lambda = [1] \quad \textit{mesure}$$

- Mise-à-jour

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

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

$$K(k+1) = P(k+1|k)\{P(k+1|k) + C_w(k+1)\}^{-1} = \frac{P(k+1|k)}{P(k+1|k) + \sigma_{laser}^{2}}$$

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

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

(on retrouve les mêmes équations qu'avec l'exemple sur la moyenne récursive!)

Filtre Kalman code matlab

- Code pour la boucle

un pas = `V*dt`

```
% Simulation du systeme
xVrai = xVrai + V*dt + Sx*randn;
z   = xVrai + Slaser*randn; % mesure laser
```

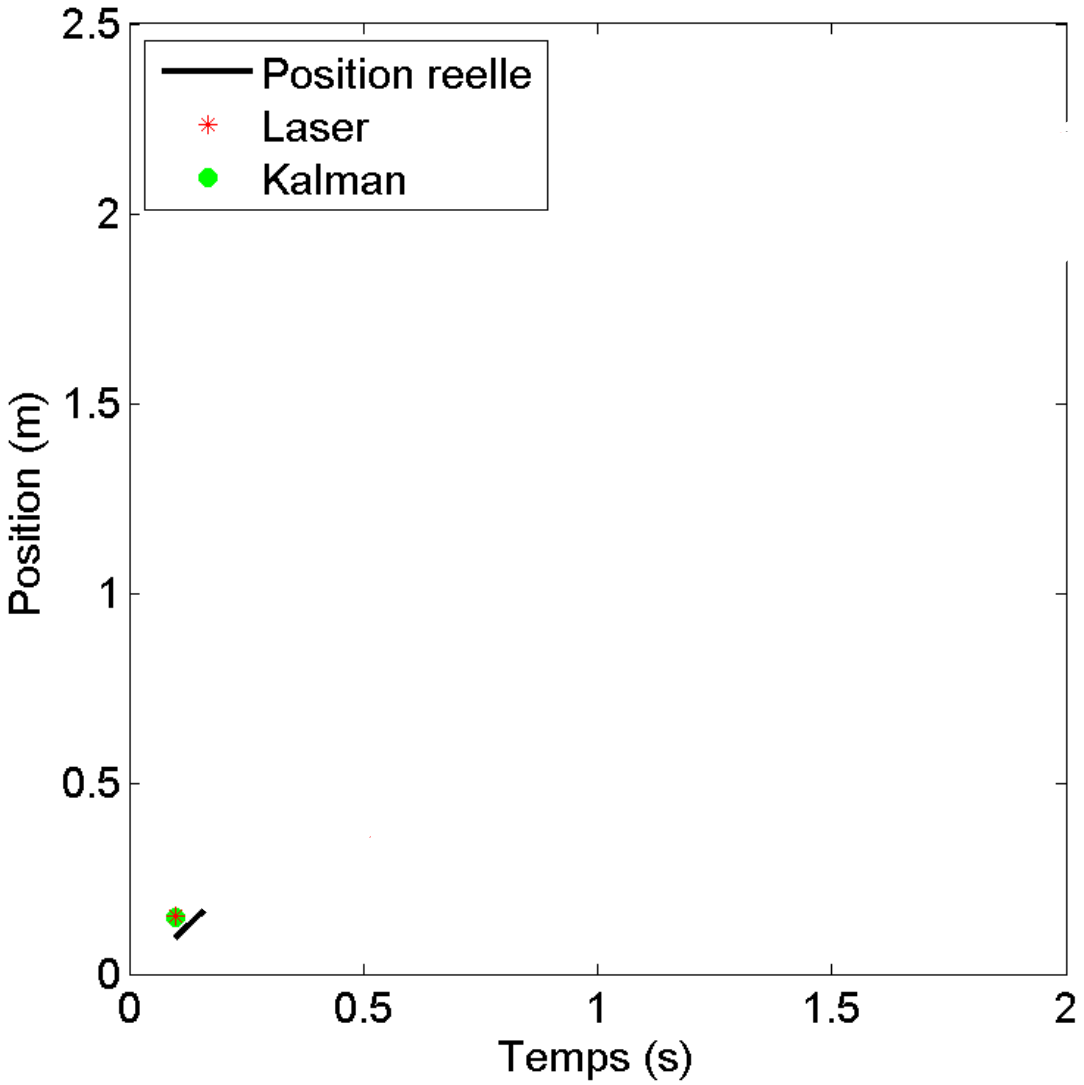
Kalman

```
% Propagation
x   = x + V*dt;
P   = P + Cv;

% Mise-a-jour (update)
K   = P/(P+Cw);
x   = x + K*(z-x);
P   = (eye(1)-K)*P;
```

V constante

Résultat simulation Kalman

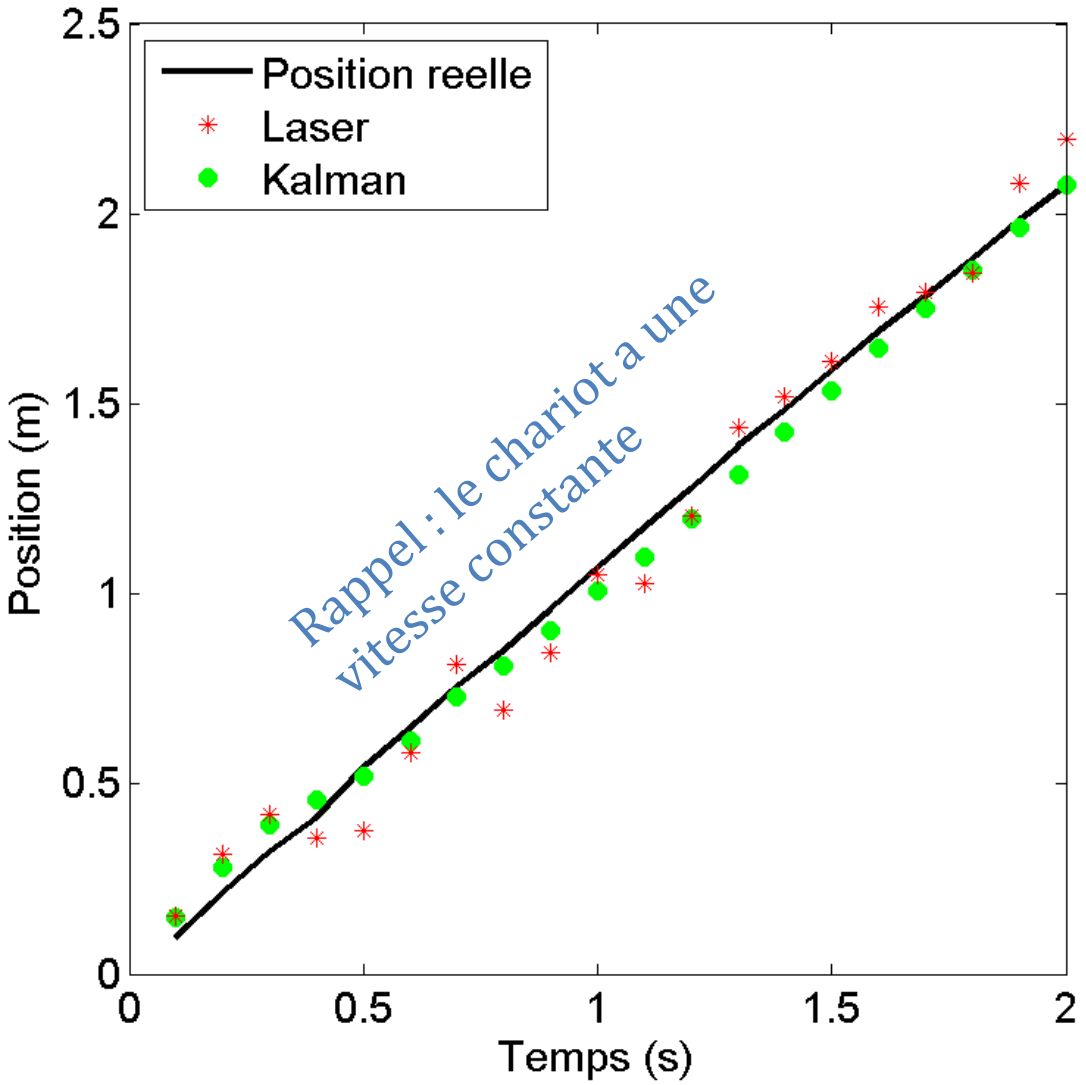
- 1ère itération : on initialise la position du filtre à la mesure laser, $P=\sigma_{laser}$



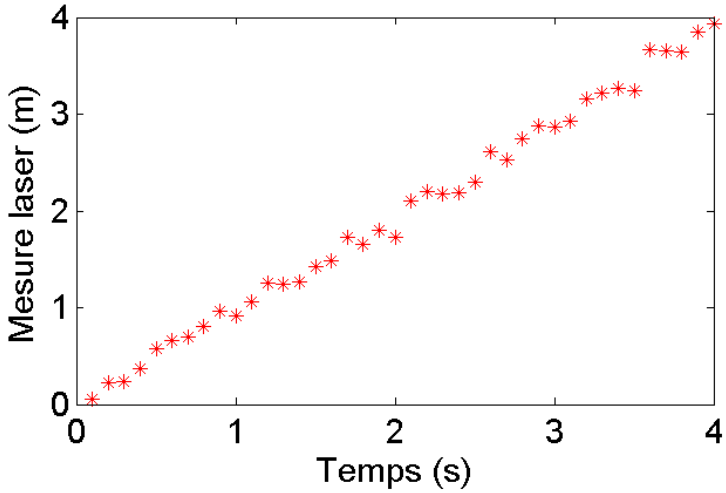
Résultat simulation Kalman

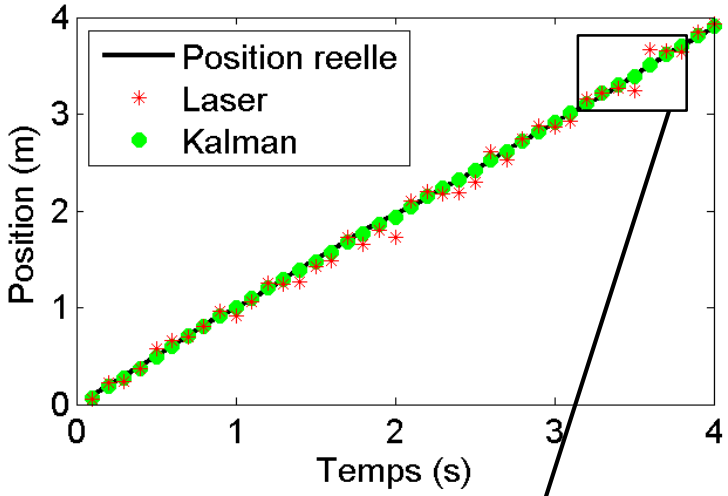
- 1ère itération : on initialise la position du filtre à la mesure laser, $P=\sigma_{laser}$

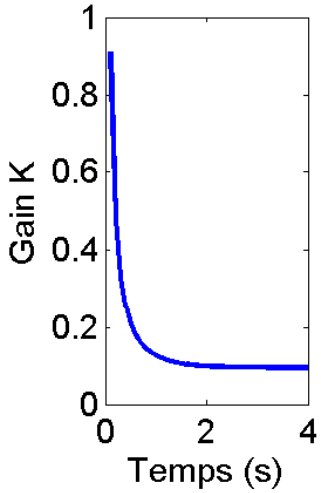
- Au fil du temps, l'erreur du filtre diminue : moins grande dispersion que les mesures laser



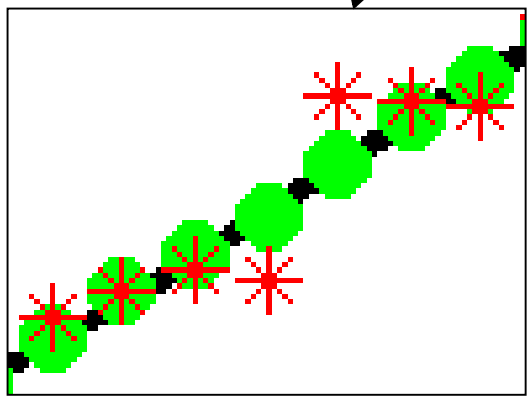
Résultat simulation Kalman

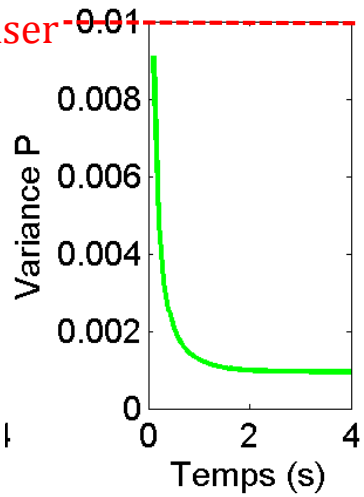






réduction ici d'un facteur 5 à 10 sur l'erreur en position, par rapport au laser





2ème Exemple : gyro + compas magnétique

- Deux capteurs placé sur un objet quelconque:
  - gyroscope à taux $N(\underline{0.1},0.2^2)$ deg/s (**avec biais**)
  - compas magnétique $N(0,10^2)$ deg (**sans biais**)
- Rotation dans un plan
- Veut *fusionner* les deux ensemble :

Proprio. – gyro est peu bruité, mais dérive ⟶ **bon à court terme**

Exterio. – compas ne dérive pas, mais est bruité ⟶ **bon à long terme**

Modèle du système : propagation

- État : angle θ et vitesse angulaire $\dot{\theta}$: $x=[\theta \ \dot{\theta}]^T$
- Gyro : <u>proprioceptif</u>, équivaut à une commande

état à k+1 | état à k | commande

$$\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}1 & \Delta t \\ 0 & 0\end{bmatrix}\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0 \\ 1\end{bmatrix}\begin{bmatrix}\dot{\theta}_{gyro}\end{bmatrix}$$

$$\Phi = \begin{bmatrix}1 & \Delta t \\ 0 & 0\end{bmatrix} \qquad \Gamma = \begin{bmatrix}0 \\ 1\end{bmatrix}$$

$$C_v = \begin{bmatrix}0 & 0 \\ 0 & 16\sigma^2_{gyro}\end{bmatrix}$$

sur-estime, pour tenir compte biais

surestimer augmente la robustesse des algos aux violations

Modèle capteur : mise-à-jour

- Je ne «mesure » qu'avec le compas → extéroceptif
- Autrement dit, je ne veux pas que le gyro intervienne dans la mise-à-jour, car il est proprioceptif

ce que je m'attends à mesurer…

$$\hat{z}(k+1)=\Lambda\hat{x}(k+1|k)=\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\end{bmatrix}=\begin{bmatrix}\theta\end{bmatrix}$$

$$\Lambda=\begin{bmatrix}1 & 0\end{bmatrix} \qquad C_w=\begin{bmatrix}\sigma^2_{compas}\end{bmatrix}$$

Notez la dimension de Λ : (**1**x**2**)

- 1 capteur
- 2 variables d'état

Code *matlab*

Paramètres

```
Cv = [0 0; 0 (4*SGyro)^2];
Cw = SCompas^2;
F = [1 dT; 0 0];
H = [1 0];
B = [0 1]';
```

Φ
Λ
Γ

en boucle

simulation

```
% Je vais osciller le capteur combine
xVrai = 40*[sin(wrot*time) wrot*cos(wrot*time)]';
```

angle — dérivée de l'angle

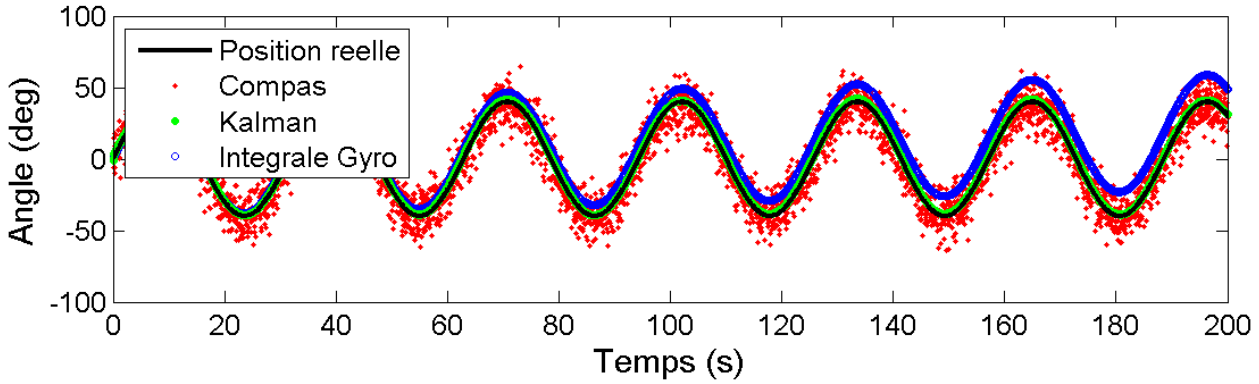
```
% Je simule la reponse des capteurs
compass =  xVrai(1) + SCompas*randn;
gyro    =  xVrai(2) + SGyro*randn + GyroBiais;

% Le vecteur de mesure
z = compass;

% La commande
U = [gyro];
```

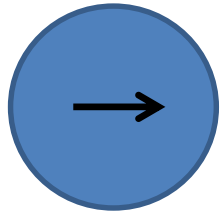
équations Kalman

```
% ====== Propagation =========
X = F*X + B*U;
P = F*P*F' + Cv;

% ======== Mise-a-jour ========
K = P*H'/(H*P*H'+Cw);
X = X + K*(z-H*X);
P = (eye(2)-K*H)*P;
```

```
% Integrer le gyro, pour fin de comparaison
XGyro    = XGyro + gyro*dT;
XGyroInt(iStep)= XGyro;
```

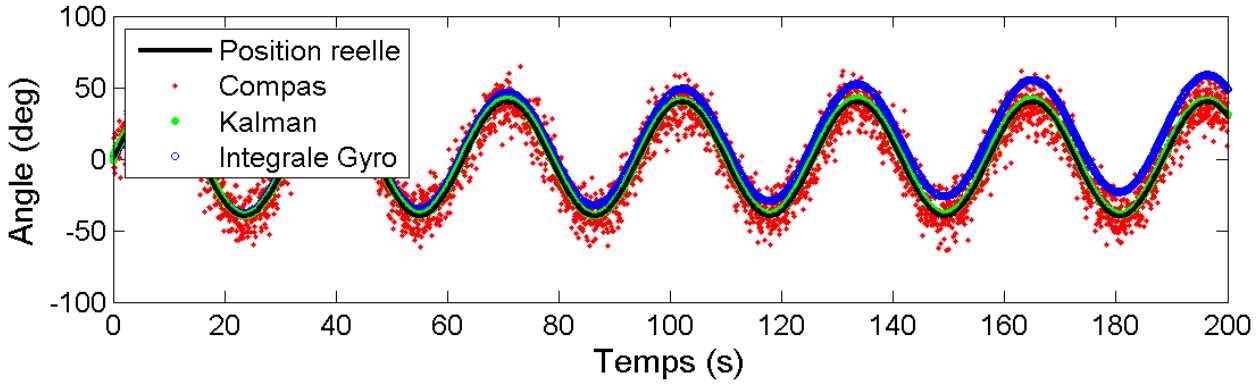
Résultat simulation (20 mesures/sec)

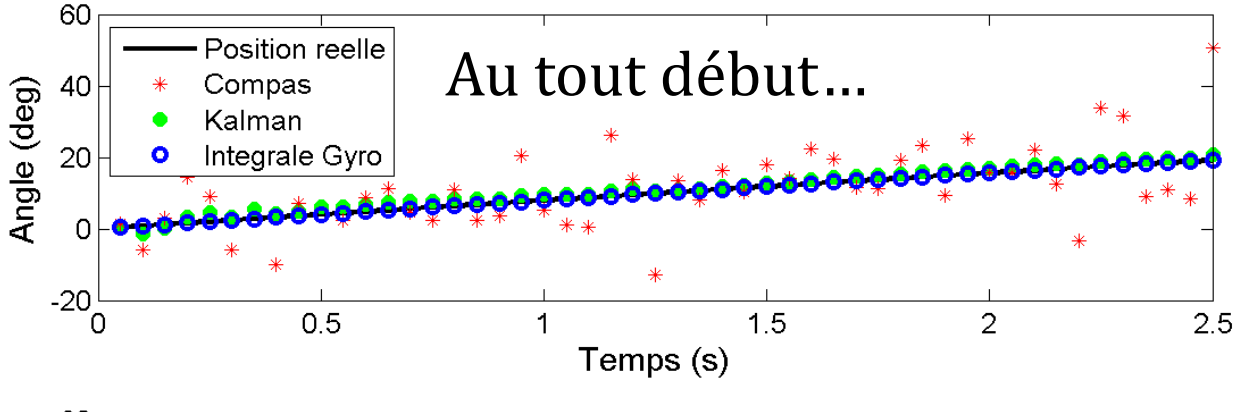




trajectoire : $\theta = 40^{o} \sin(0.2t)$

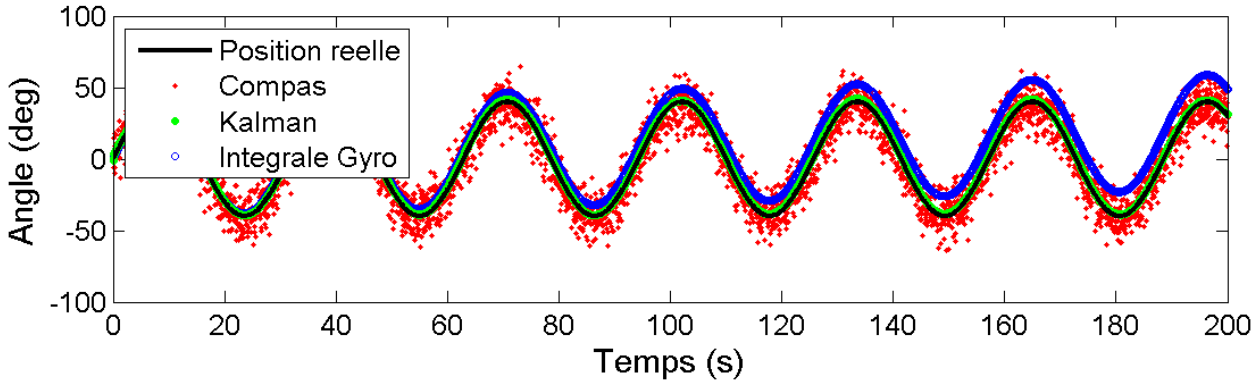
Résultat simulation (20 mesures/sec)

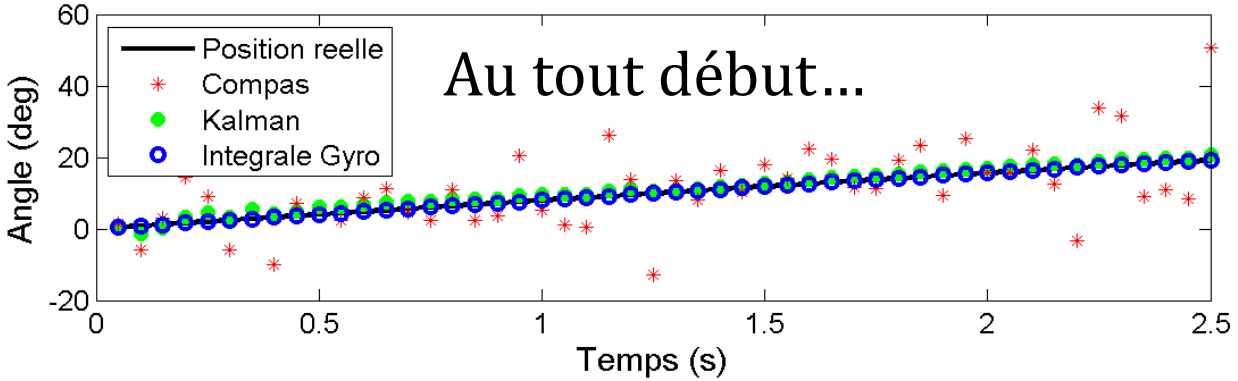


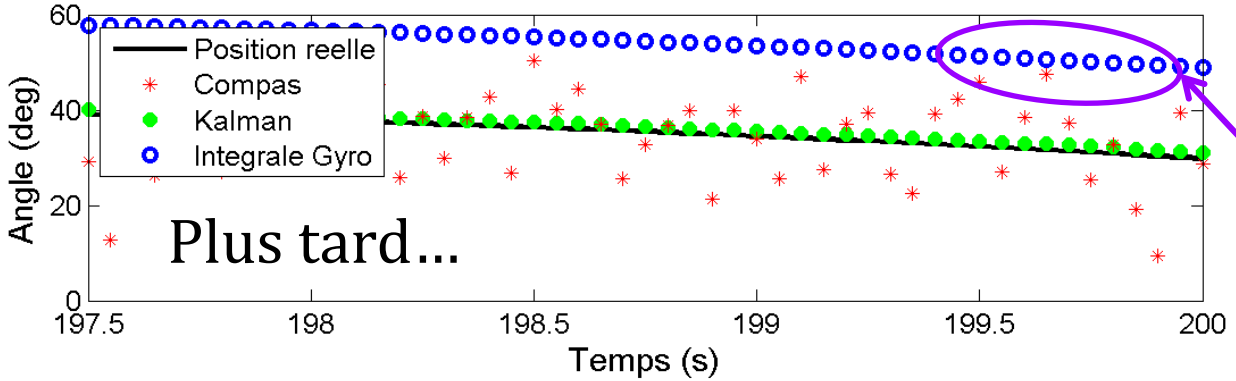


∫ gyro donne une mesure peu bruitée et proche de la réalité

Résultat simulation (20 mesures/sec)



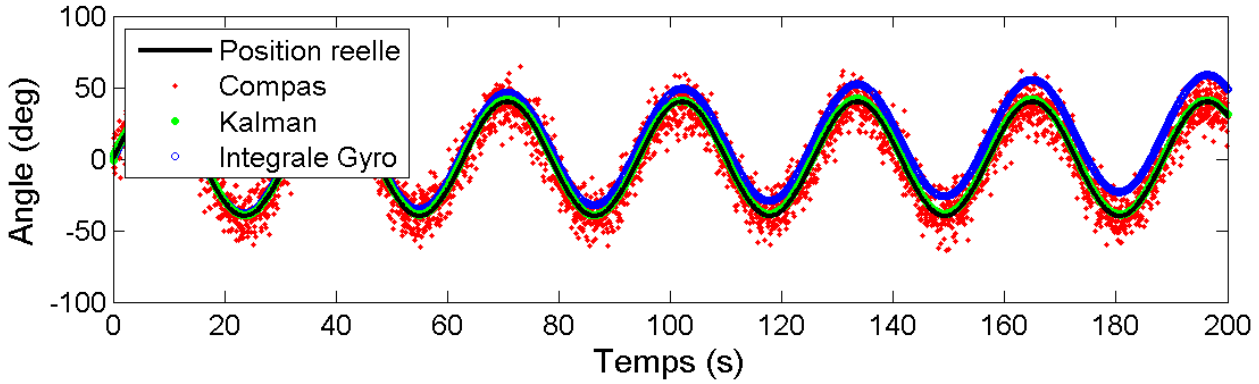


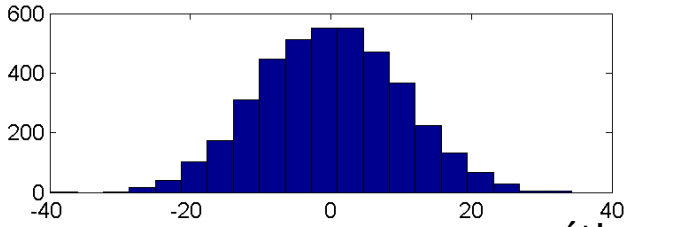


$\int$ *gyro finit par s'éloigner de la réalité : accumulation du biais*

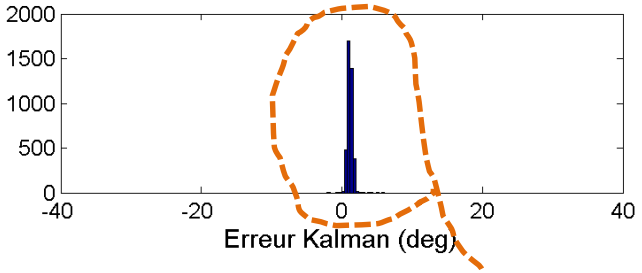
gyro diverge

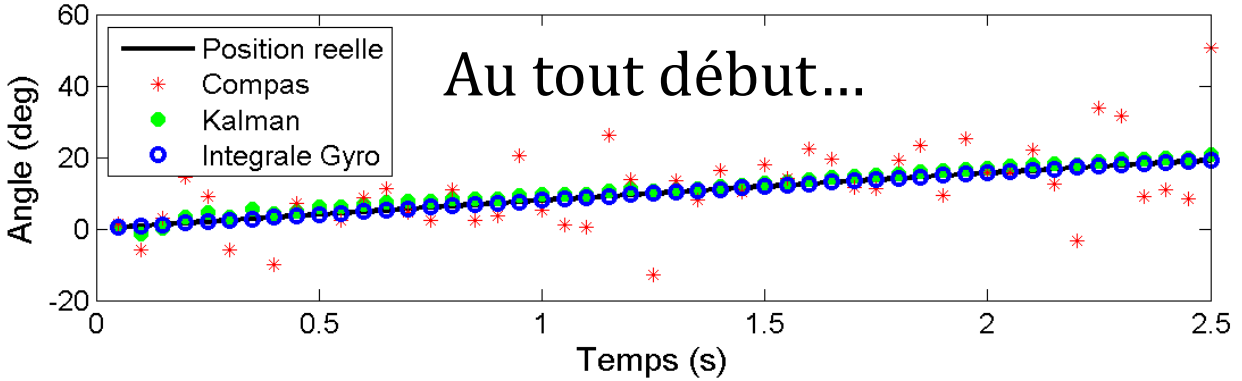
Résultat simulation (20 mesures/sec)



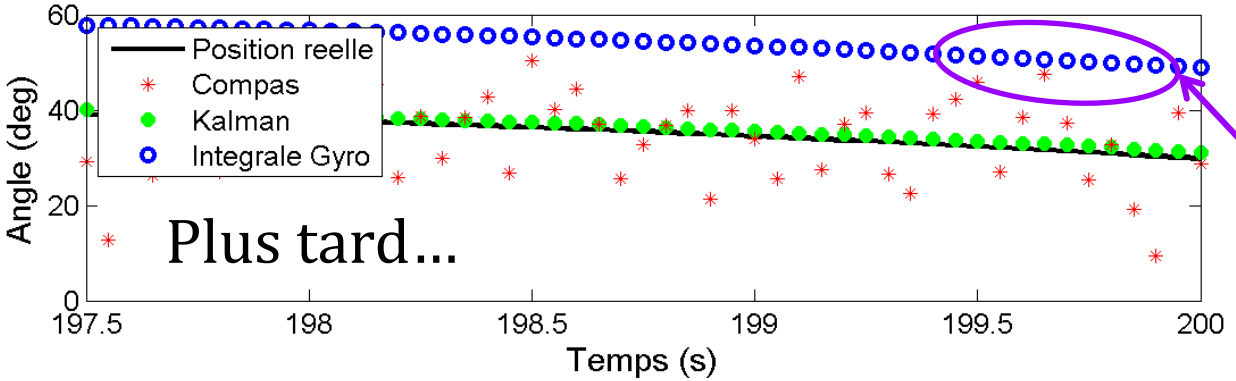


magnétique





on obtient le meilleur des deux mondes : système précis et sans dérive



gyro diverge

Estimation du biais du gyroscope

- On ajoute une entrée G_{biais} dans l'état : $X = \begin{bmatrix} \theta \\ \dot{\theta} \\ G_{biais} \end{bmatrix}$
- On le soustrait à la mesure du gyroscope :

$$\Phi = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} \theta \\ \dot{\theta} \\ G_{biais} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ G_{biais} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_{gyro} \end{bmatrix}$$

- On ajuste la vitesse qu'on estime que le biais change, via son « bruit » :
- Matrice de mesure : $\Lambda = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$

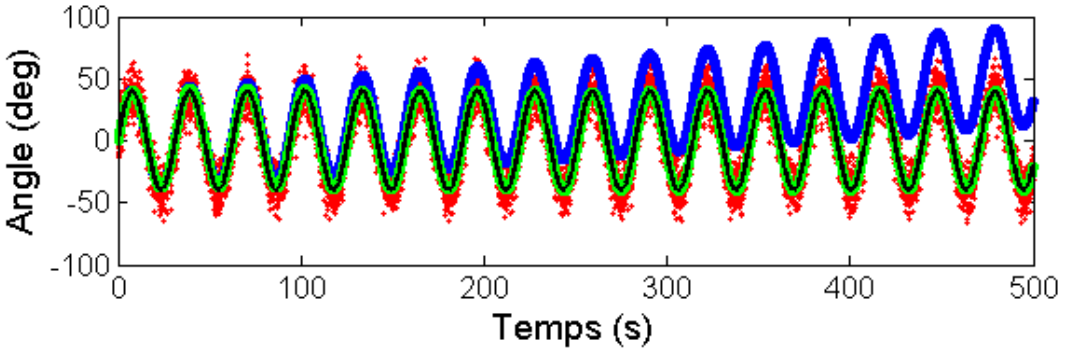
random walk

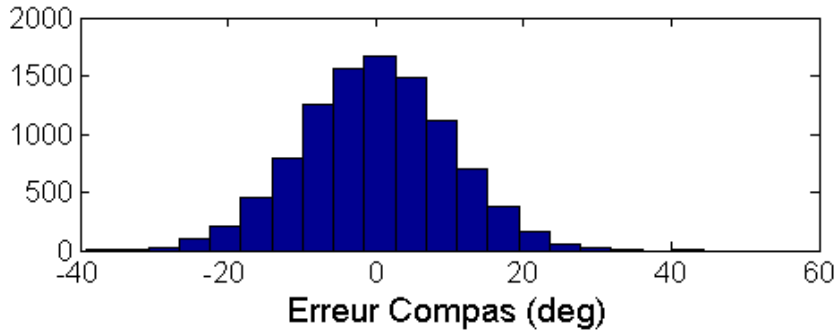
$$C_v = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sigma^2_{gyro} & 0 \\ 0 & 0 & 0.003^2 \end{bmatrix}$$

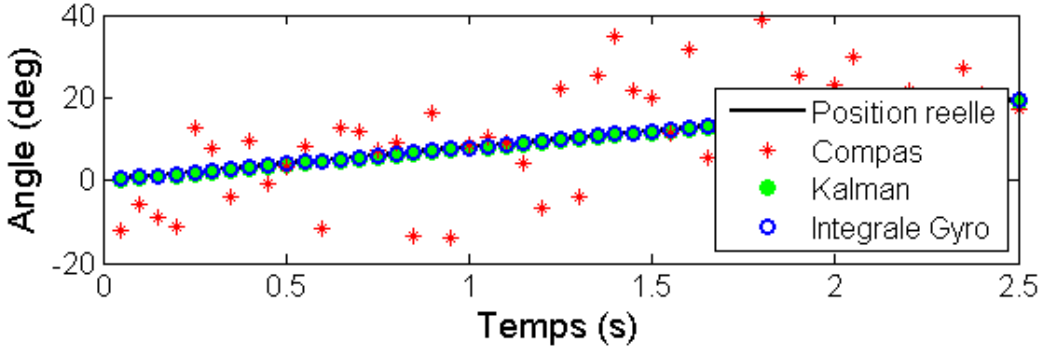
utilise maintenant la valeur exact du bruit de gyro

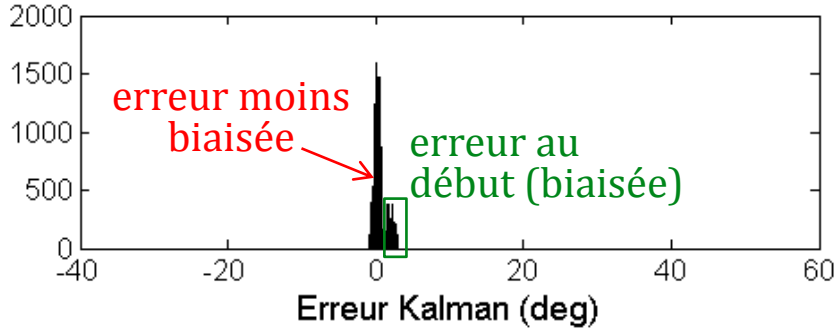
UNIVERSITÉ LAVAL

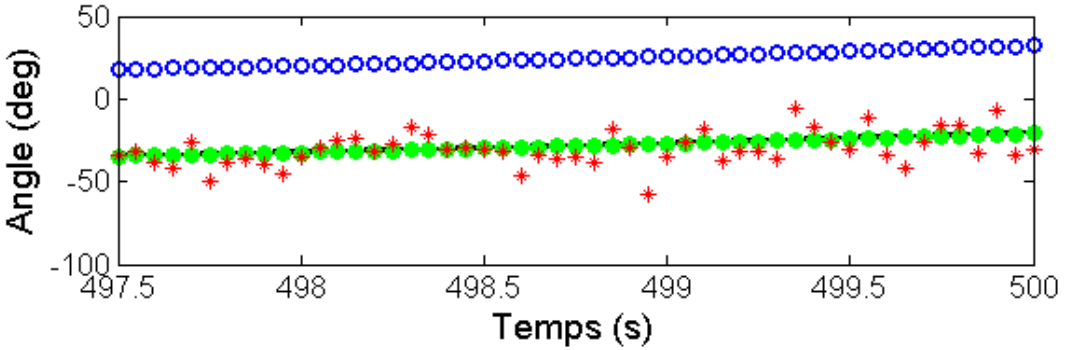
Estimation biais : résultats

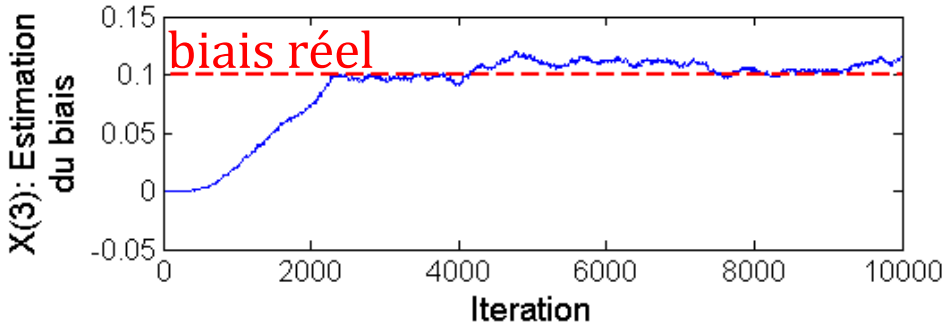












UNIVERSITÉ LAVAL

Note: seulement possible si capteurs extéroceptifs sont présents

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):

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

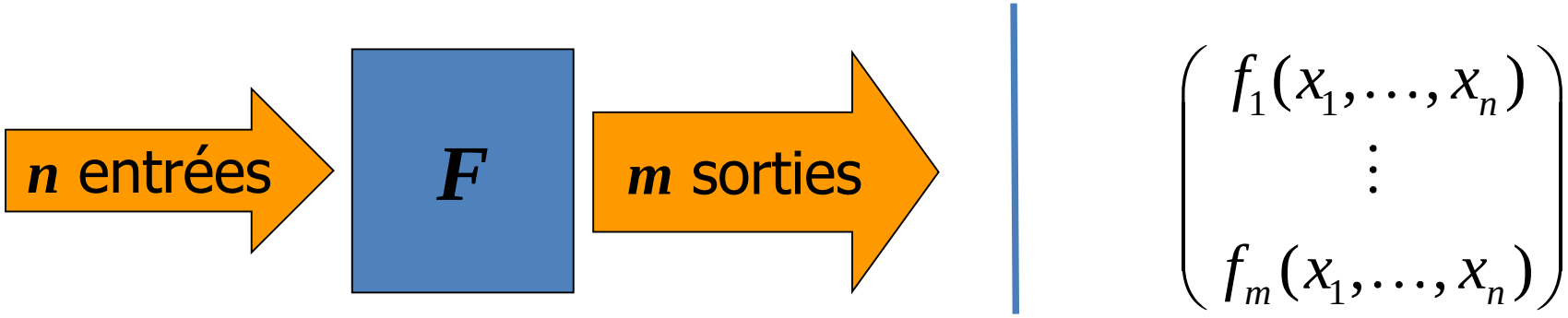
- Exemple ***f(l,m) = cos(l)ln(m)***
  - on linéarise autour de (a=*0.4 rad*, b=*30*)

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

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

Jacobienne : linéarisation généralisée

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



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

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

matrice de ***m*x*n***

Extension de la tangente pour fonctions multi-variables

Très utilisé en robotique!

source : wikipédia

Linéarisation avec Jacobienne

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

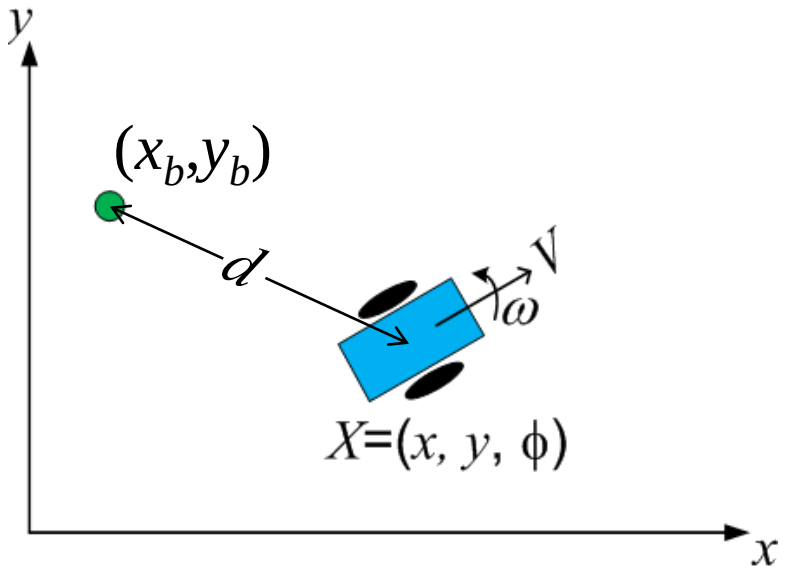
Exemple $$\vec{X} = \begin{bmatrix} x \\ \theta \\ \phi \end{bmatrix} \qquad f(\vec{X}) = \begin{bmatrix} f_1(\vec{X}) \\ f_2(\vec{X}) \end{bmatrix} = \begin{bmatrix} 3x\cos\theta\sin\phi \\ 7x + 14\ln\phi \end{bmatrix}$$

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

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

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 :

$$x_{t+1} = x_t + V\Delta t \cos\phi$$
$$y_{t+1} = y_t + V\Delta t \sin\phi$$
$$\phi_{t+1} = \phi_t + \omega\Delta t$$

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

(d est une mesure incomplète)

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

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

(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

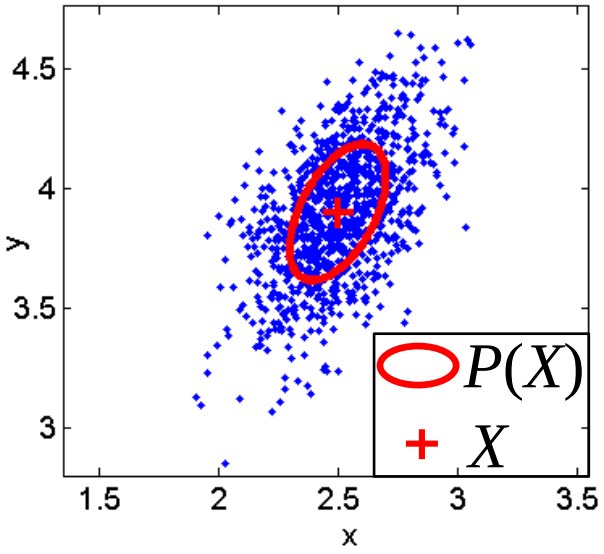
Jamais besoin d'inverser la fonction de capteur $h_z(\,)$!

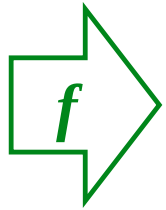
Exemple propagation $P(x)$ (sans bruit)

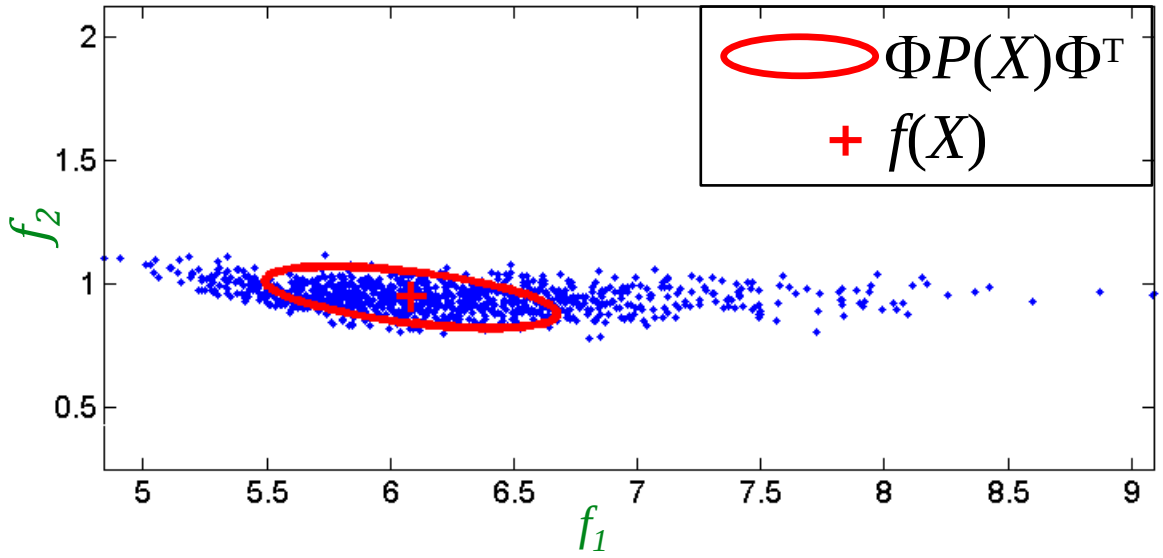
$$X = \begin{bmatrix} x = 2.5 \\ y = 3.9 \end{bmatrix}, \quad f_1(X) = 2\cos(x) + 3\sin(y) + xy \quad f_2(X) = e^{-x} + \frac{y}{x}, \quad P(X) = \begin{bmatrix} 0.04 & 0.03 \\ 0.03 & 0.08 \end{bmatrix}$$

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 jacobienne Φ.

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

- La correction à partir de la mesure $\boldsymbol{z}$ se fait avec la jacobienne Λ.

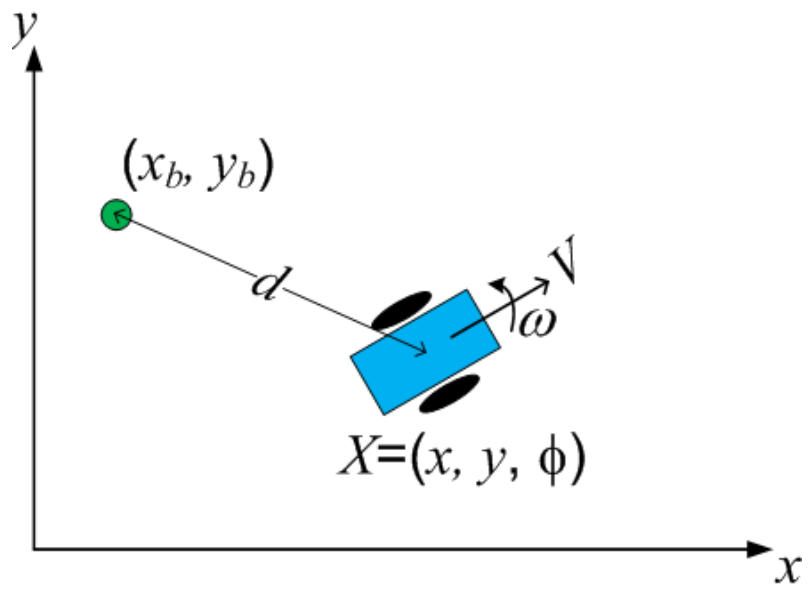
$$K(k+1) = P(k+1|k)\Lambda^T \{\Lambda P(k+1|k)\Lambda^T + C_w(k+1)\}^{-1}$$

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

Système 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 :

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

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

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

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

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

Kalman étendu : Jacobienne Γ

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

$$u = \begin{bmatrix} V & \omega \end{bmatrix}^T$$

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

$$x = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix} \qquad \Gamma = \begin{bmatrix} \frac{\partial f_x}{\partial V} & \frac{\partial f_x}{\partial \omega} \\ \frac{\partial f_y}{\partial V} & \frac{\partial f_y}{\partial \omega} \\ \frac{\partial f_\phi}{\partial V} & \frac{\partial f_\phi}{\partial \omega} \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} \Delta t\cos\phi & 0 \\ \Delta t\sin\phi & 0 \\ 0 & \Delta t \end{bmatrix}$$

Kalman étendu : Propagation

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

$$\hat{x}(k+1|k) = \begin{bmatrix} f_x \\ f_y \\ f_\phi \end{bmatrix}$$

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

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

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

- 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

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

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

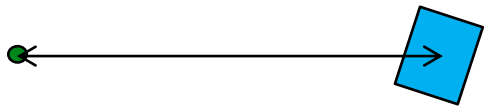
$$\Lambda = \begin{bmatrix} \frac{\partial h_z}{\partial x} & \frac{\partial h_z}{\partial y} & \frac{\partial h_z}{\partial \phi} \end{bmatrix} = \begin{bmatrix} \frac{x - x_b}{\sqrt{(x - x_b)^2 + (y - y_b)^2}} & \frac{y - y_b}{\sqrt{(x - x_b)^2 + (y - y_b)^2}} & 0 \end{bmatrix}$$

- Indique quelles variables d'état affectent directement la mesure $\boldsymbol{z}$

Jacobienne mesure Λ

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

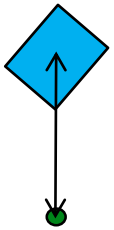
cas 1 : $y = y_b$



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

… 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

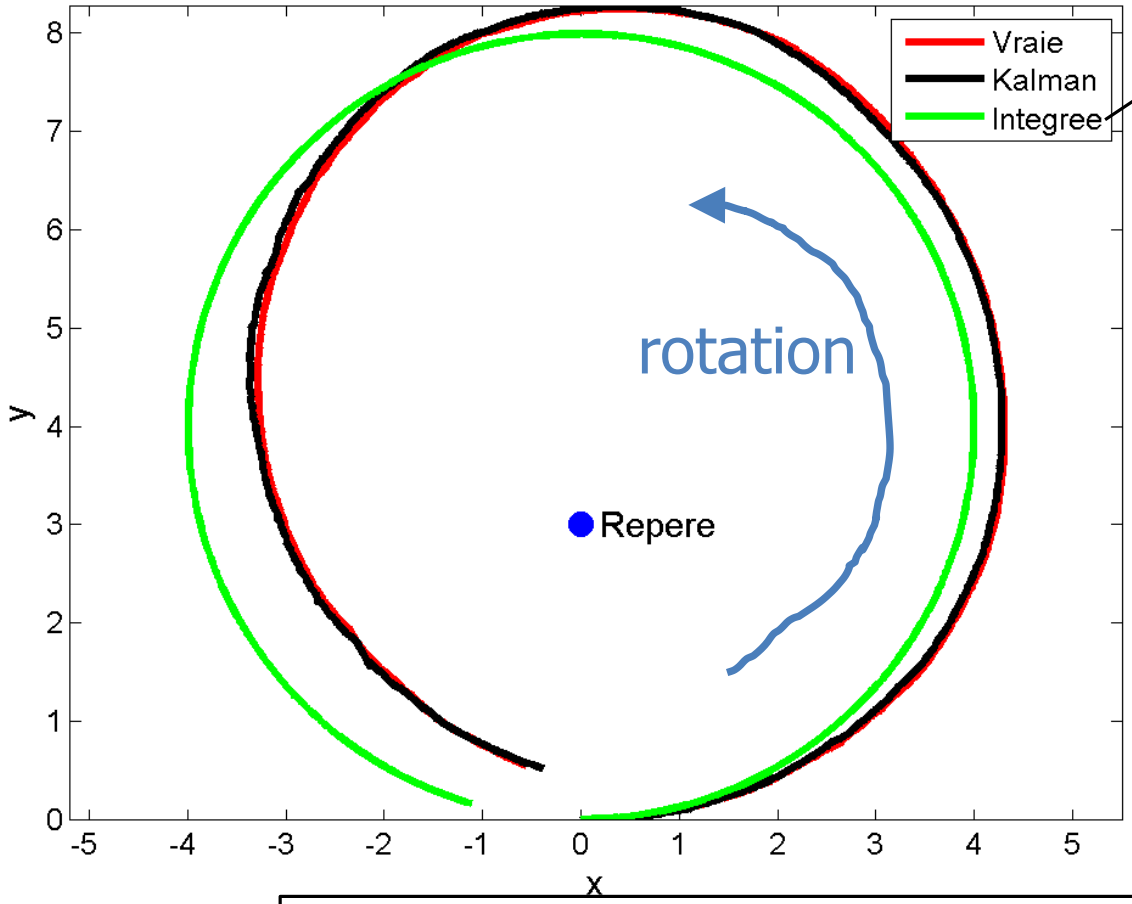
```
% ====== Propagation =========
U = [V*dT omega*dT]';
X(1) = X(1) + V*cos(X(3))*dT;
X(2) = X(2) + V*sin(X(3))*dT;
X(3) = X(3) + omega*dT;

% 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

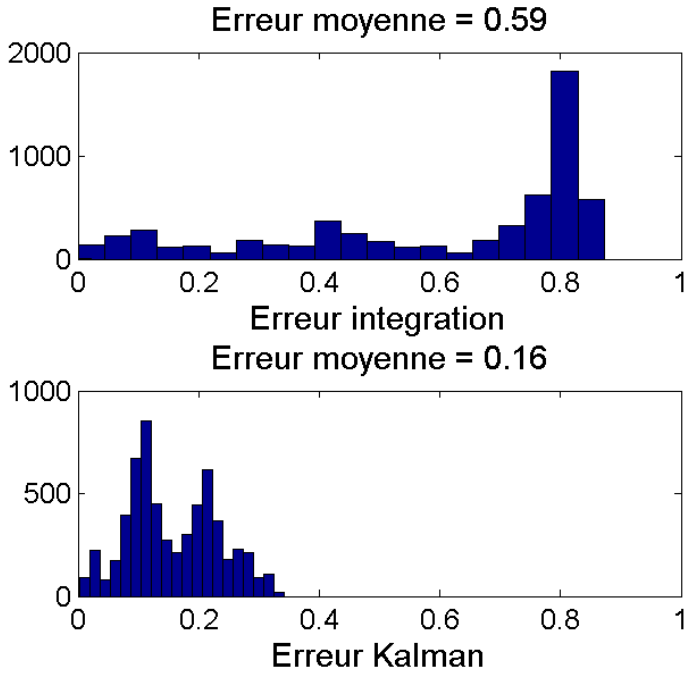
Λ

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

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 $[x\ y\ \theta]^{\mathrm{T}}$
- Propagation non-linéaire
- Mesure x, y linéaire
- Bruits de déplacement: $\sigma_V = 0.2$ m/s, $\sigma_\omega = 0.1$ rad/s
- Commande fixe $u = [10\ \text{m/s} \quad 0.1\ \text{rad/s}]^{\mathrm{T}}$: tourne en rond!
- Bruits de capteurs : $\sigma_x = \sigma_y = 12$ 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`

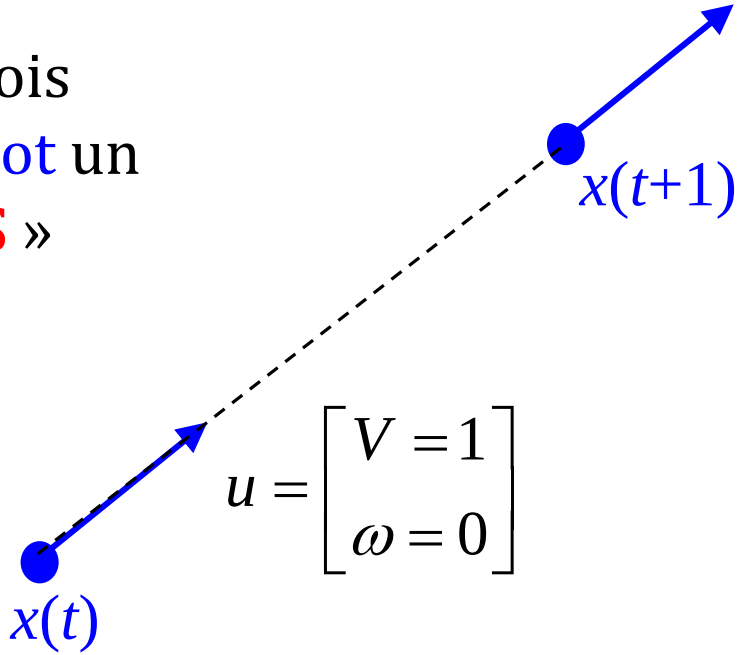
Bayes : réconcilier l'innovation

Qui est fautif :
- la position $x(t+1)$?
- l'orientation ?
- le GPS ?

■ GPS

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

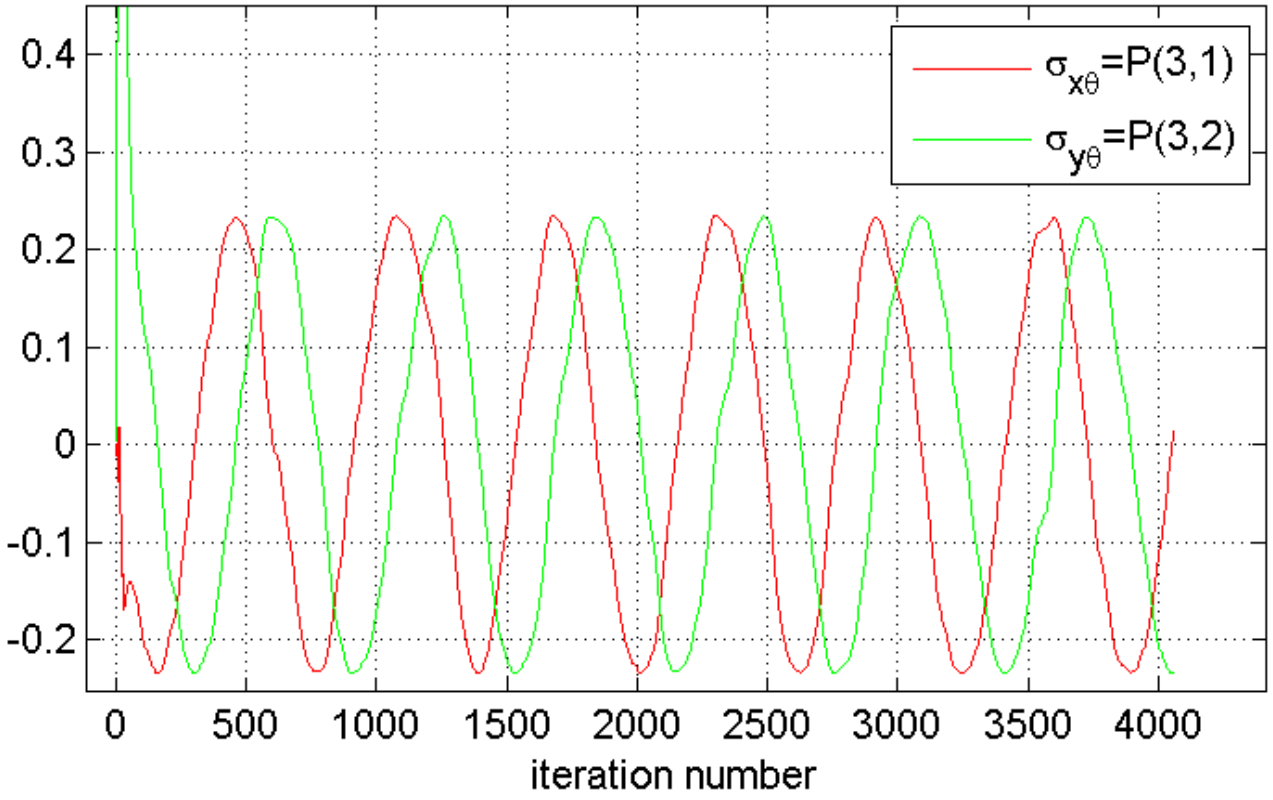
Kalman : « Je dois déplacer le robot un peu vers le GPS »



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

Termes de covariance dans *P*

- La matrice de covariance P contiendra les information sur comment corriger l'angle



L'oscillation arrive car on tourne en rond

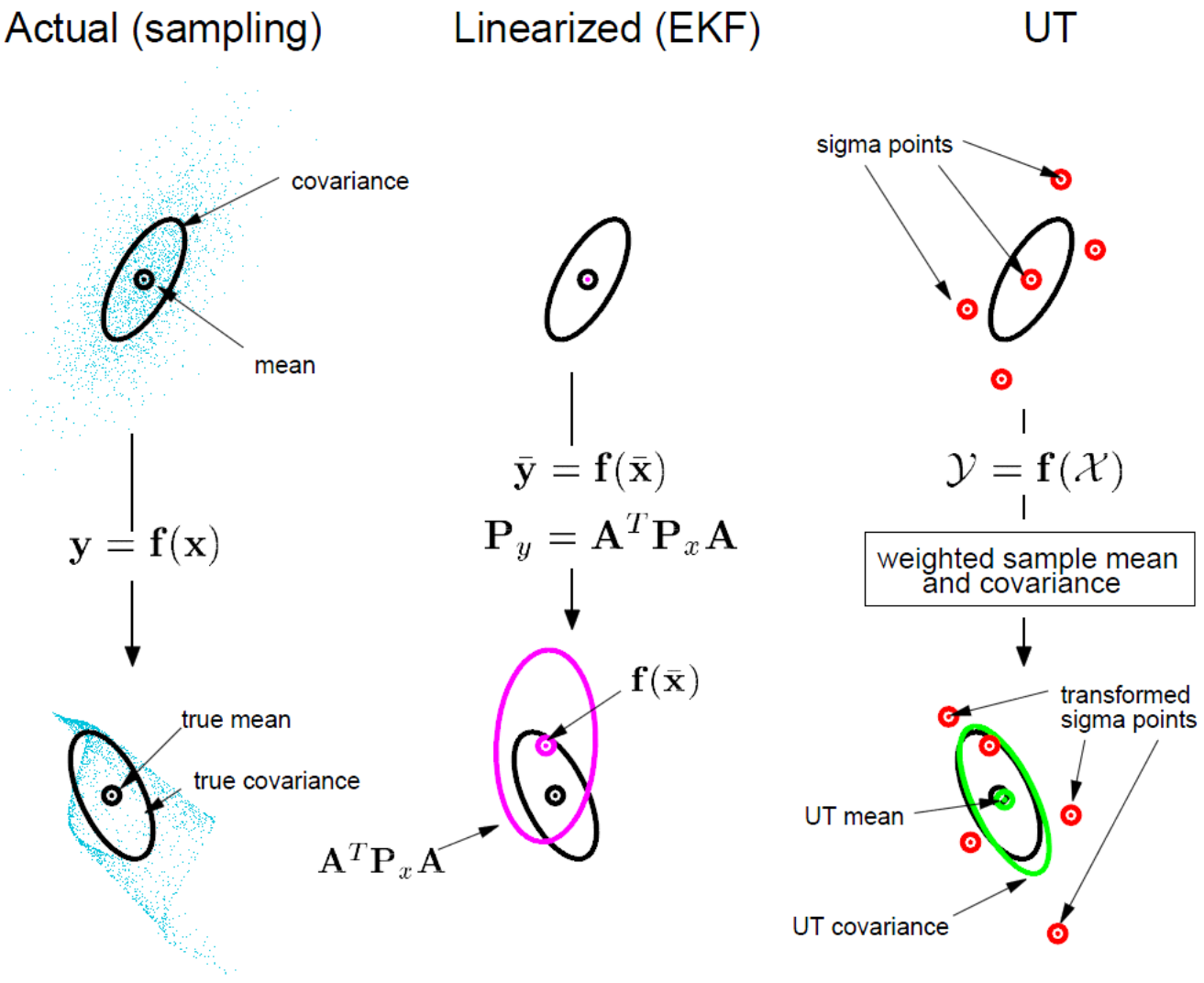
Filtre Kalman non-parfumé (unscented)

Fonctions fortement non-linéaires?

- EKF se base sur la linéarisation (série de Taylor) pour calculer la nouvelle matrice P, via la jacobienne Φ : $P(k+1|k) = \Phi P(k) \Phi^T + C_v$
- Si la fonction f est fortement non-linéaire, l'approximation ne sera pas bonne
- Nouvelle approche : utiliser des échantillons d'état X **choisis judicieusement** (***sigma points***), passé 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* $X^{[i]}$

- État à *n* dimensions : utiliser **2*n*+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 2*n* sigma points restants :

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

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

`sqrtm`

$i^{\text{ème}}$ colonne

UT: Poids *w* associés aux *sigma points*

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

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

- 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 $f(\,)$:

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

- Plus besoin de calculer les dérivés ☺

UT : exemple

- Reprise de l'exemple de linéarisation précédent :

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

- n = 2 dimensions, donne

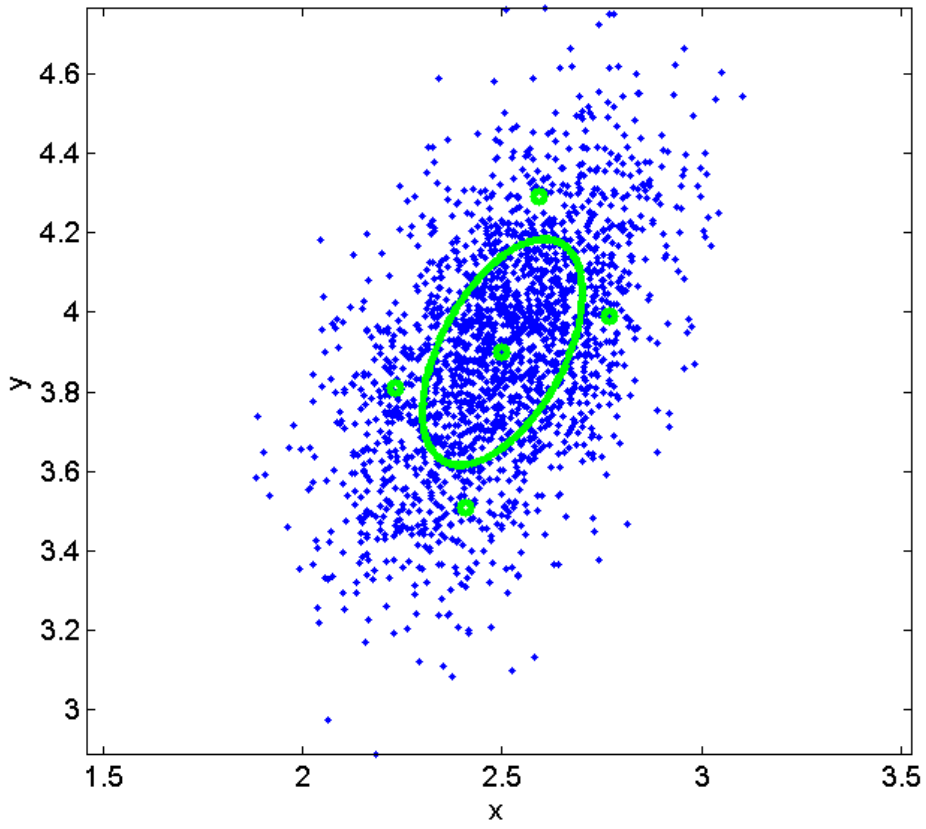
  $2n+1$= 5 sigma points

- Paramètres UT utilisés :

$$\alpha = 0.001, \beta = 2, \kappa = 1$$

$$(\mathrm{n}+\lambda)P = \begin{bmatrix} 0.08 & 0.06 \\ 0.06 & 0.16 \end{bmatrix}$$

$$\sqrt{(\mathrm{n}+\lambda)P} = \begin{bmatrix} 0.2677 & 0.0913 \\ 0.0913 & 0.3894 \end{bmatrix}$$

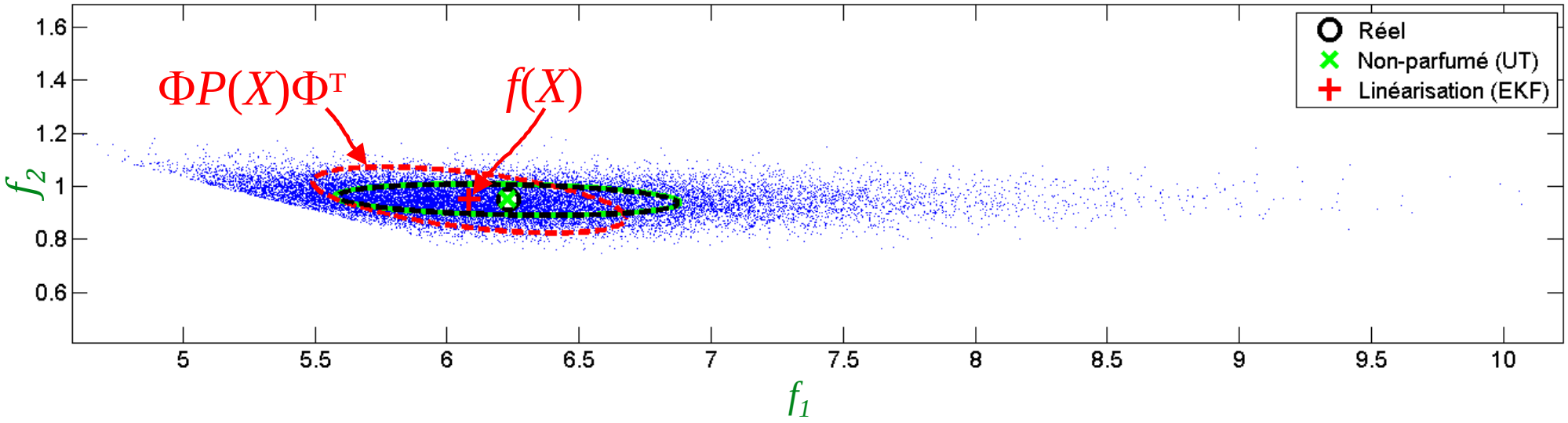


Points sigma dans matlab

```
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

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

passe les sigma points dans la fonction non-linéaire

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

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

calcule de la distribution après déplacement via $f(\ )$, 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

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

← 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

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

← EKF

$$K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$$

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 récusions
- 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