

Implémentation de filtres/régulateurs

à la recherche d'une réalisation optimale

THIBAUT HILAIRE

R2D2 (IRISA) - Lannion
IUT de Lannion

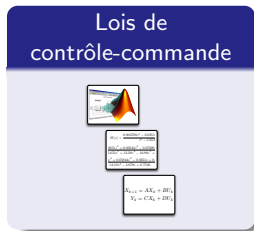
Présentation équipe Arénaire

- Thèse CIFRE IRCCyN / PSA Peugeot Citroën
Analyse et synthèse de l'implémentation des lois de contrôle-commande en précision finie
- Postdoc INRIA - R2D2
Évaluation de la précision des calculs en virgule fixe des systèmes de contrôle-commande

Plan

- 1 Problématique
 - Contexte
 - 2 approches
 - Exemple
- 2 Approche automatique
- 3 Approche Informatique
- 4 Synthèse
- 5 Conclusion

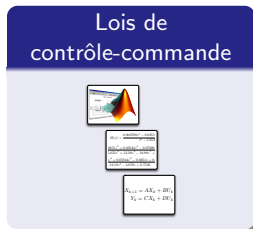
Contexte



Implémentation des systèmes de contrôle-commande

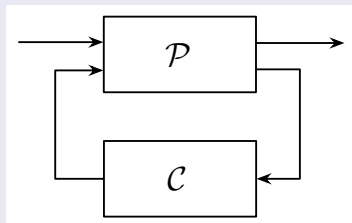
- Implémentation virgule fixe
- contrôle-commande
 - systèmes rebouclés
 - critères spécifiques (stabilité, ...)
 - systèmes *Linéaires à Paramètres Invariants*

Contexte



Implémentation des systèmes de contrôle-commande

- Implémentation virgule fixe
- contrôle-commande
 - systèmes rebouclés
 - critères spécifiques (stabilité, ...)
 - systèmes *Linéaires à Paramètres Invariants*



Contexte

Problématique *numérique*

L'implémentation de la loi entraîne une dégradation

- modification des caractéristiques
 - modification des performances
-
- Comment évaluer *a priori* la dégradation ?
 - Comment comparer différentes réalisations d'une même loi ?
 - Quelles réalisations/algorithmes sont à considérer ? Avec quel coût de calcul ?

Contexte

Problématique *numérique*

L'implémentation de la loi entraîne une dégradation

- modification des caractéristiques
 - modification des performances
-
- Comment évaluer *a priori* la dégradation ?
 - Comment comparer différentes réalisations d'une même loi ?
 - Quelles réalisations/algorithmes sont à considérer ? Avec quel coût de calcul ?

Sources de dégradations

Sources de dégradations

La dégradation *Précision Finie* a deux origines :

- Bruits de quantification qui apparaissent dans les calculs, dus aux arrondis
→ *bruits numériques*
- Quantification des coefficients mis en jeu
→ *erreurs paramétriques*

Sources de dégradations

Sources de dégradations

La dégradation *Précision Finie* a deux origines :

- Bruits de quantification qui apparaissent dans les calculs, dus aux arrondis
→ *bruits numériques*
- Quantification des coefficients mis en jeu
→ *erreurs paramétriques*

Une infinité de réalisations numériques

Pour une loi de contrôle-commande donnée, il existe une infinité de réalisations logicielles/matérielles possibles pour la mettre en œuvre

Choix de l'algorithme

Il existe de nombreuses structures de calcul (formes directes, découpage en cascade, forme d'état, opérateur δ , etc...)

→ *Pour une même structure, une infinité de paramètres possibles*

Implémentation logicielle/matérielle

Pour un algorithme donné, il existe de nombreuses possibilités d'implémentation :

→ *Des choix logiciels (position de la virgule, recadrage, représentation des coefficients)*

→ *Des choix matériels fonction de la cible (nb de bits, ...)*

Une infinité de réalisations numériques

Pour une loi de contrôle-commande donnée, il existe une infinité de réalisations logicielles/matérielles possibles pour la mettre en œuvre

Choix de l'algorithme

Il existe de nombreuses structures de calcul (formes directes, découpage en cascade, forme d'état, opérateur δ , etc...)

→ *Pour une même structure, une infinité de paramètres possibles*

Implémentation logicielle/matérielle

Pour un algorithme donné, il existe de nombreuses possibilités d'implémentation :

→ *Des choix logiciels (position de la virgule, recadrage, représentation des coefficients)*

→ *Des choix matériels fonction de la cible (nb de bits, ...)*

Communauté *automatique*

- principalement focalisée sur la forme d'état
- création de mesures *a priori*
 - on ne préjuge pas de l'implémentation matérielle/logicielle
 - on évalue la **sensibilité** à la dégradation
- on cherche, parmi les réalisations équivalentes, celle qui minimise cette évaluation *a priori*
- exploration de quelques structures particulières (opérateur δ , etc.)

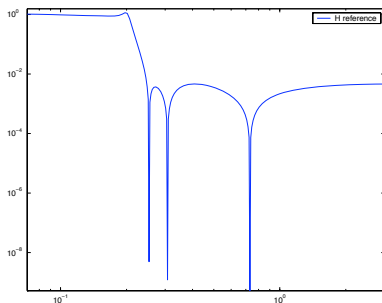
Communauté *informatique embarquée*

- l'algorithme utilisé est une forme directe I (équation aux différences)
- mesure *a posteriori* :
 - bruit de quantification (RSBQ)
 - coût matériel/logiciel (temps, surface, consommation)
- on cherche une implémentation qui minimise le coût logiciel/matériel pour un RSBQ donné

Exemple

On considère l'exemple suivant [Will86] (filtre passe bas, à bande étroite)

$$H(z) = \frac{0.004708z^6 - 0.0251z^5 + 0.05844z^4 - 0.07608z^3 + 0.05844z^2 - 0.0251z + 0.004708}{z^6 - 5.653z^5 + 13.38z^4 - 16.98z^3 + 12.18z^2 - 4.679z + 0.7526}$$



Exemple

Il existe de nombreuses façons de réaliser numériquement ce filtre

- forme directe I
- forme directe II
- forme d'état (choix d'une base pour l'état)
- utilisation de l'opérateur δ
- décomposition cascade, parallèle
- ...

$$H(z) = \frac{\sum_{i=0}^n b_i z^{-i}}{\sum_{i=0}^n a_i z^{-i}}$$

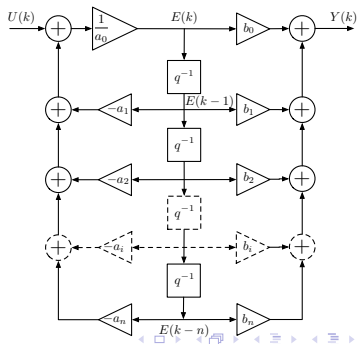
$$Y(k) = \sum_{i=0}^n b_i U(k-i) - \sum_{i=1}^n a_i Y(k-i)$$

Exemple

Il existe de nombreuses façons de réaliser numériquement ce filtre

- forme directe I
- forme directe II
- forme d'état (choix d'une base pour l'état)
- utilisation de l'opérateur δ
- décomposition cascade, parallèle
- ...

$$H(z) = \frac{\sum_{i=0}^n b_i z^{-i}}{\sum_{i=0}^n a_i z^{-i}}$$



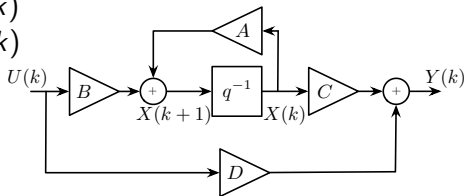
Exemple

Il existe de nombreuses façons de réaliser numériquement ce filtre

- forme directe I
- forme directe II
- forme d'état (choix d'une base pour l'état)
- utilisation de l'opérateur δ
- décomposition cascade, parallèle
- ...

$$\begin{cases} q[X(k)] = AX(k) + BU(k) \\ Y(k) = CX(k) + DU(k) \end{cases}$$

$$q[X(k)] \triangleq X(k+1)$$

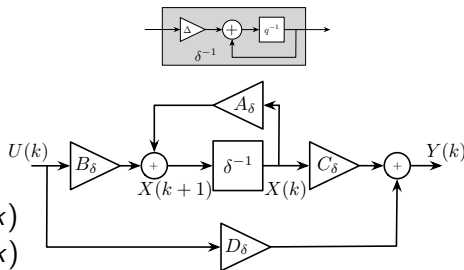


Exemple

Il existe de nombreuses façons de réaliser numériquement ce filtre

- forme directe I
- forme directe II
- forme d'état (choix d'une base pour l'état)
- utilisation de l'opérateur δ
- décomposition cascade, parallèle
- ...

$$\delta \triangleq \frac{q-1}{\Delta}$$



$$\begin{cases} \delta[X(k)] &= AX(k) + BU(k) \\ Y(k) &= CX(k) + DU(k) \end{cases}$$

Exemple

Il existe de nombreuses façons de réaliser numériquement ce filtre

- forme directe I
- forme directe II
- forme d'état (choix d'une base pour l'état)
- utilisation de l'opérateur δ
- décomposition cascade, parallèle
- ...

Exemple

Il existe de nombreuses façons de réaliser numériquement ce filtre

- forme directe I
- forme directe II
- forme d'état (choix d'une base pour l'état)
- utilisation de l'opérateur δ
- décomposition cascade, parallèle
- ...

Chacune de ces réalisations va utiliser une paramétrisation différente.

L'impact de l'implémentation (quantification des coefficients et bruits numériques) sera alors différent.

Exemple

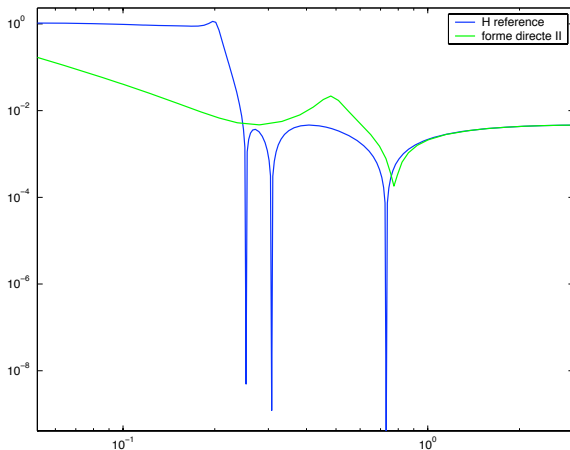


Diagramme de Bode avec coefficients virgule fixe sur 11 bits

Exemple

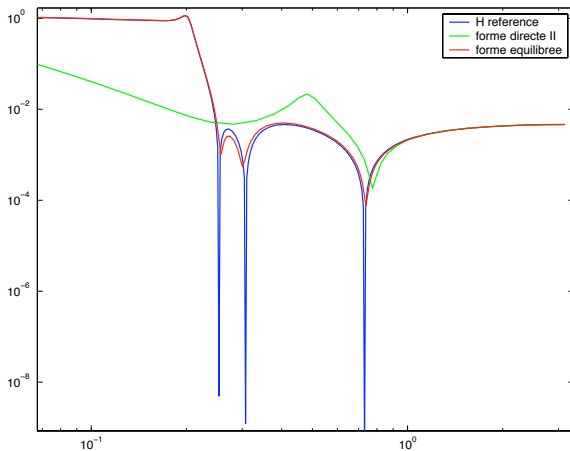


Diagramme de Bode avec coefficients virgule fixe sur 11 bits

Exemple

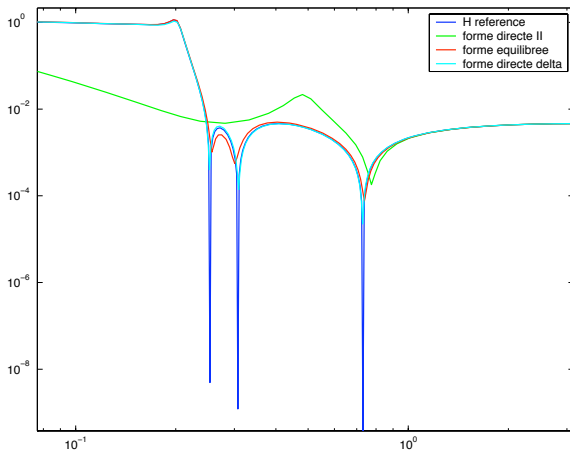


Diagramme de Bode avec coefficients virgule fixe sur 11 bits

Exemple

Comparaison *détérioration* / *coût de calcul*

Réalisation	$\ H - H^\dagger\ _2$	Nb Opérations
Forme directe II	$1.90e + 10$	$13 \times 12+$
Forme équilibrée	$3.99e + 2$	$49 \times 42+$
Forme directe en δ	$1.66e + 2$	$19 \times 18+$

Plan

- 1 Problématique
- 2 Approche automatique
 - Forme implicite
 - Exemples de réalisations
 - Mesures de sensibilité
 - Mesures
- 3 Approche Informatique
- 4 Synthèse
- 5 Conclusion

Formalisme unificateur

Comment modéliser toutes ces réalisations ?

Mes travaux précédents on portés sur

- un formalisme permettant de décrire toutes les réalisations/algorithmes de lois *linéaires à paramètres invariants*
 - écriture matricielle (dérivée de la forme d'état) qui décrit une suite de produits scalaires et blocs retard q^{-1}
 - description macroscopique
 - faisant apparaître les variables intermédiaires utilisées et les variables stockées
 - mais directement lié à l'algorithme
- la description des classes d'équivalence

Formalisme unificateur

Comment modéliser toutes ces réalisations ?

Mes travaux précédents ont porté sur

- un formalisme permettant de décrire toutes les réalisations/algorithmes de lois *linéaires à paramètres invariants*
 - écriture matricielle (dérivée de la forme d'état) qui décrit une suite de produits scalaires et blocs retard q^{-1}
 - description macroscopique
 - faisant apparaître les variables intermédiaires utilisées et les variables stockées
 - mais directement lié à l'algorithme
- la description des classes d'équivalence

Forme implicite spécialisée

On se propose de décrire les algorithmes sous la forme [Hila05a]

- $J.T_{k+1} = M.X_k + N.U_k$
- $X_{k+1} = K.T_{k+1} + P.X_k + Q.U_k$
- $Y_k = L.T_{k+1} + R.X_k + S.U_k$

Forme implicite spécialisée

$$\begin{pmatrix} J & 0 & 0 \\ -K & I & 0 \\ -L & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

Forme implicite spécialisée

On se propose de décrire les algorithmes sous la forme [Hila05a]

- 1 $J.T_{k+1} = M.X_k + N.U_k$
- 2 $X_{k+1} = K.T_{k+1} + P.X_k + Q.U_k$
- 3 $Y_k = L.T_{k+1} + R.X_k + S.U_k$

Calculs des variables intermédiaires

Forme implicite spécialisée

$$\begin{pmatrix} J & 0 & 0 \\ -K & I & 0 \\ -L & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

Forme implicite spécialisée

On se propose de décrire les algorithmes sous la forme [Hila05a]

- 1 $J.T_{k+1} = M.X_k + N.U_k$
- 2 $X_{k+1} = K.T_{k+1} + P.X_k + Q.U_k$
- 3 $Y_k = L.T_{k+1} + R.X_k + S.U_k$

Calcul de l'état

Forme implicite spécialisée

$$\begin{pmatrix} J & 0 & 0 \\ -K & I & 0 \\ -L & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

Forme implicite spécialisée

On se propose de décrire les algorithmes sous la forme [Hila05a]

- 1 $J.T_{k+1} = M.X_k + N.U_k$
- 2 $X_{k+1} = K.T_{k+1} + P.X_k + Q.U_k$
- 3 $Y_k = L.T_{k+1} + R.X_k + S.U_k$

Calcul des sorties

Forme implicite spécialisée

$$\begin{pmatrix} J & 0 & 0 \\ -K & I & 0 \\ -L & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

Forme implicite spécialisée

On se propose de décrire les algorithmes sous la forme [Hila05a]

- 1 $J.T_{k+1} = M.X_k + N.U_k$
- 2 $X_{k+1} = K.T_{k+1} + P.X_k + Q.U_k$
- 3 $Y_k = L.T_{k+1} + R.X_k + S.U_k$

Forme implicite spécialisée

$$\begin{pmatrix} J & 0 & 0 \\ -K & I & 0 \\ -L & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

Variables intermédiaires

Les variables intermédiaires introduites permettent

- de faire apparaître explicitement tous les calculs réalisés, ainsi que les variables intermédiaires utilisées
- d'indiquer l'ordonnancement des calculs
- autorisent une paramétrisation plus riche

Forme implicite

La forme présentée est implicite

- l'état ou la sortie peuvent être calculés à partir de variables intermédiaires (calculées lors de la même itération)
- une variable intermédiaire peut être calculée à partir d'une autre variable intermédiaire préalablement calculée (ordonnancement des calculs)

le calcul de T_{k+1} s'écrit $J.T_{k+1} = M.X_k + N.U_k$

$$\text{avec } J = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ \star & \ddots & 0 & & \vdots \\ \vdots & \star & 1 & 0 & \vdots \\ \vdots & & \star & \ddots & 0 \\ \star & \dots & \dots & \star & 1 \end{pmatrix}$$

Opérateur δ

Une réalisation avec l'opérateur δ est donnée par :

$$\begin{cases} \delta[X_k] &= A_\delta X_k + B_\delta U_k \\ Y_k &= C_\delta X_k + D_\delta U_k \end{cases} \quad \delta \triangleq \frac{q-1}{\Delta}$$

et il correspond à la forme implicite suivante :

$$\begin{pmatrix} I & 0 & 0 \\ -\Delta I & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & A_\delta & B_\delta \\ 0 & I & 0 \\ 0 & C_\delta & D_\delta \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

Opérateur δ

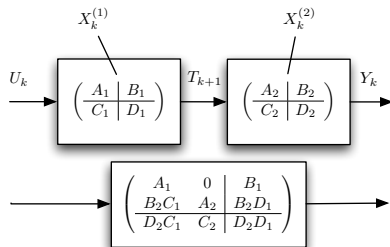
Une réalisation avec l'opérateur δ est donnée par :

$$\begin{cases} \delta[X_k] &= A_\delta X_k + B_\delta U_k \\ Y_k &= C_\delta X_k + D_\delta U_k \end{cases} \quad \delta \triangleq \frac{q-1}{\Delta}$$

et il correspond à la forme implicite suivante :

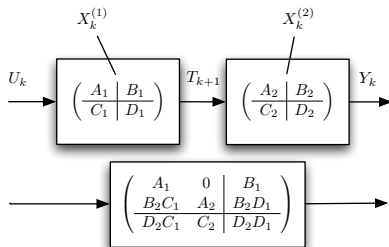
$$\begin{pmatrix} I & 0 & 0 \\ -\Delta I & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & A_\delta & B_\delta \\ 0 & I & 0 \\ 0 & C_\delta & D_\delta \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

Forme cascade



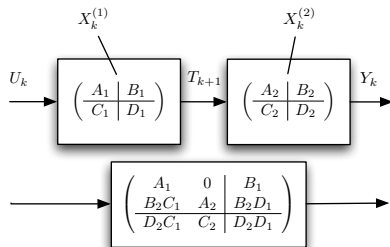
$$\begin{pmatrix} I & 0 & 0 \\ \begin{pmatrix} 0 \\ -B_2 \\ -D_2 \end{pmatrix} & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ \begin{pmatrix} X_k^{(1)} \\ X_k^{(2)} \\ Y_k \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 & \begin{pmatrix} C_1 & 0 \end{pmatrix} & D_1 \\ 0 & \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} & \begin{pmatrix} B_1 \\ 0 \end{pmatrix} \\ 0 & \begin{pmatrix} 0 & C_2 \end{pmatrix} & 0 \end{pmatrix} \begin{pmatrix} T_k \\ \begin{pmatrix} X_k^{(1)} \\ X_k^{(2)} \\ U_k \end{pmatrix} \end{pmatrix}$$

Forme cascade



$$\begin{pmatrix} I & 0 & 0 \\ \begin{pmatrix} 0 \\ -B_2 \\ -D_2 \end{pmatrix} & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{T}_{k+1} \\ X_k^{(1)} \\ X_k^{(2)} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & \begin{pmatrix} \mathbf{C}_1 & 0 \end{pmatrix} \\ 0 & \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} \\ 0 & \begin{pmatrix} 0 & C_2 \end{pmatrix} \end{pmatrix} \begin{pmatrix} \mathbf{D}_1 \\ \begin{pmatrix} B_1 \\ 0 \\ 0 \end{pmatrix} \end{pmatrix} \begin{pmatrix} T_k \\ X_k^{(1)} \\ X_k^{(2)} \\ U_k \end{pmatrix}$$

Forme cascade



$$\begin{pmatrix} I & 0 & 0 \\ \begin{pmatrix} 0 \\ -B_2 \\ -D_2 \end{pmatrix} & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ \begin{pmatrix} X_k^{(1)} \\ X_k^{(2)} \\ Y_k \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 & \begin{pmatrix} C_1 & 0 \end{pmatrix} & D_1 \\ 0 & \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} & \begin{pmatrix} B_1 \\ 0 \\ 0 \end{pmatrix} \\ 0 & \begin{pmatrix} 0 & C_2 \end{pmatrix} & 0 \end{pmatrix} \begin{pmatrix} T_k \\ \begin{pmatrix} X_k^{(1)} \\ X_k^{(2)} \\ U_k \end{pmatrix} \end{pmatrix}$$

Forme retour d'état / observateur

La forme retour d'état / observateur

$$\begin{cases} \hat{X}_{k+1} &= A_p \hat{X}_k + B_p U_k + K_f (Y_k - C_p \hat{X}_k) \\ U_k &= -K_c \hat{X}_k + Q (Y_k - C_p \hat{X}_k) \end{cases}$$

où (A_p, B_p, C_p) correspondent au système
et K_c, K_f et Q sont les paramètres du régulateur.

Une paramétrisation

$$\begin{pmatrix} \begin{pmatrix} I & 0 \\ -Q & I \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{pmatrix} \begin{pmatrix} T_{k+1}^{(1)} \\ T_{k+1}^{(2)} \\ \hat{X}_{k+1} \\ U_k \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} -C_p \\ -K_c \\ A_p \\ 0 \end{pmatrix} & \begin{pmatrix} I \\ 0 \\ 0 \end{pmatrix} \end{pmatrix} \begin{pmatrix} T_k^{(1)} \\ T_k^{(2)} \\ \hat{X}_k \\ Y_k \end{pmatrix}$$

Forme retour d'état / observateur

La forme retour d'état / observateur

$$\begin{cases} \hat{X}_{k+1} &= A_p \hat{X}_k + B_p U_k + K_f (Y_k - C_p \hat{X}_k) \\ U_k &= -K_c \hat{X}_k + Q (Y_k - C_p \hat{X}_k) \end{cases}$$

où (A_p, B_p, C_p) correspondent au système
et K_c, K_f et Q sont les paramètres du régulateur.

Une paramétrisation

$$\begin{pmatrix} \begin{pmatrix} I & 0 \\ -Q & I \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{pmatrix} \begin{pmatrix} T_{k+1}^{(1)} \\ T_{k+1}^{(2)} \\ \hat{X}_{k+1} \\ U_k \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} -C_p \\ -K_c \\ A_p \\ 0 \end{pmatrix} \begin{pmatrix} I \\ 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} T_k^{(1)} \\ T_k^{(2)} \\ \hat{X}_k \\ Y_k \end{pmatrix}$$

Forme retour d'état / observateur

La forme retour d'état / observateur

$$\begin{cases} \hat{X}_{k+1} &= A_p \hat{X}_k + B_p U_k + K_f (Y_k - C_p \hat{X}_k) \\ U_k &= -K_c \hat{X}_k + Q (Y_k - C_p \hat{X}_k) \end{cases}$$

où (A_p, B_p, C_p) correspondent au système
et K_c, K_f et Q sont les paramètres du régulateur.

Une paramétrisation

$$\begin{pmatrix} \begin{pmatrix} I & 0 \\ -Q & I \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{pmatrix} \begin{pmatrix} T_{k+1}^{(1)} \\ T_{k+1}^{(2)} \\ \hat{X}_{k+1} \\ U_k \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} -C_p \\ -K_c \\ A_p \\ 0 \end{pmatrix} & \begin{pmatrix} I \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{pmatrix} \begin{pmatrix} T_k^{(1)} \\ T_k^{(2)} \\ \hat{X}_k \\ Y_k \end{pmatrix}$$

Forme retour d'état / observateur

La forme retour d'état / observateur

$$\begin{cases} \hat{X}_{k+1} &= A_p \hat{X}_k + B_p U_k + K_f (Y_k - C_p \hat{X}_k) \\ U_k &= -K_c \hat{X}_k + Q (Y_k - C_p \hat{X}_k) \end{cases}$$

où (A_p, B_p, C_p) correspondent au système
et K_c, K_f et Q sont les paramètres du régulateur.

Une paramétrisation

$$\begin{pmatrix} \begin{pmatrix} I & 0 \\ -Q & I \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{pmatrix} \begin{pmatrix} T_{k+1}^{(1)} \\ T_{k+1}^{(2)} \\ \hat{X}_{k+1} \\ U_k \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} -C_p \\ -K_c \\ A_p \\ 0 \end{pmatrix} & \begin{pmatrix} I \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{pmatrix} \begin{pmatrix} T_k^{(1)} \\ T_k^{(2)} \\ \hat{X}_k \\ Y_k \end{pmatrix}$$

Forme retour d'état / observateur

La forme retour d'état / observateur

$$\begin{cases} \hat{X}_{k+1} &= A_p \hat{X}_k + B_p U_k + K_f (Y_k - C_p \hat{X}_k) \\ U_k &= -K_c \hat{X}_k + Q (Y_k - C_p \hat{X}_k) \end{cases}$$

où (A_p, B_p, C_p) correspondent au système
et K_c, K_f et Q sont les paramètres du régulateur.

Une paramétrisation

$$\begin{pmatrix} \begin{pmatrix} I & 0 \\ -Q & I \\ -K_f & -B_p \\ 0 & -I \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ I \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \\ I \end{pmatrix} \end{pmatrix} \begin{pmatrix} T_{k+1}^{(1)} \\ T_{k+1}^{(2)} \\ \hat{X}_{k+1} \\ U_k \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} -C_p \\ -K_c \\ A_p \\ 0 \end{pmatrix} \begin{pmatrix} I \\ 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} T_k^{(1)} \\ T_k^{(2)} \\ \hat{X}_k \\ Y_k \end{pmatrix}$$

Critères d'analyse

Des *critères génériques de qualité* permettant d'évaluer l'impact de l'implémentation ont été mis en place.

Résilience (mesures *a priori*)

- sensibilité à la quantification des coefficients
 - sensibilité entrée/sortie (fonction de transfert)
 - sensibilité pôles et zéros (mesure de distance à l'instabilité)
- développés pour les cas
 - boucle ouverte/boucle fermée
 - systèmes Multiples Entrées Multiples Sorties
 - virgule fixe, virgule flottante, représentation par blocs

Sensibilité paramétrique

Une mesure utilisée est la mesure de sensibilité de la fonction de transfert (de la loi à implémenter) vis à vis des coefficients mis en jeu.

1^{ère} mesure sensibilité

$$M_{L_2}^1 \triangleq \sum_{X \in \{J, K, L, M, N, P, Q, R, S\}} \left\| \frac{\partial H}{\partial X} \right\|_2^2$$

avec $\|\cdot\|_2$ la norme L_2

Sensibilité paramétrique

- Il ne faut pas prendre en compte les coefficients triviaux
 - $0, \pm 1$: dans la forme implicite spécialisée, de nombreux coefficients sont nuls ou égaux à 1
 - Certains coefficients (des puissances de 2, ...) peuvent être représentés exactement
- À une matrice X de coefficients, on associe une matrice de pondération W_X telle que

$$(W_X)_{i,j} = \begin{cases} 0 & \text{si } X_{i,j} \text{ est exactement implémenté} \\ 1 & \text{sinon} \end{cases}$$

Sensibilité paramétrique

- Il ne faut pas prendre en compte les coefficients triviaux
 - $0, \pm 1$: dans la forme implicite spécialisée, de nombreux coefficients sont nuls ou égaux à 1
 - Certains coefficients (des puissances de 2, ...) peuvent être représentés exactement
- À une matrice X de coefficients, on associe une matrice de pondération W_X telle que

$$(W_X)_{i,j} = \begin{cases} 0 & \text{si } X_{i,j} \text{ est exactement implémenté} \\ 1 & \text{sinon} \end{cases}$$

Sensibilité paramétrique

Sensibilité paramétrique pondérée [Hila05b]

pour une loi SISO, on obtient une mesure pondérée

$$M_{L_2}^W \triangleq \sum_{X \in \{J, K, L, M, N, P, Q, R, S\}} \left\| \frac{\partial H}{\partial X} \times W_X \right\|_2^2$$

On peut aussi l'écrire (pour faciliter les calculs)

$$M_{L_2}^W = \left\| \frac{\partial H}{\partial Z} \times W_Z \right\|_2^2 \quad \text{avec} \quad Z \triangleq \begin{pmatrix} -J & M & N \\ K & P & Q \\ L & R & S \end{pmatrix}$$

Z : matrice système généralisée

Sensibilité paramétrique

Sensibilité paramétrique pondérée [Hila05b]

pour une loi SISO, on obtient une mesure pondérée

$$M_{L_2}^W \triangleq \sum_{X \in \{J, K, L, M, N, P, Q, R, S\}} \left\| \frac{\partial H}{\partial X} \times W_X \right\|_2^2$$

On peut aussi l'écrire (pour faciliter les calculs)

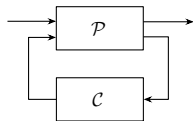
$$M_{L_2}^W = \left\| \frac{\partial H}{\partial Z} \times W_Z \right\|_2^2 \quad \text{avec} \quad Z \triangleq \begin{pmatrix} -J & M & N \\ K & P & Q \\ L & R & S \end{pmatrix}$$

Z : matrice système généralisée

Mesure de stabilité

La quantification des coefficients d'un régulateur peut amener le système en boucle fermée à devenir instable.

Il faut prendre en compte la sensibilité des pôles $(\lambda_k)_{1 \leq k \leq n}$ vis-à-vis des coefficients, ainsi que la distance à l'instabilité.



Mesure de stabilité [Hila06b]

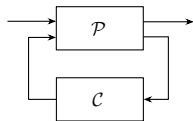
$$\mu(Z) \triangleq \min_{1 \leq k \leq n} \frac{1 - |\lambda_k|}{\left\| \frac{\partial |\lambda_k|}{\partial Z} \times W_Z \right\|_S}$$

On peut relier cette mesure au nombre de bits minimum requis (virgule fixe) pour assurer la stabilité.

Mesure de stabilité

La quantification des coefficients d'un régulateur peut amener le système en boucle fermée à devenir instable.

Il faut prendre en compte la sensibilité des pôles $(\lambda_k)_{1 \leq k \leq n}$ vis-à-vis des coefficients, ainsi que la distance à l'instabilité.



Mesure de stabilité [Hila06b]

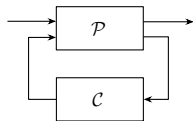
$$\mu(Z) \triangleq \min_{1 \leq k \leq n} \frac{1 - |\lambda_k|}{\left\| \frac{\partial |\lambda_k|}{\partial Z} \times W_Z \right\|_S}$$

On peut relier cette mesure au nombre de bits minimum requis (virgule fixe) pour assurer la stabilité.

Mesure de stabilité

La quantification des coefficients d'un régulateur peut amener le système en boucle fermée à devenir instable.

Il faut prendre en compte la sensibilité des pôles $(\lambda_k)_{1 \leq k \leq n}$ vis-à-vis des coefficients, ainsi que la distance à l'instabilité.



Mesure de stabilité [Hila06b]

$$\mu(Z) \triangleq \min_{1 \leq k \leq n} \frac{1 - |\lambda_k|}{\left\| \frac{\partial |\lambda_k|}{\partial Z} \times W_Z \right\|_S}$$

On peut relier cette mesure au nombre de bits minimum requis (virgule fixe) pour assurer la stabilité.

Plan

- 1 Problématique
- 2 Approche automatique
- 3 Approche Informatique
 - RSBQ
 - Produit scalaire
 - Forme implicite
 - Exemple d'algo virgule fixe
- 4 Synthèse
- 5 Conclusion

Approche Informatique

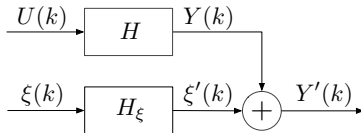
- On dispose d'un algorithme/réalisation
- On joue sur la façon d'implémenter
 - dépend de la cible
 - choix à faire (formats virgule fixe, ...)
- On fait un choix de schéma d'implémentation
- On déduit les représentations virgule fixe
- On peut évaluer formellement le RSBQ

Approche Informatique

- On dispose d'un algorithme/réalisation
- On joue sur la façon d'implémenter
 - dépend de la cible
 - choix à faire (formats virgule fixe, ...)
- On fait un choix de schéma d'implémentation
- On déduit les représentations virgule fixe
- On peut évaluer formellement le RSBQ

RSBQ

- Les quantifications dans les calculs amènent des bruits
⇒ caractéristiques statistiques connues
- $\xi(k)$ est le vecteur de bruits générés
- Le système est équivalent à un bruit $\xi'(k)$ sur la sortie



RSBQ

RSBQ

Le RSBQ est défini comme la puissance du bruit sur la sortie

$$\begin{aligned} RSBQ &= \sigma_{\xi'}^2 \\ &= \|H_{\xi} \varphi_{\xi}\|_2^2 \end{aligned}$$

φ_{ξ} tel que $\varphi_{\xi} \varphi_{\xi}^T = \psi_{\xi} = E \left\{ (\xi(k) - \mu_{\xi}) (\xi(k) - \mu_{\xi})^T \right\}$

- La fonction de transfert H_{ξ} s'exprime en fonction de la réalisation
- φ_{ξ} dépend de $\xi(k)$ (défini par l'implémentation)

RSBQ

RSBQ

Le RSBQ est défini comme la puissance du bruit sur la sortie

$$\begin{aligned} RSBQ &= \sigma_{\xi'}^2 \\ &= \|H_{\xi} \varphi_{\xi}\|_2^2 \end{aligned}$$

φ_{ξ} tel que $\varphi_{\xi} \varphi_{\xi}^T = \psi_{\xi} = E \left\{ (\xi(k) - \mu_{\xi}) (\xi(k) - \mu_{\xi})^T \right\}$

- La fonction de transfert H_{ξ} s'exprime en fonction de la réalisation
- φ_{ξ} dépend de $\xi(k)$ (défini par l'implémentation)

Cas du produit scalaire

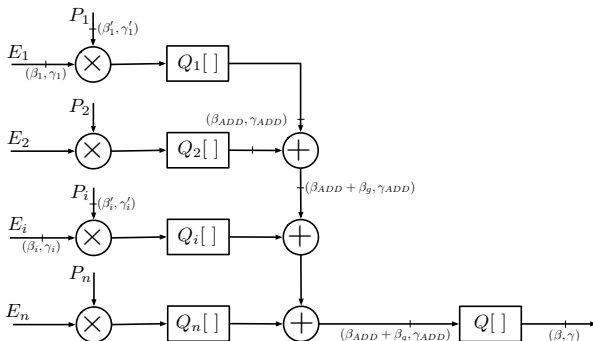
- On considère le produit scalaire

$$S = \sum_{i=1}^n P_i E_i$$

- Coefficients P_i connus
- Formats virgule fixe entrées/sorties E_i et S connus

On suppose que toutes les additions sont toutes réalisées sur le même format $(\beta_{ADD}, \gamma_{ADD})$

Cas du produit scalaire



2 options :

- Quantification *après* la multiplication
- Quantification renvoyée dans les coefficients

Cas du produit scalaire

On détermine :

- le format de l'addition
 - dépend de la dynamique de chaque produit
 - de la dynamique de la sortie
- la quantification *après* les multiplications
- la quantification des coefficients
- la quantification en sortie

Forme implicite

On peut généraliser pour la forme implicite.

On fixe :

- format virgule fixe en entrée et sortie
- le nombre de bits des variables intermédiaires, des variables stockées
- le nombre de bits des coefficients de Z
- le nombre de bits des additionneurs (bits de garde)
- le schéma d'implémentation

On en déduit :

- Tous les formats virgule fixe
- Toutes les quantifications
⇒ les moments de $\xi(k)$ et φ_ξ

Forme implicite

On peut généraliser pour la forme implicite.

On fixe :

- format virgule fixe en entrée et sortie
- le nombre de bits des variables intermédiaires, des variables stockées
- le nombre de bits des coefficients de Z
- le nombre de bits des additionneurs (bits de garde)
- le schéma d'implémentation

On en déduit :

- Tous les formats virgule fixe
- Toutes les quantifications
⇒ les moments de $\xi(k)$ et φ_ξ

Exemple

On reprend les 3 réalisations précédentes.

On considère une implémentation avec quantification ramenée dans les coefs

- Coefficients sur 16 bits
- Accumulateur 32 bits (4 bits de garde)

Format d'entrée (16,11)

Format de sortie (16,10)

Exemple

Code Forme Directe II

```
// intermediate variables
```

```
Acc ← xn(1) * 5788 + xn(2) * -13703 + xn(3) * 17387 + xn(4) * -12469 + xn(5) * 4791 + xn(6) * -771 + u(i) * 1;
```

```
xnp(1) ← Acc >> 10;
```

```
xnp(2) ← xn(1);
```

```
xnp(3) ← xn(2);
```

```
xnp(4) ← xn(3);
```

```
xnp(5) ← xn(4);
```

```
xnp(6) ← xn(5);
```

```
// outputs
```

```
Acc ← xn(1) * 6336 + xn(2) * -19119 + xn(3) * 16167 + xn(4) * 4681 + xn(5) * -12891 + xn(6) * 4886 + u(i) * 1; y(i) ← Acc >> 4;
```

```
// permutations
```

```
xn ← xnp;
```

Exemple

Code Forme Équilibrée

```
// intermediate variables
```

```
Acc ← xn(1) * 32370 + xn(2) * -3673 + Acc0 + xn(3) * 42 + xn(4) * 873 + xn(5) * -171 + xn(6) * 51 + u(i) * 458;
```

```
xnp(1) ← Acc >> 15;
```

```
Acc ← xn(1) * 3688 + xn(2) * 31858 + xn(3) * 4405 + xn(4) * 785 + xn(5) * -451 + xn(6) * 74 + u(i) * -605;
```

```
xnp(2) ← Acc >> 15;
```

```
Acc ← xn(1) * 318 + xn(2) * -4416 + xn(3) * 31250 + xn(4) * -3922 + xn(5) * 499 + xn(6) * -120 + u(i) * -790;
```

```
xnp(3) ← Acc >> 15;
```

```
Acc ← xn(1) * -900 + xn(2) * 546 + xn(3) * 3833 + xn(4) * 30304 + xn(5) * 1828 + xn(6) * -196 + u(i) * 742;
```

```
xnp(4) ← Acc >> 15;
```

```
Acc ← xn(1) * -551 + xn(2) * 1425 + xn(3) * 1760 + xn(4) * -7483 + xn(5) * 29956 + xn(6) * 1961 + u(i) * 868;
```

```
xnp(5) ← Acc >> 15;
```

```
Acc ← xn(1) * -786 + xn(2) * 1182 + xn(3) * 2572 + xn(4) * -4839 + xn(5) * -7995 + xn(6) * 29485 + u(i) * 956;
```

```
xnp(6) ← Acc >> 15;
```

```
// outputs
```

```
Acc ← xn(1) * 14733 + xn(2) * 21060 + xn(3) * -23783 + xn(4) * -22615 + xn(5) * 7488 + xn(6) * -1780 + u(i) * 77;
```

```
y(i) ← Acc >> 15;
```

```
// permutations
```

```
xn ← xnp;
```

Exemple

Code Forme Directe II opérateur δ

```
// intermediate variables
```

```
Acc ← xn(1) * -11383 + xn(2) * -31123 + xn(3) * -22773 + xn(4) * -13468 + xn(5) * -9425 + xn(6) * -1852 + u(i) << 8;
```

```
T0 ← Acc >> 13;
```

```
T1 ← xn(1);
```

```
T2 ← xn(2);
```

```
T3 ← xn(3);
```

```
T4 ← xn(4);
```

```
T5 ← xn(5);
```

```
// states
```

```
Acc ← T0 + xn(1) << 2;
```

```
xn(1) ← Acc >> 2;
```

```
Acc ← T1 + xn(2) << 3;
```

```
xn(2) ← Acc >> 3;
```

```
Acc ← T2 + xn(3) << 2;
```

```
xn(3) ← Acc >> 2;
```

```
Acc ← T3 + xn(4) << 2;
```

```
xn(4) ← Acc >> 2;
```

```
Acc ← T4 + xn(5) << 3;
```

```
xn(5) ← Acc >> 3;
```

```
Acc ← T5 + xn(6) << 2;
```

```
xn(6) ← Acc >> 2; // outputs
```

```
Acc ← xn(1) * 792 + xn(2) * 12559 + xn(3) * 12190 + xn(4) * 29211 + xn(5) * 22483 + xn(6) * 30784 + u(i) * 19;
```

```
y(i) ← Acc >> 13 );
```

Exemple

Code Forme Directe II opérateur δ : *Roundoff After Multiplication*

// intermediate variables

```
Acc ← (xn(1) * -22767) >> 1 + xn(2) * -31123 + xn(3) * -22773 + (xn(4) * -26936) >> 1 + (xn(5) * -18851)
>> 1 + (xn(6) * -29635) >> 4 + u(i) << 8;
```

T0 ← Acc >>13;

T1 ← xn(1);

T2 ← xn(2);

T3 ← xn(3);

T4 ← xn(4);

T5 ← xn(5);

// states

Acc ← T0 + xn(1) <<2;

xn(1) ← Acc >>2;

Acc ← T1 + xn(2) <<3;

xn(2) ← Acc >>3;

Acc ← T2 + xn(3) <<2;

xn(3) ← Acc >>2;

Acc ← T3 + xn(4) <<2;

xn(4) ← Acc >>2;

Acc ← T4 + xn(5) <<3;

xn(5) ← Acc >>3;

Acc ← T5 + xn(6) <<2;

xn(6) ← Acc >>2; // outputs

```
Acc ← (xn(1) * 25342) >> 5 + (xn(2) * 25118) >> 1 + (xn(3) * 24379) >> 1 + xn(4) * 29211 + xn(5) *
22483 + xn(6) * 30784 + (u(i) * 19746) >> 10;
```

y(i) ← Acc >>13);

Plan

- 1 Problématique
- 2 Approche automatique
- 3 Approche Informatique
- 4 Synthèse
 - Réalisations optimales
 - Classes d'équivalence
- 5 Conclusion

Réalisations optimales

Pour une matrice de transfert H donnée, on note \mathcal{R}_H l'ensemble des réalisations \mathcal{R} équivalentes, sans contrainte d'ordre.

Le problème de synthèse de réalisation optimale consiste à trouver, parmi \mathcal{R}_H , celles qui minimise un critère \mathcal{J} (significatif en précision finie).

$$\mathcal{R}^{\text{opt}} = \arg \min_{\mathcal{R} \in \mathcal{R}_H} \mathcal{J}(\mathcal{R})$$

\mathcal{R}_H étant trop large, on s'intéresse en pratique à un sous-ensemble, celui de réalisations équivalentes ayant une structure donnée (réalisations d'ordre minimal en q , δ , retour d'état observateur, ...)

Réalisations optimales

Pour une matrice de transfert H donnée, on note \mathcal{R}_H l'ensemble des réalisations \mathcal{R} équivalentes, sans contrainte d'ordre.

Le problème de synthèse de réalisation optimale consiste à trouver, parmi \mathcal{R}_H , celles qui minimise un critère \mathcal{J} (significatif en précision finie).

$$\mathcal{R}^{\text{opt}} = \arg \min_{\mathcal{R} \in \mathcal{R}_H} \mathcal{J}(\mathcal{R})$$

\mathcal{R}_H étant trop large, on s'intéresse en pratique à un sous-ensemble, celui de réalisations équivalentes ayant une structure donnée (réalisations d'ordre minimal en q , δ , retour d'état observateur, ...)

Classes d'équivalences

Le *Principe d'Inclusion* (Šiljak) permet de caractériser des classes d'équivalences de réalisations.

Il est possible de l'étendre à la forme implicite, et permet de décrire comment parcourir une classe d'équivalence.

Dans le cas simple où l'on se restreint à ne pas changer les dimensions du système, on peut parcourir la classe d'équivalence par une similarité sur Z

$$\begin{pmatrix} \mathcal{Y} & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix} Z \begin{pmatrix} \mathcal{W} & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix}$$

\mathcal{U} , \mathcal{Y} , \mathcal{W} inversibles

Exemple

On peut parcourir la classe d'équivalence des réalisations structurée en δ à partir de la donnée d'une réalisation

$$\mathcal{R}_H^\delta(Z_0) = \left\{ \begin{array}{l} Z = \begin{pmatrix} \mathcal{U}^{-1} & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix} Z_0 \begin{pmatrix} \mathcal{U} & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix} \\ \forall \mathcal{U} \text{ inversible} \end{array} \right\}$$

$$\mathcal{U}^{opt} = \arg \min_{\mathcal{U} \text{ inversible}} \mathcal{J}(\mathcal{U})$$

On utilise un algorithme d'optimisation globale (Recuit Simulé Adaptatif ASA) ou local.

Exemple

On peut parcourir la classe d'équivalence des réalisations structurée en δ à partir de la donnée d'une réalisation

$$\mathcal{R}_H^\delta(Z_0) = \left\{ \begin{array}{l} Z = \begin{pmatrix} \mathcal{U}^{-1} & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix} Z_0 \begin{pmatrix} \mathcal{U} & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix} \\ \forall \mathcal{U} \text{ inversible} \end{array} \right\}$$

$$\mathcal{U}^{opt} = \arg \min_{\mathcal{U} \text{ inversible}} \mathcal{J}(\mathcal{U})$$

On utilise un algorithme d'optimisation globale (Recuit Simulé Adaptatif ASA) ou local.

Exemple

Dans le cas d'une réalisation en forme d'état avec opérateur δ

$$\begin{cases} \delta[X_k] &= A_\delta X_k + B_\delta U_k \\ Y_k &= C_\delta X_k + D_\delta U_k \end{cases} \quad \delta \triangleq \frac{q-1}{\Delta}$$

on peut trouver une réalisation qui minimise la sensibilité
entrée/sortie

Exemple

Dans le cas d'une réalisation en forme d'état avec opérateur δ on peut trouver une réalisation qui minimise la sensibilité entrée/sortie

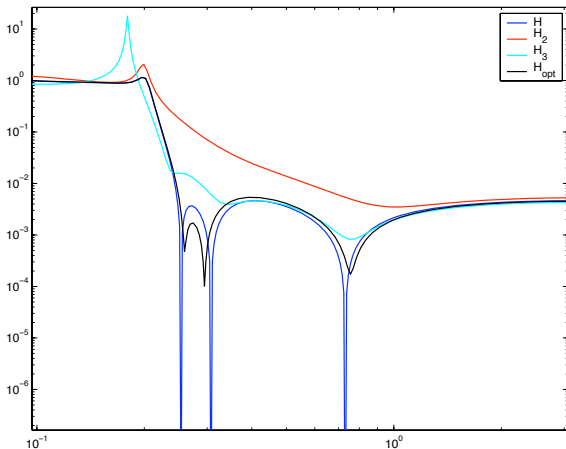


Diagramme de Bode avec coefficients virgule fixe sur 6 bits

Réalisations optimales

On peut rechercher une réalisation optimale selon des structures connues :

- découpage cascade ou parallèle
- forme d'état q, δ
- réalisations creuses (ρ DFilt, ...)
- ...

On minimise selon des critères de performance :

- sensibilité entrée/sortie
- sensibilité des pôles/zéros
- RSBQ

Développement d'une toolbox Matlab :

Finite Wordlength Realizations Toolbox

Réalisations optimales

On peut rechercher une réalisation optimale selon des structures connues :

- découpage cascade ou parallèle
- forme d'état q, δ
- réalisations creuses (ρ DFilt, ...)
- ...

On minimise selon des critères de performance :

- sensibilité entrée/sortie
- sensibilité des pôles/zéros
- RSBQ

Développement d'une toolbox Matlab :

Finite Wordlength Realizations Toolbox

Plan

- 1 Problématique
- 2 Approche automatique
- 3 Approche Informatique
- 4 Synthèse
- 5 Conclusion

Conclusions

Conclusion

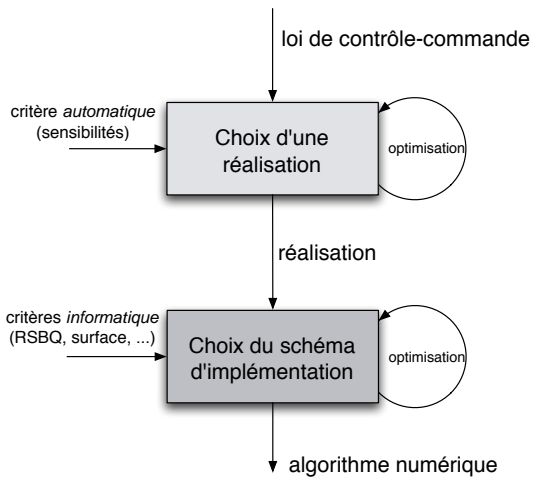
Mise en place d'outils pour répondre à la question

Comment implémenter *au mieux* de telles lois ?

à partir d'une description formelle

- Critères de performances, de précision numériques, de coût de calcul
- Itérations sur la structure (algorithmes/réalisations équivalentes)

Conclusions



Perspectives

Perspectives

- Itérations sur les paramètres de l'implémentation HW/SW
- coupler ces deux itérations dans une même démarche
- poursuivre l'effort logiciel (FWR Toolbox pour matlab)
- mettre en place d'autres schémas d'implémentations
- étendre à d'autres modèles de lois
- ...

