

# Filtre de Kalman

Sarah Curtit

## 1 Présentation du Filtre de Kalman

### 1.1 Le filtre de Kalman continu

#### 1.1.1 Le modèle

Le filtre de Kalman est un modèle d'état défini par deux équations :

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) + Mw(t) & \text{équation d'état} \\ y(t) = Cx(t) + Du(t) + v(t) & \text{équation de mesure} \end{cases}$$

$x$  est un vecteur d'état,  $y$  un vecteur de mesures,  $u$  un vecteur de commandes (données déterminées), tandis que les signaux  $w(k)$  et  $v(k)$  sont des bruits blancs gaussiens centrés de densité spectrale de puissance  $W$  et  $V$  respectivement. On a

$$\begin{aligned} E[w(k)w(k+l)^T] &= W\delta(l) \quad \delta(l) = 1 \text{ si } l=0; 0 \text{ sinon} \\ E[v(k)v(k+l)^T] &= V\delta(l) \quad (V \text{ doit être inversible}) \\ E[w(k)v(k+l)^T] &= 0 \end{aligned}$$

#### 1.1.2 Minimisation de l'erreur d'estimation

En pratique, le filtre doit retourner en sortie l'état estimé du système, noté  $\hat{x}$ . L'équation d'état du filtre nous est donnée par

$$\dot{\hat{x}}(t) = A_f\hat{x}(t) + B_fu(t) + K_fy(t)$$

où  $A_f$ ,  $B_f$  et  $K_f$  sont des matrices à déterminer.

Soit  $\varepsilon(t) = x(t) - \hat{x}(t)$  l'erreur d'estimation du système. On souhaite que cet estimateur soit non biaisé, c'est à dire que  $\lim_{t \rightarrow +\infty} \bar{\varepsilon} = 0$  quel que soit le profil de commande et quel que soit l'état initial ( $\bar{\varepsilon}$  étant l'espérance mathématique de  $\varepsilon$ ).

L'écriture de ces conditions nous donne finalement l'équation du filtre de Kalman

$$\dot{\hat{x}}(t) = (A\hat{x} + Bu) + K_f(y - C\hat{x} - Du)$$

$K_f$  est appelé le gain du filtre. C'est lui qui traduit la confiance que l'on a dans le modèle ou les mesures. Si on considère que le modèle est très fiable et les mesures très bruitées on accordera peu d'importance à celles-ci, et  $K_f$  sera petit. Si l'on fait au contraire plus confiance aux mesures qu'au modèle,  $K_f$  sera grand.

Afin d'obtenir le filtre le plus fiable possible, le gain  $K_f$  doit minimiser la variance de l'erreur d'estimation, c'est à dire la trace de la matrice de covariance de l'erreur d'estimation,

$$P(t) = E[(x(t) - \hat{x}(t))(x(t) - \hat{x}(t))^T]$$

$P(t)$  vérifiant l'équation

$$\dot{P}(t) = (A - K_f C)P(t) + P(t)(A - K_f C)^T + MWM^T + K_f V K_f^T$$

En minimisant  $P$  on obtient

$$\begin{aligned} K_f &= P(t)C^T V^{-1} \\ \dot{P}(t) &= AP(t) + P(t)A^T - P(t)C^T V^{-1} C P(t) + MWM^T \end{aligned}$$

Cette deuxième équation étant appelée équation différentielle de Riccati.

Une fois les erreurs d'initialisation corrigées (régime constant), on obtient

$$\begin{aligned} K_f &= P(t)C^T V^{-1} && \text{Désormais constant.} \\ 0 &= AP(t) + P(t)A^T - P(t)C^T V^{-1} C P(t) + MWM^T && P \text{ étant la solution positive de l'équation.} \end{aligned}$$

## 1.2 Le filtre de Kalman discret

### 1.2.1 Le modèle

Dans les faits, les appareils numériques fonctionnent de manière discrète. On utilisera donc en pratique le modèle de Kalman discret

$$\begin{cases} x(k+1) = A_d x(k) + B_d u(k) + M_d w_d(k) & \text{équation d'état} & x \in \mathbb{R}^n, y \in \mathbb{R}^m, w_d \in \mathbb{R}^q \\ y(k) = C_d x(k) + D u(k) + v_d(k) & \text{équation de mesure} & y \in \mathbb{R}^m, v_d \in \mathbb{R}^p \end{cases}$$

Les signaux  $w_d$  et  $v_d$  sont cette fois associés à des matrices de covariance  $W_d$  et  $V_d$  telles que

$$\begin{aligned} E[w_d(k)w_d(k+l)^T] &= W_d \delta(l) \\ E[v_d(k)v_d(k+l)^T] &= V_d \delta(l) \quad (V_d \text{ doit être inversible}) \\ E[w_d(k)v_d(k+l)^T] &= 0 \end{aligned}$$

On note  $dt$  le pas de temps associé au système discret.

### 1.2.2 Passage du modèle continu au modèle discret

Le modèle de Kalman discret est très proche du modèle continu, et on peut obtenir les nouvelles matrices discrètes à partir de celles du modèle continu. On a

$$A_d = e^{A dt} \quad B_d = \int_0^{dt} e^{A v} B dv \quad M_d = I_n \quad C_d = C \quad V_d = \frac{V}{dt}$$

L'expression de  $W_d$  est plus compliquée puisqu'on a  $W_d = \int_0^{dt} e^{A v} M W M^T e^{A^T v} dv$ . Cependant dans l'hypothèse où  $dt$  est petit par rapport au temps de réponse du système, on peut écrire

$$W_d \approx dt M W M^T$$

### 1.2.3 Représentation d'état, implémentation du filtre en pratique

Comme dans le modèle continu, on souhaite optimiser le gain  $K_f$  afin de minimiser l'erreur d'estimation.

Dans la suite de cette section, on distinguera

l'état prédit à l'instant $k+1$	$\hat{x}(k+1 k)$
connaissant toutes les mesures jusqu'à l'instant $k$	
l'état estimé connaissant la mesure à l'instant $k+1$ (après recalage)	$\hat{x}(k+1 k+1)$

L'état prédit est calculé de façon déterministe

$$\hat{x}(k+1|k) = A_d \hat{x}(k|k) + B_d u(k)$$

Tandis que l'équation d'état du filtre

$$\hat{\hat{x}}(k|k) = A_d \hat{x}(k|k) + B_d u(k) + K_f(k+1)(y(k) - C_d \hat{x}(k|k) - D \hat{x}(k|k))$$

nous donne

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K_f(k+1)(y(k) - C_d \hat{x}(k|k) - D \hat{x}(k|k))$$

La combinaison de ces deux équations nous permet d'obtenir la représentation d'état du filtre de Kalman permanent :

$$\boxed{\begin{cases} \hat{x}(k|k) &= (I - K_f C_d) \hat{x}(k|k-1) + (K_f \quad -K_f D) \begin{pmatrix} y(k) \\ u(k) \end{pmatrix} \\ \hat{x}(k+1|k) &= A_d(I - K_f C_d) \hat{x}(k|k-1) + (A_d K_f \quad B_d - A_d K_f D) \begin{pmatrix} y(k) \\ u(k) \end{pmatrix} \end{cases}}$$

C'est la deuxième équation, calculant l'état prédit à l'instant  $k+1$  qui nous intéresse et qui sera implémenté dans l'algorithme du filtre de Kalman.

Afin de totalement déterminer cette équation, il reste cependant à déterminer  $K_f$ , choisi encore une fois afin de minimiser l'erreur d'estimation  $P(k+1|k+1)$  (dont l'expression est extrêmement compliquée).

La résolution de l'équation différentielle nous donne

$$\begin{aligned} K_f(k+1) &= P(k+1|k) C_d^T (C_d P(k+1|k) C_d^T + V_d)^{-1} \\ P(k+1|k) &= A_d P(k|k-1) A_d^T - A_d P(k|k-1) C_d^T (C_d P(k|k-1) C_d^T + V_d)^{-1} C_d P(k|k-1) A_d^T \\ &\quad + M_d W_d M_d^T \end{aligned}$$

En régime permanent la matrice de covariance de l'erreur de prédiction est constante et vérifie l'équation discrète de Riccati

$$P_p = A_d P_p A_d^T - A_d P_p C_d^T (C_d P_p C_d^T + V_d)^{-1} C_d P_p A_d^T + M_d W_d M_d^T$$

On peut en déduire le gain du filtre et la matrice de covariance de l'erreur d'estimation

$$\begin{aligned} K_f &= P_p C_d^T (C_d P_p C_d^T + V_d)^{-1} \\ P_e &= (I - K_f C_d) P_p \end{aligned}$$

La plupart des solveurs actuels proposent des fonctions permettant de résoudre l'équation de Riccati. Dans le cadre de ce projet, nous avons travaillé avec Python et la fonction `scipy.linalg.solve_discrete_are`.

## 2 Application au problème

### 2.1 Problème 1D

Pour simplifier le problème, considérons tout d'abord que l'outil utilisé est parfaitement perpendiculaire à la surface au point de contact et qu'il n'y a pas de frottement. Dans ce cas, on peut se permettre de considérer seulement les données selon  $z$  (repère orienté selon la normale à la surface au point courant).

On prend alors comme vecteurs d'état, de mesure et de commande respectivement

$$x = \begin{pmatrix} z & \dot{z} \end{pmatrix}, \quad y = (F_z), \quad u = \begin{pmatrix} \dot{z}_{robot} \\ F_{obj} \end{pmatrix}$$

ce qui simplifie grandement le problème

#### 2.1.1 Détermination des données du problème dans le cas continu

$v$  et  $w$  sont ici respectivement le vecteur des signaux aléatoires qui polluent les mesures  $y$  et le vecteur des signaux inconnus qui viennent perturber directement l'équation d'état du système. Ici  $v$  représente l'imprécision du capteur et  $w$  rend compte des aspérités de la surface et de l'imprécision du robot.

La matrice  $M$  relie le bruit d'état  $w_x$  à  $w$  par la relation  $w_x = Mw$

En faisant pour l'instant abstraction des vecteurs d'erreurs (et donc du dernier terme de chaque égalité) le système s'écrit dans notre cas comme

$$\begin{cases} \begin{pmatrix} \dot{z} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} z \\ \dot{z} \end{pmatrix} \\ (F_z) = (-k \quad 0) \cdot \begin{pmatrix} z \\ \dot{z} \end{pmatrix} + (0 \quad 1) \begin{pmatrix} \dot{z}_{robot} \\ F_{obj} \end{pmatrix} \end{cases}$$

Le choix a été fait ici de considérer l'accélération comme nulle car le déplacement du bras du robot est commandé en fonction de la force exercée sur le ressort, il n'en dépend pas directement.

La matrice  $C$  est elle déterminée à partir d'une simple loi de Hooke :  $F_z = -k * d_z$ .

Déterminons désormais les matrices associées vecteurs d'erreur  $v$  et  $w$ .

on note

$$\begin{aligned} v &= \begin{pmatrix} v_{F_z} \end{pmatrix}^T & \text{où } v_{F_z} \text{ représente l'incertitude sur la mesure } F_z \text{ du capteur} \\ w &= \begin{pmatrix} w_z \end{pmatrix} & \text{où } w_z \text{ représente l'incertitude de position du robot selon } z \end{aligned}$$

Pour ce qui est du vecteur  $v$  son unique composante comprend en fait 3 termes associés respectivement

à l'incertitude de mesure du capteur  
à l'incertitude sur les aspérités de la surface  
à l'incertitude sur la position du robot

Bien que ces deux dernières contributions soient intuitivement perçues comme des bruits d'état, elle ne viennent en fait pas perturber directement la vitesse ou l'accélération du robot, perçus comme infiniment rigide devant le poussoir à bille. Elle viennent cependant comprimer plus ou moins le ressort du poussoir et donc modifier les données reçues par le capteur.

$w(t)$  et  $v(t)$  sont des vecteurs gaussiens centrés de densité spectrale de puissance  $W$  et  $V$  respectivement. On a :

$$\begin{aligned} E[w(t)w(t+\tau)^T] &= W\delta(\tau) \\ E[v(t)v(t+\tau)^T] &= V\delta(\tau) \\ E[w(t)v(t+\tau)^T] &= 0 \end{aligned}$$

$V$  peut-être calculé à partir des données d'échantillonnage du capteur. Le certificat d'étalonnage nous indique en effet les précisions suivantes :

$\delta F_x$	$\delta F_y$	$\delta F_z$	$\delta M_x$	$\delta M_y$	$\delta M_z$
1.00%	1.25%	0.75%	1.00%	1.25%	1.50%

On estime également les ordres de grandeur des forces et moments qui seront rencontrés par le capteur (avec les hypothèses émises précédemment)

$\bar{F}_x$	$\bar{F}_y$	$\bar{F}_z$	$\bar{M}_x$	$\bar{M}_y$	$\bar{M}_z$
0N	0N	20N	0N.m	0N.m	0N.m

La précision du robot et des aspérités de la surface sont

$\delta z_{\text{robot}}$	$\delta z_{\text{asp}}$
0.02mm	1mm

L'ordre de grandeur des aspérités représente ici la différence de hauteur de surface non prévue maximale qui puisse être rencontrée entre deux pas de temps.

$V$  est la matrice de densité spectrale de puissance du vecteur  $v$ . Chaque source d'erreur étant indépendante, cela signifie que la matrice est diagonale, et on a

$$\begin{aligned} V_{ii} &= \mathcal{L}[\phi_{ii}(\tau)] \\ &= \int_{-\infty}^{+\infty} E[v_i(t)v_i(t+\tau)^t] d\tau \end{aligned}$$

Le vecteur  $v$  étant un bruit blanc gaussien centré, les choses se simplifient. En effet on a  $\forall v_i$

$$\phi(t, \tau)_{ii} = \sigma_i^2 \delta(0)$$

et donc

$$V_{ii} = \sigma_i^2 = Var(v_i)$$

Il ne reste plus qu'à calculer les écarts types associés à chaque erreur. L'intervalle de confiance à 95% étant l'intervalle  $[m - 1.96\sigma, m + 1.96\sigma]$  ( $m$  étant la moyenne) on a

$$\begin{aligned} [-\delta F_z * \bar{F}_z ; \delta F_z * \bar{F}_z] &= [-1.96 * \sigma_{F_z} ; 1.96 * \sigma_{F_z}] \\ [-\delta z_{asp} ; \delta z_{asp}] &= [-1.96 * \sigma_{z_{asp}} ; 1.96 * \sigma_{z_{asp}}] \\ [-\delta z_{robot} ; \delta z_{robot}] &= [-1.96 * \sigma_{z_{robot}} ; 1.96 * \sigma_{z_{robot}}] \end{aligned}$$

Soit

$$\begin{aligned} \sigma_{F_z} &= \frac{\delta F_z * \bar{F}_z}{1.96} \\ \sigma_{z_{asp}} &= \frac{\delta z_{asp}}{1.96} \\ \sigma_{z_{robot}} &= \frac{\delta z_{robot}}{1.96} \end{aligned}$$

On en déduit

$$\begin{aligned} V &= (\sigma_{F_z}^2 + \sigma_{z_{asp}}^2 + \sigma_{z_{robot}}^2) \\ &= \left( \frac{(\delta F_z * \bar{F}_z)^2 + \delta z_{asp}^2 + \delta z_{robot}^2}{1.96^2} \right) \\ &= (5.857 * 10^{-3}) \end{aligned}$$

On remarque ici que la contribution de l'imprécision du robot à la matrice  $V$  est négligeable (de l'ordre de  $10^{-8}$ ) devant l'imprécision du capteur. Le robot est en effet extrêmement précis. La contribution des aspérités de surfaces est également ici négligeable, mais il serait envisageable d'augmenter l'ordre de grandeur de ces aspérités si notre système est efficace.

$W$  est ensuite calculé de la même façon, en considérant que  $\delta \dot{z}_{robot} = \delta z_{robot} = 0.02mm$

$$\begin{aligned} [-\delta \dot{z}_{robot} ; \delta \dot{z}_{robot}] &= [-1.96 * \sigma_{\dot{z}_{robot}} ; 1.96 * \sigma_{\dot{z}_{robot}}] \\ \sigma_{\dot{z}_{robot}} &= \frac{\delta \dot{z}_{robot}}{1.96} \\ W &= (\sigma_{\dot{z}_{robot}}^2) \\ &= \left( \left( \frac{\delta \dot{z}_{robot}}{1.96} \right)^2 \right) \\ &= (1.041 * 10^{-8}) \end{aligned}$$

Déterminons enfin la matrice  $M$  qui permet de "relier" le vecteur  $w$  à l'équation d'état. Elle est ici triviale puisqu'on a tout simplement

$$M = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Toutes les données de l'équation ayant été définies, on peut désormais calculer les matrices associées au modèle discret

$$\begin{aligned}
 A_d &= e^{Adt} \\
 &= e^{\begin{pmatrix} 0 & dt \\ 0 & 0 \end{pmatrix}} \\
 &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & dt \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} + \dots \\
 &= \begin{pmatrix} 1 & dt \\ 0 & 1 \end{pmatrix}
 \end{aligned}
 \qquad
 \begin{aligned}
 B_d &= \int_0^{dt} e^{Av} B dv \\
 &= 0
 \end{aligned}
 \qquad
 \begin{aligned}
 W_d &= dt M W M^T \\
 &= \begin{pmatrix} \left(\frac{\delta \dot{z}_{robot}}{1.96}\right)^2 * dt & 0 \\ 0 & 0 \end{pmatrix}
 \end{aligned}$$

$$\begin{aligned}
 C_d &= C \\
 &= \begin{pmatrix} -k & 0 \end{pmatrix}
 \end{aligned}
 \qquad
 \begin{aligned}
 V_d &= \frac{V}{dt} \\
 &= \begin{pmatrix} \frac{(\delta F_z * \bar{F}_z)^2 + \delta z_{asp}^2 + \delta z_{robot}^2}{1.96^2 * dt} \end{pmatrix}
 \end{aligned}$$

Et on peut calculer la matrice d'erreur P à partir de l'équation de Ricatti discrète.

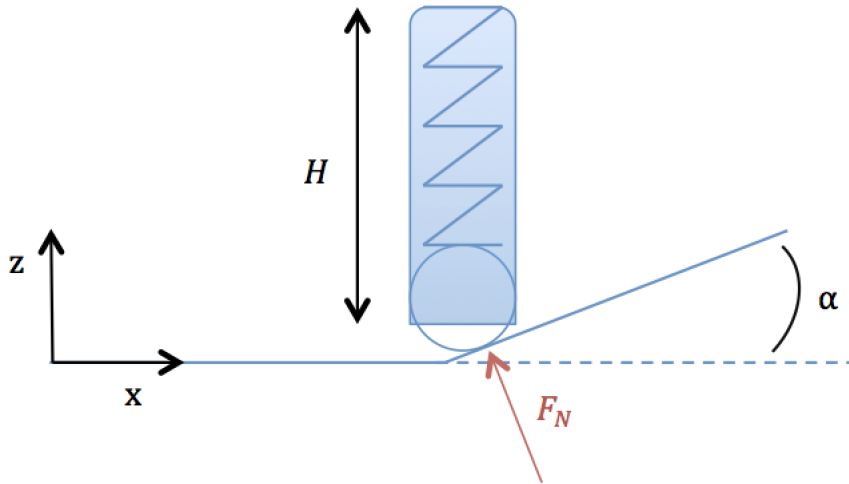
$$P_p = A_d P_p A_d^T - A_d P_p C_d^T (C_d P_p C_d^T + V_d)^{-1} C_d P_p A_d^T + M_d W_d M_d^T$$

Une résolution numérique nous donne

$$P_p = \begin{pmatrix} 2.524 * 10^{-9} & 9.032 * 10^{-25} \\ 9.032 * 10^{-25} & 0 \end{pmatrix}$$

## 2.2 Modèle 2D

Afin d'être plus réalistes, considérons désormais des forces selon les axes x et z, ainsi que le moment autour de l'axe y. Nous négligerons cependant toujours les frottements.



Plaçons nous dans le cas où l'outil n'est plus perpendiculaire à la surface (voir schéma ci-dessus)

Etant donné qu'il n'est pas possible de changer l'orientation du robot XXX -> CARO, nous nous bornerons toujours à corriger la trajectoire du robot.

Les valeurs mesurées par le capteur dépendent désormais de l'angle  $\alpha$  formé avec l'horizontale. On a

$$\begin{aligned} F_x &= F_N \sin(\alpha) \\ F_z &= F_N \cos(\alpha) \\ M_y &= F_x * H \quad \text{en faisant l'hypothèse que la hauteur découverte de la bille} \\ & \quad \text{est négligeable devant la hauteur du poussoir} \\ &= F_N \sin(\alpha) * H \end{aligned}$$

L'objectif force devant désormais être mesuré selon la nouvelle normale à la surface, il convient de corriger la matrice D de l'équation de mesure. On détermine tout d'abord  $\alpha$  grâce aux données du capteur :

$$\alpha = \arctan\left(\frac{F_x}{F_z}\right)$$

On a

$$F_z = -k\delta z + F_{obj}\cos(\alpha)$$

soit

$$D = \begin{pmatrix} 0 & \cos(\alpha) \end{pmatrix}$$

Afin de tenir compte de la pente imprévue, il faut également ajuster le déplacement et la vitesse selon z.

Pour cela, on prend désormais en compte un vecteur commande u correspondant à la vitesse du robot. Afin de suivre la pente, on souhaite désormais imposer une vitesse verticale supplémentaire égale à  $\sin \alpha * \dot{z}_{robot}$ .

On pose donc

$$\begin{aligned} u &= \begin{pmatrix} \dot{z}_{robot} \\ F_{obj} \end{pmatrix} \\ B &= \begin{pmatrix} \sin(\alpha) & 0 \\ 0 & 0 \end{pmatrix} \end{aligned}$$

Les matrices V et W sont modifiées car une nouvelle source d'imprécision est apporté par l'introduction du paramètre alpha.

## 2.3 Modèle 2D avec prise en compte du frottement