Estimation d'Etat par Filtre de Kalman
Travaux Pratiques

A INTRODUCTION

Ce TP a pour objectif de présenter une méthode de mise en œuvre d'un filtre de Kalman et d'un filtre de Kalman étendu sous Matlab. Pour cela, on considérera le problème ci-dessous.

On souhaite estimer l'état du système modélisé par la représentation *d'état non linéaire* à temps continu suivante

$$\begin{cases} \dot{\underline{x}}(t) = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -x_2\, u + \beta \\ -\alpha\, x_2^2 + u\, x_2 \end{bmatrix} + \zeta(t) \\ y(t) = C\,\underline{x} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \omega(t) \end{cases} \qquad \textbf{(1)}$$

où $\underline{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$ est un vecteur d'état, y la mesure, u la commande, $\alpha = 4$ et $\beta = 1$.

B OBJECTIFS ET DIFFERENTS FICHIERS

L'objectif de ce travail est d'estimer l'état du système considéré et d'évaluer l'estimateur mis en œuvre en simulation.

Dans un premier temps, on linéarisera le système autour d'un point d'équilibre et on mettra en œuvre un filtre de Kalman. Dans un second temps, afin d'obtenir un meilleur estimé de l'état, on mettra en œuvre un filtre de Kalman étendu en se basant sur les équations d'état non linéaires.
Puis, on évaluera les performances de ce filtre de Kalman étendu à l'aide de simulations de Monte-Carlo.

- ***Simulation du système réel***

Deux fichiers *simulink* différents peuvent être utilisés pour simuler le comportement du système réel :

- *SimNL.mdl* : simulation du système non linéaire
- *SimLin.mdl* : simulation du système linéarisé.

Ces fichiers utilisent des paramètres définis dans le fichier *ParametresReels.m*. **Donc avant, de lancer une simulation, exécutez *ParametresReels*.**

Ces fichiers renvoient les valeurs de la commande, de l'état réel et de la sortie non bruitée respectivement dans les variables *UReel*, *XReel* et *YReel*. Ces grandeurs sont stockées à la période *ParamReels.Te*.

- ***Filtrage***

La simulation du système réel et le filtrage seront effectués à l'aide du même fichier. On considèrera trois types de filtrage effectués à l'aide des fichiers :

- *LanceSimFK.m* : filtrage à l'aide d'un filtre de Kalman, *c.-à-d.* en considérant le modèle linéarisé.

- *LanceSimFKE.m* : filtrage à l'aide d'un filtre de Kalman étendu.

- *LanceMonteCarloFKE.m* : évaluation du filtre de Kalman étendu par des simulations de Monte-Carlo.

Afin de faciliter l'implémentation et le **"debuggage",** il est préférable de simuler le système réel jusqu'au temps final choisi, puis d'effectuer le filtrage. La simulation du système réel est effectuée en lançant l'exécution des fichiers simulink mentionnés précédemment, à l'aide de la commande ***sim*.**

Les paramètres du système réel sont stockés dans la structure *ParamReels* et ceux utilisés pour le filtrage sont stockés dans la structure *ParamFilt*. Ceci permet éventuellement de tester la robustesse du filtre en choisissant des paramètres différents pour simuler le système réel et pour le filtrage.

- ***Autres fichiers***

  - *ParametresReels.m* : définition des paramètres réels. Ils sont stockés dans la structure *ParamReels*.

  - *Linearise.m* : calcul des matrices *A* et *B* du système linéarisé. Il s'agit d'une fonction qui s'utilise de la façon suivante :

```
[A,B]=Linearise(X2e,Ue,Param);
```

    où

    `A et B`: matrices du système linéarisé
    `X2e et Ue` : valeurs de `X2 et U` autour desquelles on linéarise
    Param : Structures contenant les paramètres (ParamReels pour le système réel ou ParamFilt pour le filtrage)

  - *IntegEtat.m* : intégration numérique de l'équation d'état **(1)**. Il s'agit d'une fonction qui s'utilise de la façon suivante :

```
[Xkplus]=IntegEtat(Xk,U,Param,Te);
```

    où

    Xk : valeur de l'état (ou de son estimé) à l'instant *k Te*
    U : valeur de la commande *U* à l'instant *kTe*
    Param : Paramètres *(ParamFilt ou ParamRéels)*
    ParamReels.Te : Période d'échantillonnage *Te*
    Xkplus : valeur de l'état (ou de sa prédiction) à l'instant *(k+1) Te*

C MISE EN ŒUVRE D'UN FILTRE DE KALMAN

Il s'agit de mettre en œuvre un Filtre de Kalman en se basant sur le modèle linéarisé.

C.1 Linéarisation :

Travail :

- *Déterminer un état d'équilibre* $\underline{x}_e = \begin{bmatrix} x_{1e} & x_{2e} \end{bmatrix}^T$ *pour une entrée* $\underline{u}_e$.
- *Linéariser le système* **(1)** *autour du point d'équilibre déterminé à la question précédente de façon à obtenir un système linéarisé de la forme :*

$$\begin{cases} \dot{\underline{x}}_l = A\underline{x}_l + Bu_l + \zeta(t) \\ y_l = C\underline{x}_l + \omega(t) \end{cases} \qquad \textbf{(2)}$$

avec $\underline{x}_l = \underline{x} - \underline{x}_e$, $\underline{u}_l = \underline{u} - \underline{u}_e$ *et* $y_l = y - x_{1e}$.

On effectuera un développement limité de la fonction autour du point d'équilibre.

Linéarisation (Rappels) *: Soit une fonction non linéaire* $f(\varepsilon,\theta)$ *ayant pour point d'équilibre* (ε_0,θ_0) *. Pour de petites variations autour de ce point, i.e.,*

$$\varepsilon = \varepsilon_0 + \delta\varepsilon$$
$$\theta = \theta_0 + \delta\theta$$

le développement limité est donné par :

$$f(\varepsilon,\theta) = f(\varepsilon_0,\theta_0) + \left.\frac{\partial f}{\partial \varepsilon}\right|_{\varepsilon_0,\theta_0} (\varepsilon - \varepsilon_0) + \left.\frac{\partial f}{\partial \theta}\right|_{\varepsilon_0,\theta_0} (\theta - \theta_0) + t.o.s.$$

- *Compléter les fichiers* ***parametresReels.m*** *et* ***Linearise.m*** *de façon à y calculer les matrices A et B.*

C.2 Simulation du système linéarisé

Les paramètres du système réel ont été définis dans le fichier ***parametresReels.m***. Dans ce fichier, le calcul des matrices *A* et *B* est effectué à l'aide de la fonction *Linearise*. Le code de cette fonction est contenu dans le fichier ***Linearise.m*****.**

TRAVAIL :

- Examiner le fichier ***parametresReels.m*****,** puis l'exécuter.
- Simuler le système linéarisé **(2)** sous Simulink. Cette simulation sera effectuée en complétant le fichier que l'on appellera ***SimLin.mdl*****.** Pour cette simulation, on prendra une commande de type échelon :

$$u(t) = \begin{cases} u_e & si\ \ t < 4s \\ u_e + 1 & si\ \ t \geq 4s \end{cases}$$

Les bruits seront rajoutés dans le fichier Matlab.

- Compléter le fichier ***LanceSimFK.m*** de façon à mettre en œuvre un filtre de Kalman.

- Compléter le fichier *LanceSimFK.m* de façon à tracer
  - Pour chaque composante de l'état, l'erreur d'estimation
  - L'évolution des composantes du gain de Kalman, *K*

- Évaluer le comportement de ce filtre lorsque le système réel est simulé avec ***SimLin.mdl*** (Commande : *sim('SimLin',ParamReels.tfin) ;*). Testez différentes valeurs de X00 et des autres paramètres.

- Évaluer le comportement de ce filtre lorsque le système réel est simulé avec ***SimNL.mdl*** (Commande : ***sim('SimNL',ParamReels.tfin)***). Conclusions ?

D MISE EN ŒUVRE D'UN FILTRE DE KALMAN ETENDU

Il s'agit de mettre en œuvre un filtre de Kalman étendu en se basant sur le modèle non linéaire **(1)**. Pour cela, on complètera le fichier *LanceSimFKE.m*.

- *Exposer le principe d'un tel filtrage*

Remarques

✓ Le fichier ***LanceSimFKE.m*** permet de simuler le système réel et d'estimer l'état à l'aide d'un filtre de Kalman étendu. On notera l'utilisation de la fonction ***ode45*** (dans la fonction ***IntegEtat.m***) pour l'intégration numérique de l'équation d'état.

✓ La représentation d'état non linéaire obtenue ne se prête pas directement à l'utilisation de l'algorithme du Filtre de Kalman Etendu vu en cours. En effet, cet algorithme suppose d'avoir obtenu une représentation d'état à temps discret, or, nous ne disposons que d'une représentation d'état à temps continu **(1).** Il est difficile d'intégrer analytiquement l'équation **(1)**. Par conséquent, ce paragraphe décrit les modifications à apporter à l'algorithme du Filtre de Kalman Etendu afin de pouvoir estimer l'état du système considéré.

✓ Dans l'algorithme classique du filtre de Kalman étendu, la prédiction $\hat{x}_{k+1|k}$ générée par le filtre est obtenue en propageant l'estimé $\hat{x}_{k|k}$ à travers l'équation d'état non bruitée entre les instants $t = k\,T_e$ et $t = (k+1)T_e$. De façon similaire, dans le cas que nous considérons, $\hat{x}_{k+1|k}$, est la solution $x(t = (k+1)T_e)$, de l'équation différentielle **(1)** intégrée entre les instants $t = k\,T_e$ et $t = (k+1)T_e$ avec la condition initiale $x(t = kT_e) = \hat{x}_{k|k}$. Bien qu'il soit difficile d'intégrer cette équation analytiquement, à l'aide de MATLAB, elle peut facilement être intégrée numériquement. Ceci est peut-être réalisé à l'aide de la fonction *IntegEtat.m* (voir utilisation dans le §B).

✓ Afin de calculer la matrice $P_{k+1|k}$, l'équation **(1)** est linéarisée puis discrétisée. En effet, en linéarisant **(1)** autour de $\hat{x}_{k|k}$, on obtient une équation d'état continue de la forme :

$$\dot{\underline{x}} = A\,\underline{x} + Bu$$

Les matrices A et B peuvent être calculées à l'aide de la fonction *Linearise.m* (voir utilisation dans le §B).

✓ On discrétise alors cette équation et on obtient :

$$\underline{x}_{k+1} = F_k\ \underline{x}_k + G_k \underline{u}_k$$

Les matrices F_k et G_k pourraient être calculées grâce à la commande *c2d*. Cependant, afin de réduire les temps de calculs, il est préférable de calculer la matrice F_k à l'aide de la formule $F_k = \exp(A\,T_e)$, ce qui se traduit par la commande Matlab :

ParamFilt.F=expm(ParamFilt.Te*A);.

✓ Le calcul de la matrice G_k n'est pas utile pour calculer $P_{k+1|k}$. La matrice $P_{k+1|k}$ peut alors être calculée au moyen de l'équation de prédiction classique du Filtre de Kalman.

✓ L'équation de sortie étant linéaire, l'étape de correction est identique à celle de l'algorithme du Filtre de Kalman.

TRAVAIL :

- Compléter le fichier ***LanceSimFKE.m*** de façon à mettre en œuvre un filtre de Kalman Étendu.
- Compléter le fichier *LanceSimFKE.m* de façon à tracer
  - Pour chaque composante de l'état, sur une même figure, l'état réel et l'état estimé
  - Pour chaque composante de l'état, l'erreur d'estimation
  - L'évolution des composantes diagonales de P_{kk}.
  - L'évolution des composantes du gain de Kalman, K
- Évaluer le comportement de ce filtre. Testez différentes valeurs de X00 et des autres paramètres. Comparez avec le filtre mis en œuvre dans le §C

E ÉVALUATION DU FILTRE DE KALMAN ÉTENDU PAR DES SIMULATIONS DE MONTE-CARLO

Nous allons dans cette partie évaluer les performances de ce filtre de Kalman étendu à l'aide de simulations de Monte-Carlo.

Travail

- Compléter le fichier ***LanceMonteCarloFKE.m*** de façon à effectuer des simulations de Monte-Carlo pour évaluer le Filtre de Kalman étendu mis en œuvre dans le paragraphe précédent.
- On remarquera que l'état initial réel varie d'une simulation à l'autre. En effet, il est généré de façon à être gaussien d'espérance ***ParamReels.X0Vrai*** et de covariance ***ParamReels.P00***. Ceci est réalisé à l'aide de l'instruction :

  *ParamReels.X0=ParamReels.X0Vrai+sqrtm(ParamReels.P00)*randn(*2,1*);*

- Compléter le fichier *LanceMonteCarloFKE.m* de façon à tracer :
  - Pour chaque composante de l'état, sur une même figure : l'évolution de la moyenne empirique de l'erreur d'estimation, l'évolution du maximum et du minimum de l'erreur d'estimation
  - Pour chaque composante de l'état, sur une même figure, la covariance empirique de l'erreur d'estimation et la moyenne sur toutes les simulations de la composante diagonale correspondante de la matrice P_{kk}.
- Évaluer le comportement de ce filtre. Testez différentes valeurs de l'estimé initial et des autres paramètres.